Proyecto

Coche teledirigido con RaspberryPi

DEF

(Documento Especificación Funcional)

Antonio Duce Gimeno

ALU.50301

1. Introducción

La finalidad principal del proyecto es estudiar la capacidad para controlar elementos físicos desde aplicaciones móviles, aplicaciones Web y aplicaciones de escritorio. Para esta finalidad usaremos un dispositivo (RaspberryPi) que nos permitirá interactuar con el medio físico mediante los puertos GPIO.

Inicialmente el estudio lo vamos a centrar en la capacidad de controlar los movimientos de un coche teledirigido, al cual dotaremos de diversos sensores ( Video, Proximidad ) que nos permitirán tener un conocimiento del entorno en el que estamos trabajando .

Estudiaremos de esta misma forma las distintas formas que tenemos de comunicarnos con el dispositivo que vamos a crear, transmitir ordenes y recibir información.

Podemos englobar el proyecto dentro de una categoría de vehículo autónomo, para ello construiremos un prototipo de coche teledirigido que sea capaz de reconocer el entorno en el que circula.

1. Requisitos funcionales

* Construir un dispositivo que tenga capacidad de movimiento
* El dispositivo creado tiene que ser capaz de obtener información del medio que le rodea
* Construir un sistema multi-plataforma que nos permita comunicarnos con el dispositivo
* El sistema tiene que ser capaz de comunicarse con el dispositivo para obtener y mandar información
* El sistema tiene que ser capaz de establecer una conexión con el dispositivo por medio de una red PAN
* El sistema tiene que ser capaz de establecer una conexión con el dispositivo por medio de una red LAN
* El sistema tiene que ser capaz de establecer la conexión con el dispositivo por medio de una red WAN.
* El sistema tiene que ser capaz de controlar los movimientos del dispositivo creado
* El dispositivo tiene que ser capaz de mandar información del medio que le rodea la sistema controlador.
* El dispositivo tiene que ser capaz de detectar obstáculos
* El dispositivo tiene que se capaz de calcular trayectorias para esquivar obstáculos.

1. Estudio General

Dentro de este campo, controlar elementos físicos desde un ordenador tenemos una gran variedad de proyectos desde casas domóticas, hasta coches autónomos sin conductor, pasando por multitud de proyectos industriales, todos ellos se basan en el mismo principio, obtener un control telemático del medio físico.

En este apartado pondremos varios ejemplos de uso y estudiaremos un poco mas o fondo como están desarrollados estos proyectos.

1. Vehículos Autónomos

Otro gran precedente dentro de este campo son los coche autónomos, coches que pueden conducir de forma autónoma sin necesidad de un conductor, se define este hecho como la capacidad que tiene un sistema de imitar las acciones humanas de manejo y control de un vehículo, por lo tanto un vehículo autónomo es capaz de percibir el medio que le rodea mediante diversos sensores y técnicas de navegación y actuar en consecuencia.

Últimamente se ha hablado mucho de un proyecto el cual parece que es el el principal representante de este campo, pero realmente aunque es una innovación aplicar automatismos para conseguir que un sistema sea capaz de tomar la gran cantidad de decisiones que implica conducir, ya existían varios proyectos prototipos sobre este tema.

Por ejemplo en la Exposición universal de 1939 se presento un coche eléctrico que conducía gracias a un circuito integrado en el pavimento.

En 1980 DARPA (Defense Advanced Research Projects Agency) hizo que un vehículo condujese mas de 600 metros por terreno abrupto fuera de mapa, gracias a un sistema de guía láser.

En 1994 los vehículos construidos por Daimler-Benz y Ernst Dickmans , condujeron por una autopista de 3 carriles en París mas de 1000 kilómetros con trafico intenso a una velocidad de 130km/h solo con pequeñas intervenciones humanas.

En 1995 El equipo Dickmans aplico un sistema de visión computerizada a su coche y permitió realizar un viaje Munich- Copenague ida y vuelta a mas de 175Km/h, con un 95% de conducción autónoma.

Dentro de este tema se puede destacar también el gran proyecto de Google de coche autónomo derivado de un proyecto ganadar de DARPA Grand challenge 2005 (Stanley) coche autónomo creado por el equipo de Sebastian Thrun ingerniero de Google y Director de Stanford Artificial Intelligence Laboratory.

Ilustración 1: Google Driverless Car.

El dotar de inteligencia a los automóviles es una linea que se ha ido reforzando con el paso del tiempo gracias a los avances en electrónica y computación, en realidad actualmente muchos de los coches actuales ya implementan sistemas inteligentes. El principal objetivo de dotar de esta inteligencia a los automóviles se basa en erradicar el error humano consiguiendo de estar forma vehículos capaces de circular de forma autónoma.

Aunque la tecnología ha avanzado mucho en este aspecto, aun dista mucho el objetivo de tener vehículos autónomos circulando libremente. Por este motivo actualmente el desarrollo de vehiculos inteligentes de forma comercial esta centrado en los **sistemas avanzados de asistencia a la conducción (ADAS),** estos sistemas están basados en sensores de entorno (Ultrasonidos, Radar) Cámaras de vídeo que detectan objetos y son capaces de detectar las inmediaciones del vehiculo, ayudando al conductor a hacer la conducción mas cómoda y segura.

Entre los sistemas ADAS existentes encontramos las siguientes:

* **Sistema de navegación a bordo (In-vehicle navigation system)**. Con GPS y TMC para ofrecer posicionamiento en carretera e información del tráfico
* **Control de crucero adaptativo (Adaptative cruise control).** Adapta automáticamente la velocidad del vehículo para mantener una distancia seguridad con los vehículos que le anteceden.

|  |  |
| --- | --- |
| Ilustración 2: Control de crucero adaptativo | Ilustración 3: Navegador de a bordo |

* **Asistencia de cambio de carril (Lane change assistance).** Son sistemas que cubren el ángulo muerto de los vehículos avisando de la presencia de otros en sus inmediaciones.
* **Sistema de advertencia de abandono de carril (Lane departure warning system).** dvierte al conductor cuando el vehículo comienza a salirse de su carril en autopistas y carreteras principales.

|  |  |
| --- | --- |
| Ilustración 4: Asistencia de cambio de carril | Ilustración 5: Advertencia de abandono de carril |

* **Sistema anticolisión (Collision avoidance system).** Detecta obstáculos u otros vehículos

detenidos en la vía y actúa sobre el freno del vehículo en caso de prever colisión.

* **Adaptación de velocidad inteligente (Intelligent speed adaptation).** Este sistema

monitoriza la velocidad límite de la carretera y advierte al conductor o actúa sobre el

vehículo en caso de que este sobrepase dicho límite.

|  |  |
| --- | --- |
| Ilustración 6: Sistema anti-colisión | Ilustración 7: Adaptación de velocidad inteligente |

* **Reconocimiento de señales de tráfico (Traffic sign recognition).** Capta las señales de tráfico y advierte al conductor sobre su presencia en la carretera.
* **Ayuda en aparcamiento (Park assist).** Es un sistema que cuando se activa indica al conductor mediante alertas la cercanía a los obstáculos en la maniobra de aparcamiento.

|  |  |
| --- | --- |
| Ilustración 8: Reconocimiento de señales | Ilustración 9: Ayuda en aparcamiento |

* **Control de luces adaptativo (Adaptive light control).** Actúa sobre las luces de del vehículo en respuesta a las condiciones de visibilidad, dirección, suspensión, velocidad del vehículo, o presencia de otros vehículos.
* **Visión nocturna (Automotive night visión).** Dispone de una pantalla donde se monitoriza la carretera en visión nocturna. Sirve para mejorar la percepción del conductor en la oscuridad.

|  |  |
| --- | --- |
| Ilustración 10: Control de luces adaptativo | Ilustración 11: Visión nocturna |

* **Control de descenso (Hill descend control).** Permite un descenso de pendientes suave y controlado en terrenos irregulares sin que el conductor tenga que pisar el freno.
* **Detección de somnolencia en el conductor (Driver drowsiness detection).** Dispone de una cámara que monitoriza la cara del conductor y detecta cuando éste está somnoliento.
* **Sistemas de comunicación Vehicular (Vehicular communication systems).** Son nodos que se sitúan en las carreteras y que se comunican con los vehículos transmitiéndoles información sobre el tráfico o advertencias de seguridad.

En conclusión vemos como cada vez se van integrado mas automatismos o sistemas ADAS que van tomando mas protagonismo en la conducción en busca de la meta de la conducción segura y autónoma que ira tomando relevancia según vayamos aceptando estas nuevas mejoras.

Se puede observar que dentro del ámbito de la conducción autónoma no existe ningún estándar puesto que aun esta en vías de desarrollo una solución única y cada modelo esta basado en sus propios sistemas, no cabe duda que este es un gran proyecto que tarde o temprano terminara por salir a luz y estandarizarse puesto que los controles de seguridad necesarios para conseguir que un coche conduzca sin intervención humana son gigantescos.

1. Drones

Vehículo aéreo no tripulado (VANT) o Dron es una aeronave autónoma o controlada a distancia, Inicialmente usados con fines militares se popularizaron en la población civil.

Históricamente los VANT eran siempre vehículos pilotados remotamente , pero se ve una cierta evolución a tener un control sobre el vehículo desde una ubicación remota o dándoles instrucciones de vuelo pre programadas dándoles un complejo sistema de automatización dinámica.

Se pudo observar un punto de inflexión por el cual se popularizaron entre la población civil con la aparición de los primeros drones de Parrot AR.Drone, que aunque no fue ni mucho menos el primero, si que lo popularizo este tipo de vehículos.

Otros Coches RC

1. Analisis de la situación actual

Existen múltiples productos de estas características desde versiones comerciales a proyectos independientes y modificaciones sobre productos comerciales.

Aunque a nivel precio es imposible competir con una versión comercial de un coche teledirigido, este nos plantea varios problemas.

1. No es posible modificar su funcionamiento
2. No suelen trabajar sobre Open Hardware y por lo tanto es complicado hacer modificaciones.
3. Estamos restringidos a los programas de control y de comunicaciones que nos proporcione el fabricante, no podemos ampliar nuestro dispositivo en la forma que deseemos o cambiar elementos a nuestro antojo.
4. No es posible modificar el software proporcionado puesto que no tenemos el código fuente de este y por lo tanto no podemos variar el comportamiento.

En definitiva en versiones comerciales estamos restringidos a lo que el fabricante quiera aportar.

Otra alternativa es recurrir a proyectos independientes, aunque es una alternativa valida estamos en una situación similar a la actual vamos a tener de desarrollar nuestro propio programa de o se nos proporcionaran API de desarrollo con las cuales podremos avanzar en la creación de nuestro prototipo, en la mayoría de las ocasiones solo proporcionan pequeños ejemplo o programas a medias para controlar los dispositivos.

En este caso nos encontramos con la siguientes dificultades.

1. Por regla general estos proyectos están desarrollados con componentes hardware específicos que no tienen por que encajar dentro dentro de nuestro proyecto.
2. Tenemos que trabajar lo mismo en implementar una versión independiente que en hacer nuestro propio vehículo y software de control.
3. En la mayoría de los casos implementar modificaciones es muy costoso puesto que o el hardware que se distribuye no es compatible con modificaciones o es necesario investigar para implementar nuestras modificaciones.

Dadas todas estas características Nos hemos decantado por crear nuestro propio coche

1. Percepción del entorno

Uno de los fundamentos básicos de la conducción es la percepción del entorno, este hecho no es diferente en un coche autónomo puesto que es necesario que este de forma independiente sea capaz de reconocer el terreno y los posibles obstáculos que le rodean.

Esta percepción se puede realizar con muchas y variadas técnicas que pasamos a relatar un poco mas a fondo a continuación

1. Visión artificial

Sin duda esta es una de las técnicas mas importantes dentro de la automatización de un vehículo, podriamos definirla como la técnica dentro de la inteligencia artificial capaz de “entender” o “detectar” los objetos en una imagen.

Como características básicas dentro de la visión artificial podriamos destacar la siguientes:

* Detección y localización de objetos (caras , coches..).
* Registro y evaluación de resultados
* Seguimiento de objetos entre distintas imagenes.
* Mapeo de un escenario para conseguir un modelo tridimensional , permitirá moverse por el escenario capturado.

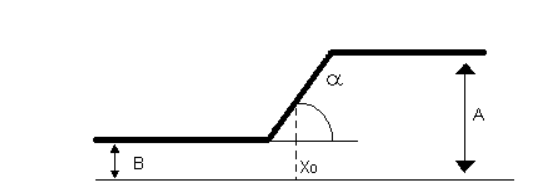
Todos estos objetivos se consiguen por distintas técnicas unas mas complejas que otras , como reconocimiento de patrones, geometría y proyección, aprendizaje estático, teoría de grafos...

Uno de los aspectos mas importantes de la visión artificial es sin duda el reconocimiento de objetos , patrones o identificación de figuras y formas, este reconocimiento puede ir desde ejemplos simples (reconocer los elementos rojos de un fotografía) hasta la detección y reconocimiento de personas.

Detección de objetos

En este campo tenemos muchas alternativas de estudio como la detección de objetos por patrones, colores o formas pero dato que en el mundo real la variación es demasiado grande tenemos que recurrir a cálculos mas avanzados como la detección de bordes.

Para obtener la forma de un imagen es necesario obtener los bordes de esta, se podria definir como la transición entre dos regiones significativamente distintas, esto nos proporciona una información muy importante sobre los objetos que existen en la imagen.

Ilustración 12: Modelo de borde ideal

En la imagen anterior se muestra un modelo unidimensional y continuo de un borde, este modelo representa la transición entre una región Rango B y otra completamente diferente Rango A.

El resultado de aplicar una detección de bordes a una imagen puede reducir significativamente la cantidad de datos a ser procesados y así descartar información no relevante para los cálculos, existen múltiples algoritmos para conseguir este objetivo, aunque encontramos implementaciones creadas sobre estas funciones, explicamos brevemente algunas de ellas:

* **Detector de bordes Canny.** Esta función utilizada en gran cantidad de software de procesamiento gráfico debe su nombre a su autor (John F. Canny), el algoritmos consiste en seleccionar los píxeles candidatos a pertenecer a un contorno mediante dos umbrales, uno alto y uno bajo, si un pixel tiene un gradiente superior a el umbral alto es considerado como borde si por el contrario es meno que el umbral bajo es descartado, si el pixel esta entre los dos umbrales solo se acepta si esta en contacto con un pixel de umbral alto.
* **Transformadas de Hough.** Con este calculo es posible detectar todo tipo de figuras que sea posible expresarlas mediante un formula matemática, deben su nombre al su inventor (Paul Hough), para este calculo se usa cuya dimensión es igual al numero de parámetros desconocidos del problema.

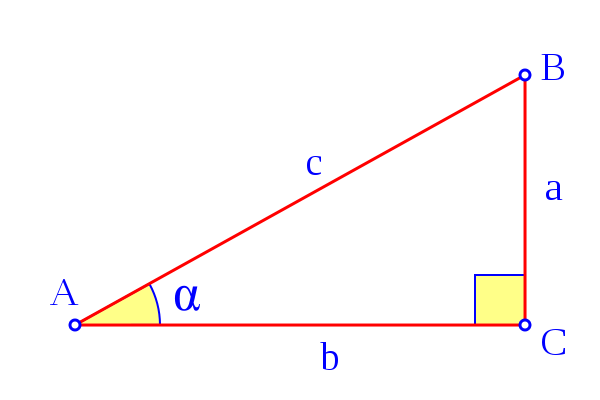
Calculo de distancias

El calculo de distancias mediante una cámara es relativamente sencillo puesto que disponemos de dos de los datos mas relevantes:

* Altura a la que tenemos la cámara respecto al suelo
* Ángulo de inclinación de la cámara.

Con estos datos podemos obtener otros datos interesantes y útiles mediante el uso de sencillas formulas de trigonometría.

* **Calculo de la hipotenusa:**

Ilustración 13: Esquema de trigonometría

* **Calculo de la altura:**
* **Calculo de la distancia:**

En este punto se nos plantea una incógnita como trasladamos estos datos a una medida física, es decir como traducimos la distancia en píxeles a centímetros, este calculo es también sencillo, puesto que conocemos la altura tanto en el medio real como en el medio físico podemos hacer una regla de tres por la cual obtendremos la distancia aproximada con el objeto.

El problema principal de este método es que necesitamos puntos de referencia con los cuales calcular las distancias,

1. Visión Artificial Estéreo

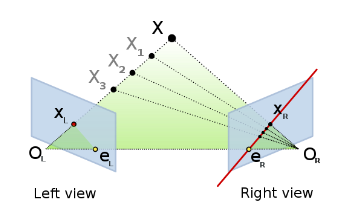
La visión estero es el modelo mas usado en la visón artificial para la proyección 3d, y calcular distancias y posiciones en el mundo real.

Una definición mas exacta es la visión binocular donde gracias dos imágenes y mediante una serie de cálculos conseguimos una única imagen.

Cuando trabajamos con cámaras estéreo es necesario tener en cuenta y conocer una serie de datos de importantes:

* Debemos conocer la distancia entre las dos cámaras así como su ángulo de rotación.
* Debemos conocer la distorsión racial de las lentes de las cámaras.
* Debemos conocer las distancias focales de las cámaras.

Estos datos podemos obtenerlos mediante una serie de cálculos automatizados , usando los algoritmos de Zhang[[1]](#footnote-2) , los cuales mediante un proceso de hacer varias fotografías a un cuadro de ajedrez en distintas posiciones espaciales nos permite obtener estos datos.

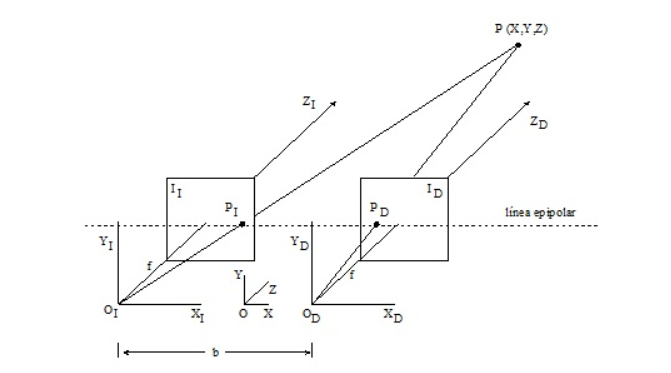
Ilustración 14: Visión Estéreo

* + - 1. Geometría de Estereoscópica

Como hemos definido anteriormente para poder hacer uso de la visión estereoscópica necesitamos dos cámaras cuyos ejes ópticos sean paralelos y separadas espacialmente en uno de los ejes, esta separación la definiremos como linea base y es una de las variables que se usaran para el calculo de la profundidad.

Tenemos que tener en cuenta que los ejes ópticos de las dos cámaras solo debe estar separados en una dimensión y nos basaremos en este dato para realizar el resto de los cálculos.

Según el siguiente gráfico

Ilustración 15: Gráfico geometría epipolar

podemos definir que O es el origen de nuestras imágenes, f es la distancia focal y b es la linea base. En cada una de las cámaras estableceremos un punto de coordenadas 3D (X,Y,Z) en el mundo real que se ven representadas por los puntos Pi(X1,Y1,Z1) y Pd(X2,Y2,Z2) que son las proyecciones del puntos real en cada una de las imágenes.

Consideramos también las lineas Poi y Pod para la linea trazada desde el centro focal de las cámaras izquierda y derecha respectivamente, estas dos lineas definen el plano epipolar.

Con este sistema definimos que la diferencia obtenida de restar los dos puntos según el eje de separación de las cámaras X1-X2 es la disparidad d entre las dos imágenes.

Utilizando las siguientes ecuaciones.

Según esta formula maestra podemos calcular la distancia a un punto en el espacio desde dos imágenes tomadas en paralelo.

1. Detección y reconocimiento de objetos con visión computerizada

El campo de la detección de la visón computerizada, uno de los procesos que mas interés puede despertar es la detección e objetos.

Existen varias formas de afrontar este problema cada una de ellas afronta el problema de una forma distinta y tiene una serie de beneficios e inconvenientes propios al algoritmo utilizado.

Pasamos a relatar alguna de los algoritmos que existen y explicaremos sus características.

como ya hemos detallado antes la forma básica de detección de objetos es aplicando algoritmos de detección de bordes, pero esto nos plantea un problema ¿que es lo que hemos detectado?, esta claro que tenemos que implementar un sistema que nos permita reconocer y categorizar los objetos.

El ojo humano es capaz de diferenciar un objeto de otro a través de la forma o el color del objeto, En el campo de la visión computerizada esto no es distinto.

* + - 1. Reconocimiento de objetos por color

Como hemos dicho antes el reconocimiento de imágenes por color es una de las formas básicas de diferenciar un objeto, en este momento se nos plantea un problema como detectamos un color en una imagen.

La forma mas sencilla de reconocer un color determinado en un imagen seria buscando pixel a pixel dicho color, pero esto además de no ser eficaz puesto que analizar un imagen pixel a pixel seria demasiado costoso, el color en una imagen digital puede variar de sustancialmente impidiendo que detectemos el color o directamente ofrecernos falso positivos.

Otra forma seria comparar el canal de color RGB deseado y intentar hacer una aproximación, comprobando que el resto de canales, el resto de canales no debería ser muy alto o por lo menos lo suficientemente bajo como para que no afecte a nuestro color, esta técnica nos plantea el problema de que tendríamos que usar colores básicos, limitando de esta forma nuestras posibilidades.

|  |  |
| --- | --- |
| Ilustración 16: Detección de colores por canal RGB | Ilustración 17: Resultado detección canales |

Otra forma mas óptima seria analizar la imagen y aplicando diversos filtros llegar al punto de poder diferencia sin lugar a dudas un color de otro, esto se puede realizar de varias funciones sobre el modelo de color HSV[[2]](#footnote-3), la mayoría de las librerías de visión artificial nos proporciona la capacidad de manejar imágenes en este modelo de color sobre el cual podremos hacer acciones que nos permitirán aplicando valores de (Hue , Saturation , Value ) extraer un color del resto.

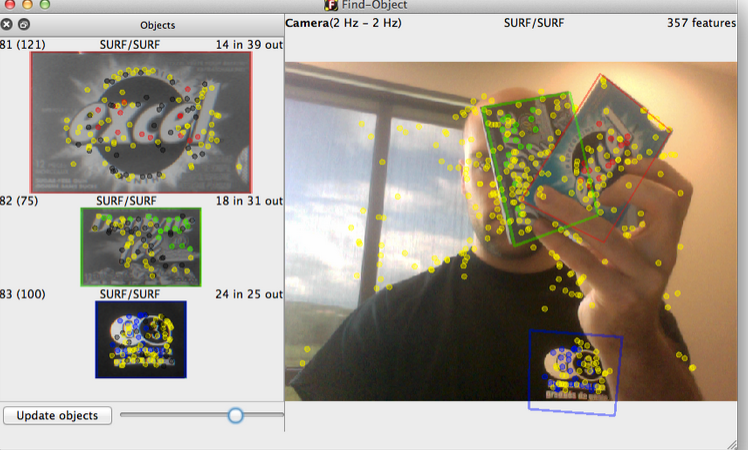
|  |
| --- |
| Ilustración 18: Filtro HSV sobre imagen |

Llegados a este punto podríamos identificar fácilmente que en la imagen existe un objeto y que es de un determinado color y actuar en consecuencia.

* + - 1. Reconocimiento de objetos por forma

Aunque en el punto anterior ya se ha conseguido detectar un objeto por color esto se nos plantea insuficiente, necesitamos evolucionar un paso mas, algo que nos permita identificar un objeto de forma un equivoca, la forma de hacer esto es detectar los objetos por su forma o contorno, este problema se podrá abordar de diversas formas y para ello nos apoyaremos en librerías de visión computerizada que nos dan las herramientas necesarias para este fin.

Una de las formas de detección de objetos por forma es hacer uso de SU[[3]](#footnote-4)RF, es un algoritmo de visión por computador, capaz de obtener un representación visual de un imagen y obtener información detallada de esta. Esto nos permitirá reconocer objetos que anteriormente hallamos visto y seleccionado, entraría dentro del campo de la inteligencia artificial.

Ilustración 19: Detección de objetos por SURF

El problema principal de esta técnica es la cantidad de procesamiento necesario para detectar y analizar todos los puntos de interés de un imagen y detectar si un objeto se encuentra en dicha imagen.

Otro procedimiento aunque no tan dinámico como SURF es el método denominado como HAAR-like[[4]](#footnote-5), este método nos permite localizar objetos dentro de una imagen por medio de descriptores precargados en el sistema.

La localización de objetos por Haar es extremadamente rápida puesto que busca objetos que concuerden con las característica del fichero que hemos precargado y que anteriormente hemos precalculado.

El problema de este método es que es necesario procesar miles de imágenes y entrenar al sistema para que sea capaz de reconocer un objeto, Existen programas preparados para realizar estos cálculos de forma autónoma sin necesidad de intervención humana, el punto positivo de esta técnica es que nos permite identificar de forma rápida y eficaz los objetos que deseemos descartando el resto de la imagen.

|  |
| --- |
| Ilustración 20: Detección objetos con Haar |

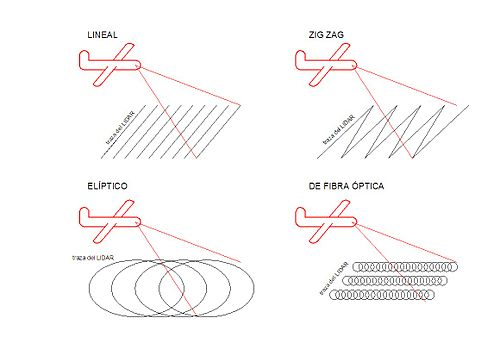
1. LIDAR

Es el acronimo de (Light Detection and Ranging) es una tecnología que permite medir la distancia desde un emisor a un objeto gracias a un emisor laser pulsado, la distancia al objeto es determinada por el retraso entre la emisión y la recepción del pulso reflejado.

Aunque es mas usado en el campo de la aeronáutica existen distintos proyectos de coches autónomos e implementan este sistema para reconocimiento del terreno y la distancia entre los objetos, también conocida por ser la tecnología usada en los controles de velocidad.

Existen distintas técnicas de uso de este sistema que nos permite hacer un reconocimiento del terreno cada una con sus beneficios y problemas.

* Lineal: El haz láser es desviado por un espejo en un sentido, da unas medidas uniformes, pero existen puntos ciego al tener un barrido en una única dirección.

Ilustración 21: Tipos de LIDAR

* ZigZag: Es una evolución del anterior en este caso el movimiento del espejo es en dos sentidos , pero existen descompensaciones de la cantidad de puntos obtenidos, en los extremos, teniendo menos puntos ciegos pero con zonas con menor resolución.
* Fibra óptica: En es caso el sistema monta múltiples espejos mas pequeños con múltiples haces lo que nos da mayor resolución pero un ángulo menor de escaneado .
* Eliptico en este caso el haz es desviado por dos espejos produciendo un escaneado eliptico que nos da lecturas del mismo punto desde distintos ángulos, lo que nos beneficia en algunos aspectos pero complica el procesamiento al tener distintas lecturas.

1. GPS

Esta tecnología hoy por hoy no es desconocida, este sistema de localización permite ubicar con una precisión de centímetros, según la técnica usada, la ubicación de un objeto a nivel mundial.

Esta localización se realiza gracias a un red de satélites en orbita que ubican el objeto gracias a la trilateracion, dándonos la ubicación en el globo con una precisión de metros.

|  |  |
| --- | --- |
| Ilustración 22: Cálculo de la posición por triangulación | Ilustración 23: Red Navstar de satelites GPS |

Su funcionamiento es relativamente sencillo pues se basa en le calculo del tiempo transcurrido entre la recepción de las cuatro señales de los satélites y la posición de estos, el dispositivo gps utilizara estos datos para triangular su ubicación respecto a los satélites.

El mensaje que nos mandan (efemeride) dando la posición precisa de los mismos. Esta información se cambia frecuentemente, siendo actualizada por las estaciones de seguimiento de la Tierra. Los parámetros orbitales de los satélites se van actualizando a medida que su movimiento se ve alterado por la atracción del Sol y la Luna, la diferencia de gravedad entre distintas zonas de la corteza terrestre, viento solar, etc. Un período de cambio típico sería de 4 horas. .

1. RADAR

acrónimo de Radio detection and ranging, es un sistema que usa hondas electromagnéticas para medir distancias, altitudes y velocidades a objetos estáticos o en movimiento.

Su funcionamiento no difiere mucho del resto de tecnologías, siendo en realidad la precursora de ellas, básicamente calcula la distancia midiendo el tiempo entre la emisión y la recepción del eco en la misma posición del emisor, de este eco se puede obtener gran cantidad de información, el uso de hondas electromagnéticas de distintas frecuencias nos puede aportar distintos tipos de información.

1. Radio Control

Básicamente podríamos definir como el sistema por el cual podemos controlar remotamente un objeto de forma inalámbrica, esta comunicación puede ser por cualquier medio de comunicación existente, comúnmente se usan emisoras RF de control remoto.

En el campo del radio control entra en escena tres actores fundamentales electrónica , electricidad y mecánica, que son los actores fundamentales de toda automatización que queramos lograr.

Actualmente existen multitud de vehículos comerciales de todos tipo , coches , aviones, barcos helicópteros.... radio controlados.

1. Implementación
2. Elementos necesarios
3. Para el desarrollo de este proyecto vamos a necesitar varios elementos hardware con los cuales montaremos el dispositivo que queremos controlar remotamente.
4. Estudiaremos las diversas alternativas que existen actualmente en el mercado y seleccionaremos aquellas que nos ofrezcan mejores resultados.
5. Inicialmente y como hardware de partida tendremos el siguiente:
6. RaspberryPi (Modelo B, B+)
7. Módulo Bluetooth USB (V2, V4)
8. Módulo Wifi USB ( B/G )
9. Driver Motores En este caso estudiaremos diversos tipos de control I2C Adafruit PCA9685, I2C, PiBorg Reverse PIC16F1824, Puente H L298N , I2C Port Expander MCP23017
10. Servo Motores Rotación Continua ( En este caso hemos seleccionado los siguientes tipos xce motores SM-S4303R, FS90R)
11. Motores DC En este caso no seleccionamos ningún modelo especifico pero indicamos la necesidad de un sistema de moto reducción para limitar la RPM y aumentar el PAR motor,
12. Sensores de proximidad Modelo HC-SR04
13. Cámara USB No seleccionamos modelo especifico, pero debe ser capaz de comprimir por hardware el fuljo de video.
    1. Descripción Software Utilizado
14. IDES Desarrollo

Detallar eclipse netbeans

1. Sistema Operativo

Detallar linux y raspbian

1. Librerías de programación

Dentro de este proyecto hemos tenido que usar varias librerías java las cuales por defecto no existen para la arquitectura en la que estamos desarrollando “arm[[5]](#footnote-6)”, por lo tanto nos vemos en la obligación de recompilar dichas librerías y generar un interfaz JNI[[6]](#footnote-7) con el que poder programa en java.

Aunque la mayoría de estas librerías tienen la posibilidad de generarse por medio de CrossCompiling[[7]](#footnote-8) se ha podido comprobar que los resultados no son todo lo óptimos que cabe esperar, comparado con la compilación directa de la librería sobre la raspberryPi, por lo tanto aunque es bastante mas costoso en tiempo se decide compilar sobre la plataforma destino.

* + - 1. OpenCV

Ya hemos comentado en este proyecto las características principales de esta librería de visión computerizada, en este punto vamos a destacar como hemos generado la librería y los pasos necesarios para ello.

El resultado que nos interesa de esta generación son dos ficheros específicos “libopencv\_java300.so” y “opencv-300.jar” que son respectivamente la librería de desarrollo de opencv y la interfaz java para nuestro proyecto.

Actualizamos nuestro entorno de sistema

sudo add-apt-repository ppa:webupd8team/java

sudo apt-get update

sudo apt-get upgrade

sudo rpi-update

Instalamos las dependencias necesarias para compilar

sudo apt-get install build-essential cmake pkg-config ant oracle-java8-installer oracle-jdk8 installer oracle-java8-set-default

Instalamos librerías necesarias de desarrollo

sudo apt-get install libjpeg8-dev libtiff4-dev libjasper-dev libpng12-dev sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev

Configuramos el entorno de desarrollo

wget -O opencv-3.0.0.zip https://github.com/Itseez/opencv/archive/3.0.0-rc1.zip

unzip opencv-3.0.0.zip

cd opencv-2.4.10

mkdir build

cd build

1. Generamos los ficheros de compilación

export JAVA\_HOME=/usr/lib/jvm/jdk-8-oracle-arm-vfp-hflt

cmake -DBUILD\_SHARED\_LIBS=OFF ..

1. Compilamos e instalamos

make -j8

sudo make install

* + - 1. JDK DIO

Esta se trata de un extensión de programación orientada inicialmente a Java Me que nos ofrece la capacidad de controlar los puertos GPIO de nuestro dispositivo, como punto fuerte tiene la capacidad de controlar distintos tipo de dispositivos no solo RaspberryPi como es el caso de Pi4J , además se trata de una librería realmente rápida ala estar orientada a desarrollar para dispositivos móviles.

Como punto negativo cabe indicar que se trata de una librería de bajo nivel y que muchas de las mejoras que se han introducido en WiringPi o Pi4J que hacen la vida del desarrollador mas sencilla, aquí se tiene que controlar a mano, como por ejemplo en control de puertos PWM por software.

Los pasos para generar son los siguiente, como en el anterior caso compilaremos dentro del sistema RaspberryPi de esta forma evitaremos hacer CrossCompiling

Actualizamos nuestro entorno de sistema

sudo add-apt-repository ppa:webupd8team/java

sudo apt-get update

sudo apt-get upgrade

sudo rpi-update

Instalamos las dependencias necesarias para compilar

sudo apt-get install build-essential mercurial

Configuramos el entorno de desarrollo

mkdir java-dio  
cd java-dio/

export PI\_TOOLS=/usr  
export JAVA\_HOME=/usr/lib/jvm/jdk-8-oracle-arm-vfp-hflt

Descargamos el código fuente del repositorio

hg clone http://hg.openjdk.java.net/dio/dev

export DIO\_DEV=/java-dio/dev

Compilamos

make

Copiamos resultados a las directorios de sistema

cp -r $DIO\_DEV/build/deviceio/lib/\* $JAVA\_HOME/jre/lib

Para ejecutar un programa con esta librería deberemos acordarnos de ejecutar con los siguientes parámetros donde indicaremos las características y permisos de nuestro dispositivo

sudo $JAVA\_HOME/bin/java -Djdk.dio.registry= ./config/dio.properties-raspberrypi -Djava.security.policy=./gpio.policy -jar Programa.jar

* + - 1. Blue-Cove

Esta es una librería de desarrollo que nos permitirá controlar desde nuestro programa Java dispositivos Bluetooth asi como abrir sockect Bluettoth para transmitir datos, ordenes o incluso ficheros.

Como en los casos anteriores no disponemos de una librería compilada para sistemas ARM y por lo tanto tendremos que generarla para este sistema.

Al contrario de las dos anteriores en esta librería no se permite CrossCompiling, de esta forma no queda mas alternativa que compilarla dentro del entorno RaspberryPi.

El caso de esta librería es un poco mas especifico puesto que tendremos que generar el interfaz JNI desde una librería ya generada.

Los pasos para generar esta librería son los siguientes.

Descargamos las herramientas necesarias en el sistema

sudo apt-get install bluetooth bluez-utils blueman

Preparamos el entorno de trabajo

sudo mkdir -p -v /src/main/java

sudo mkdir -p -v /src/main/c

sudo mkdir -p -v /target/classes

sudo mkdir -p -v /target/native

sudo mkdir -p -v /jars/com/intel/bluetooth

Descargamos Las librerías sobre las que vamos a trabajar

wget http://code.google.com/p/bluecove/downloads/detail?name=bluecove-gpl-2.1.0-sources.tar.gz&can=2&q=

wget http://code.google.com/p/bluecove/downloads/detail?name=bluecove-gpl-2.1.0.jar&can=2&q=

Descomprimimos el código fuente generado

tar -zxvf bluecove-gpl-2.1.0-sources.tar.gz

Compilamos el código fuente

javac -d /target/classes -g -Xlint:unchecked -source 1.3 -target 1.1 -cp /jars/bluecove-2.1.0.jar /src/main/java/com/intel/bluetooth/BluetoothStackBlueZ\*.java

cp /target/classes/com/intel/bluetooth/BluetoothStackBlueZ\*.class /jars/com/intel/bluetooth

Generamos los ficheros de inclusión de la interfaz JNI

javah -d /src/main/c com.intel.bluetooth.BluetoothStackBlueZ  
com.intel.bluetooth.BluetoothStackBlueZConsts com.intel.bluetooth.BluetoothStackBlueZNativeTests

Compilamos en C el interfaz JNI

cd /target/native

gcc -fPIC -c /src/main/c/\*.c -I/opt/java/jdk1.8.0/include -I/opt/java/jdk1.8.0/include/linux

gcc -shared -lbluetooth -WI,-soname,libbluecove-2.1.0 -o /target/libbluecove.so /target/native/\*.o

Copiamos los resultados

cp /target/libbluecove.so /jars/libbluecove\_arm.so

1. Preparación del entorno de trabajo
   * + 1. Clasificadores Haar

Puesto que una gran parte del proyecto esta basada en la conducción autónoma del vehículo, es decir que el coche sea capaz de reconocer señales u obstáculos y de variar su ruta con respecto a estas hemos creado una serie de obstáculos predefinidos que nos permitirán realizar esta acción.

Para realizar el training seria necesario hacer cientos de imágenes en las que mostráramos el objeto que queremos buscar sobre un fondo y necesitaríamos a su vez cientos de fondos, generalmente se suele necesitar una proporción de 2 imágenes positivas por cada negativa.

Como es imposible tomar la cantidad de imágenes que necesitamos, recurrimos a una serie de programas compilados en un punto anterior, compilación de openCV.

El primer programa que necesitaremos sera opencv\_createsamples este programa nos permitirá crear las imágenes de muestra desde un repositorio de imágenes de fondo, por ejemplo **http://tutorial-haartraining.googlecode.com/svn/trunk/data/negatives/,** que podemos descargar de internet y nuestro objetos recortados individualmente, de esta forma con 10 simple imágenes de nuestros objetos recortados sobre fondo transparente podremos crear miles de imágenes de muestra.

Para esto seguiremos los siguientes paso:

Descargaremos las imágenes de fondo y crearemos una lista con los nombre de las imágenes

cd negativeImageDirectory  
wget -nd -r -A "neg0\*.jpg" http://tutorial-haartraining.googlecode.com/svn/trunk/data/negatives/

cd negativeImageDirectory  
ls -l1 \*.jpg > negatives.txt

Con nuestra imágenes de de muestra editadas, necesitaremos al menos 10 imágenes de muestra de cada objeto para poder generar un conjunto de samples suficientemente grande y variado.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Tabla 1: Imágenes de muestra

Ejecutaremos la siguiente instrucción

opencv\_createsamples -img <IMG\_MUESTRA> -bg negativeImageDirectory/negatives.txt -info sampleImageDirectory/<SALIDA\_IMG\_MUESTRA.TXT> -num <NUMERO\_SAMPLES\_GENERAR> -bgcolor <COLOR\_FONODO> -bgthresh 8 -w 48 -h 48

Esta instrucción nos generara el numero indicado de imágenes de nuestra imagen sobre un fondo aleatorio, este proceso lo tendremos que repetir con cada uno de las imágenes de nuestro objeto.

Cuando tengamos todas nuestras imágenes originales procesadas tendremos que generar un fichero de índice que contenga todas las salidas que hemos definido para ello ejecutaremos la siguiente instrucción.

cd sampleImageDirectory

cat \*.txt > positives.txt

Con este fichero generaremos el índice de imágenes positivas que usaremos posteriormente para conseguir un fichero cascadedescriptor, con el que identificaremos el objeto por medio de la camara.

opencv\_createsamples -info sampleImageDirectory/positives.txt -bg negativeImageDirectory/negatives.txt

-vec <FICHERO\_INDICE.vec> -num <NUM\_SAMPLES\_GENERADOS> -w 48 -h 48

En este punto tenemos que empezar con el entrenamiento, para conseguir el cascadeDescriptor, para ello ejecutaremos la siguiente instrucción.

opencv\_traincascade -data outputDirectory -vec <FICHERO\_INDICE.vec> -bg negativeImageDirectory/negatives.txt   
-numPos <NUM\_SAMPLES\_VALIDOS> -numNeg <NUM\_SAMPLES\_NO\_VALIDOS> -numStages <NUMERO\_PASOS> -precalcValBufSize 1024 -precalcIdxBufSize 1024 -featureType HAAR -minHitRate 0.995 -maxFalseAlarmRate 0.5 -w 48 -h 48

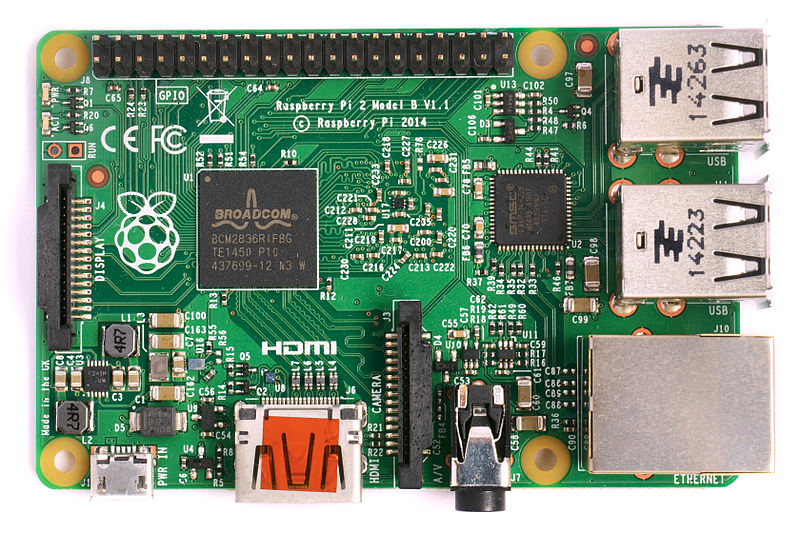
1. Descripción del hardware utilizado
2. Controlador del sistema

Como elemento principal de sistema y encargado de controlar todos los dispositivos y donde residira la lógica del proyecto usaremos dispositivos de tipo raspberryPI, en este caso nos hemos decidido por el modelo B/B+ puesto que nos ofrece una serie de características que consideramos necesarios.

Pasmos a describir rápidamente este tipo de dispositivo.

RaspberryPI B/B+

Es un ordenador de bajo coste basado en el procesador ARM11, aunque nació como un ordenador de bajo coste para facilitar la enseñanza de informática en colegios, se ha convertido en un precedente a nivel mundial y ha abierto un nuevo mundo de proyectos hasta hace poco inabordables por coste o dificultad.

Ilustración 24: RaspberryPi B+

Las características principales por las que elegimos este dispositivo son las siguientes:

4 puerto USB 2.0

Procesador de ARM11 1176JZF-S a 700 MHz overclokeable de forma sencilla hasta 1Ghz

512 Mb de memoria Ram

Conexión de red mediante RJ45

Lector de tarjetas SD/MSD

Alimentación de bajo consumo 5V/2A

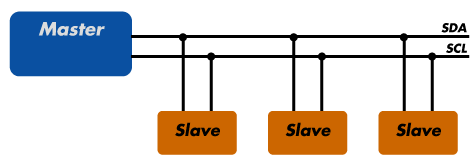
SO basado en linux

1. Controlador de Motores
   * + 1. I2C

I2c ( Inter-Integrated Circuit ) es un bus de comunicaciones serie usado principalmente para comunicación entre microcontroladores y periféricos .

La principal característica de I2C es que hace uso de dos lineas para transmitir la información, una de ellas transporta los datos (SDA) y la otra transporta la señal de reloj (SCL) . Es necesaria una tercera linea utilizada única y exclusivamente como masa (GND).

Los dispositivos conectados por I2C tiene una dirección única que los identifica inequívocamente el dispositivo en la serie de periféricos , estos se conectan en cascada permitiendo conectar varios dispositivos y que todos funcionen al mismo tiempo.

Ilustración 25: Esquema de conexión I2C

Habiendo varios dispositivos conectados sobre el bus, es lógico que para establecer una comunicación a través de él se deba respetar un protocolo. Digamos, en primer lugar, lo más importante: existen dispositivos **maestros** y dispositivos **esclavos**. Sólo los dispositivos maestros pueden iniciar una comunicación

La condición inicial, de **bus libre**, es cuando ambas señales están en estado lógico alto. En este estado cualquier dispositivo maestro puede ocuparlo, estableciendo la condición de **inicio** (start). Esta condición se presenta cuando un dispositivo maestro pone en estado bajo la línea de datos (SDA), pero dejando en alto la línea de reloj (SCL).

El primer byte que se transmite luego de la condición de inicio contiene siete bits que componen la dirección del dispositivo que se desea seleccionar, y un octavo bit que corresponde a la operación que se quiere realizar con él (lectura o escritura).

Si el dispositivo cuya dirección corresponde a la que se indica en los siete bits (A0-A6) está presente en el bus, éste contesta con un bit en bajo, ubicado inmediatamente luego del octavo bit que ha enviado el dispositivo maestro. Este bit de **reconocimiento** (ACK) en bajo le indica al dispositivo maestro que el esclavo **reconoce** la solicitud y está en condiciones de comunicarse. Aquí la comunicación se establece en firme y comienza el intercambio de información entre los dispositivos.

* + - 1. I2C Adafruit PCA9685

Este es uno de los controladores de motor que usaremos en nuestras pruebas , fabricado por adafruit ([https://www.adafruit.com](https://www.adafruit.com/)) se nos ofrece esta placa en un kit de sencillo ensamblado.

Inicialmente este dispositivo es usado para controlar servomotores o leds RGB.

Cada una de las 16 entradas independientes que tiene el dispositivo nos permite controla un motor distinto e independiente de lo demás.

Contradictoriamente solo podremos controlar un máximo de 4 led RGB dado que cada color del led sera controlado por un canal distinto.

|  |  |
| --- | --- |
| Ilustración 26: 16-channel, 12-bit PWM Fm+ I2C-bus LED controller | Ilustración 27: PCA9685 |

Esta placa nos ofrece las siguientes capacidades:

Controlador PWM sobre I2C , lo que nos permite no tener que estar controlando los pulsos ni mandar información constante a los dispositivos controlados.

Capacidad de una doble alimentación para alimentar el microcontrolador a 3.3V y un segunda entrada de potencia que permite alimentar los dispositivos controlados de forma independiente.

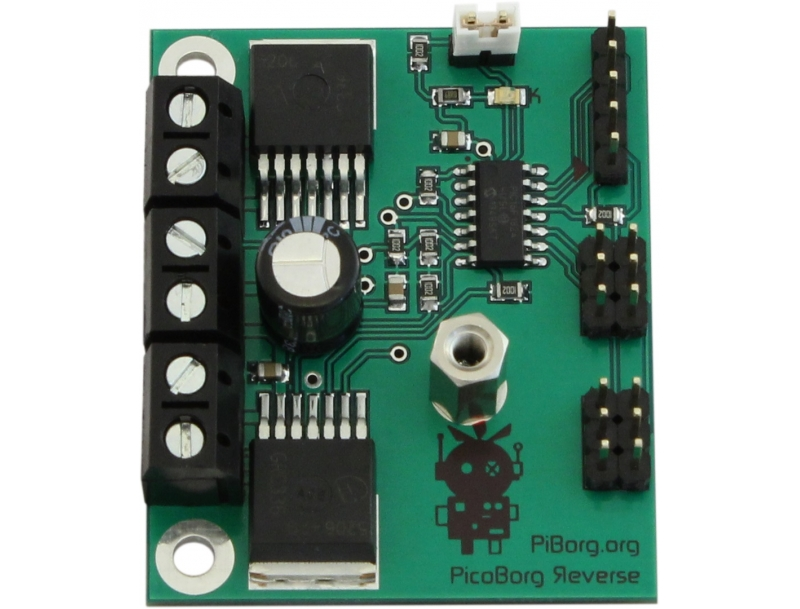
Permite configurar la dirección de controlador anteriormente indicada, ofreciendo la posibilidad de configura nuestro sistema en distintas direcciones.

Permite frecuencias de hasta 1,6Khz lo que nos ofrece una alta tasa de transferencia.

Permite el apagado directo de todos los dispositivos de forma automatica escribiendo en una única posición de memoria.

* + - 1. I2C PiBorg Reverse PIC16F1824

Este controladora de motores al no permite manejar 2 motores DC o un motor de pasos, al igual que la anterior funciona mediante el protocolo I2C.

Ilustración 28: I2C PiBorg Reverse

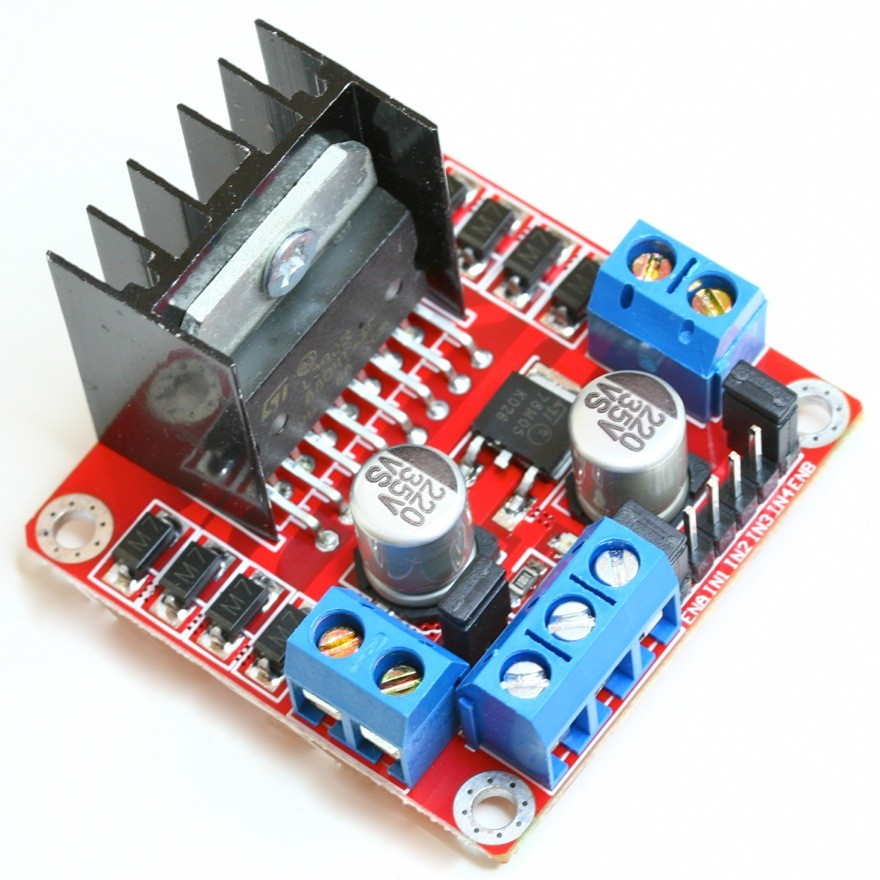
Aunque con menores capacidades en principio es un placa completamente valida puesto que no tenemos necesidad de mas de dos motores para controlar el coche.

Esta controladora nos ofrece la posibilidad de controlar automáticamente un motor de pasos , característica que puede ser de utilidad a la hora de controlar exactamente el giro del motor.

* + - 1. Puente H L298N

Este elemento al igual que el anterior nor permite controlar dos motores DC o un motor de pasos, pero a diferencia del anterior no es controlado por I2C.

Este elemento es un componente mucho mas económico que los dos anteriores, pero como inconveniente nos aporta la dificultad de control del interfaz, teniendo que controlar la potencia y el estado de cada uno de los pines constantemente.

Ilustración 29: H Bridge L298N

Por otro lado tiene la problemática del control de temperatura puesto que este tipo de elemento tiende a calentarse en exceso impidiendo un uso prolongado del interfaz.

* + - 1. I2C Port Expander MCP23017

Como ultimo elemento dentro de los componentes I2C testearemos las capacidades de este micro controlador que nos ofrece una capacidad de expansión de puertos.

Con este controlador por bus I2C tenemos la posibilidad de ampliar el numero de puertos I/O a nuestro dispositivo RaspberryPI.

|  |  |
| --- | --- |
| Ilustración 30: Controlador MCP23017 | Ilustración 31: Esquema MCP23017 |

Este microcontrolador nos ofrece las siguientes características:

Ampliación de 16 puertos I/O

Nos permite aislar los puertos GPIO de la raspberryPi

Controlado por medio del protocolo I2C

Fácil configuración mediante pins de la dirección de memoria utilizada.

Parada de emergencia de todas las salidas mediante la escritura en una única dirección de memoria.

Permite la conexión en paralelo de hasta 9 dispositivos en cascada permitiendo aumentar el numero de puertos disponibles de forma sencilla.

1. Implantación

En este punto vamos a definir que elementos vamos a usar y como vamos a implementar la solución a nuestro proyecto.

1. Visión artificial y detección de objetos

Unos de los puntos mas importantes del proyecto es este pues vamos a crear un sistema que sea capaz de conducir de forma autónoma esquivando obstáculos que pongamos en su camino.

Partiremos de la implementación de la librería de procesamiento de imágenes OpenCV, esta librería es usada en multitud de proyectos, incluidos algunos proyectos de conducción autónoma.

Actualmente esta librería nos da la capacidad de obtener imágenes de una cámara procesarlas y obtener información de ellas.

1. Detección de obstáculos.

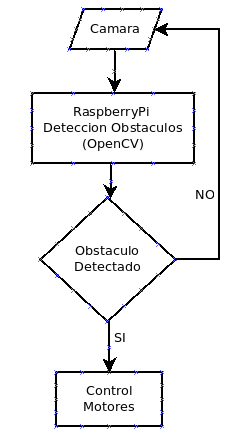
Aunque existen múltiples formas de abordar este tema nos centraremos inicialmente un hecho simple, vamos a partir de que todos los obstáculos que nos encontraremos tiene algo en común, aunque no es un dato muy realista nos servirá para avanzar en la implantación del proyecto y continuar desde este punto hacia objetivos mas complejos.

Para ello definiremos que todos los obstáculos que vamos a encontrar tendrán un circulo redondo en el centro del obstáculo que podamos identificar mediante visión artificial.

Esto nos permitirá de forma sencilla detectar si tenemos algún obstáculo en nuestro campo de visión, el siguiente punto que abordaremos mas adelante sera el calculo de la distancia hasta dicho obstáculo.

1. Diagrama de procesamiento

Como hemos definido anteriormente vamos a actuar sobre obstáculos con una forma determinada, aun así de forma genérica podemos construir un procedimiento que iremos evolucionando a lo largo del proceso.

Ilustración 32: Etapa inicial proceso

Inicialmente partiremos del siguiente diagrama :

Como Inicio del proceso nuestro sistema buscara obstáculos definidos por una imagen con un circulo negro, para ello usaremos las librerías OpenCV y mas concretamente la función **HoughCircle.**

La función HoughCircles es una evolución de la función HouhLines la cual realiza varios cálculos para detectar lineas rectas dentro de una imagen, en nuestro la función nos retornara un vector con los círculos detectados indicando su centro y su radio.

Podemos observar por los siguientes ejemplos como funcionara el sistema.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Tabla 2: Detección Círculos con OpenCV

En este ejemplo vemos como por medio de la cámara capturamos una imagen , buscamos posibles círculos que se encuentren en la imagen y los marcamos dibujando el circulo sobre esta.

Hemos coloreado los las zonas donde no se encuentran los círculos para destacar las posibles rutas que podremos seguir.

|  |
| --- |
| public Mat detectarCirculos()  {  capturarImagen();  Mat circulosDetectados = new Mat();    Imgproc.erode(m\_ultimaImagenGris, m\_ultimaImagenGris,new Mat());  Imgproc.dilate(m\_ultimaImagenGris, m\_ultimaImagenGris,new Mat());  Imgproc.Canny(m\_ultimaImagenGris, m\_ultimaImagenGris, 5, 70);  Imgproc.GaussianBlur( m\_ultimaImagenGris, m\_ultimaImagenGris, new Size(9, 9), 2, 2 );    Imgproc.HoughCircles(m\_ultimaImagenGris, circulosDetectados, Imgproc.CV\_HOUGH\_GRADIENT, 1, 1 ,200,100,0,0);  System.out.println(circles);  for (int i = 0; i < circulosDetectados.cols(); i++)  {  double[] circuloBucle = circulosDetectados.get(0, i);  Point pCentroBuble = new Point(circuloBucle[0], circuloBucle[1]);  Imgproc.circle(m\_ultimaImagen, pCentroBuble, (int) circuloBucle[2]- 10 , new Scalar(0, 255, 0),10);  }  return m\_ultimaImagen;  } |

Realmente no es necesario realizar ninguna modificación en la imagen que seria muy costosa en tiempo de procesamiento, simplemente nos bastaría con la información contenida en la variable circulosDetectados para continuar con el procesamiento.

Como se puede observar se realiza un procesamiento previo de la imagen, este procesamiento no es necesario pero para evitar posibles fallos en la detección es mejor hacerlo.

1. Detección de la distancia al objeto

Para el calculo de las distancias haremos uso de de uno o varios sensores que sean capaz de calcular las distancias hasta los obstáculos detectados.

Para ellos haremos uso de la librería Pi4J y de un componente hardware HC-SR04, detallado anteriormente en la sección de hardware.

Una vez que hemos detectado que tenemos un obstáculo delante nuestro ejecutaremos la rutina para que nos diga la distancia hasta dicho obstáculo.

Ilustración 33: HC-SR04

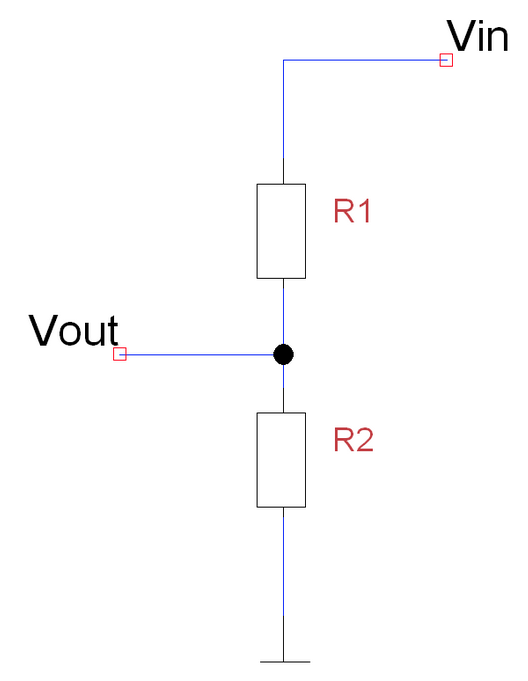
La formula que aplicaremos sera la siguiente que es el proporcional al tiempo que tarda el sonido en recorrer la distancia desde el emisor al obstáculo y volver, podríamos simplificar la formula al tratarse de constantes .

1. Calculo de la distancia

El sistema se basa en el uso de los puertos GPIO de la RaspberryPi los cuales hemos definido anteriormente en la sección hardware, para manejar este sensor es necesario hacer uso de una librería externa Pi4J la cual nos da acceso a una serie de métodos con los cuales podremos controlar los Puertos GPIO Leer y escribir información de ellos así como activarlos y desactivarlos.

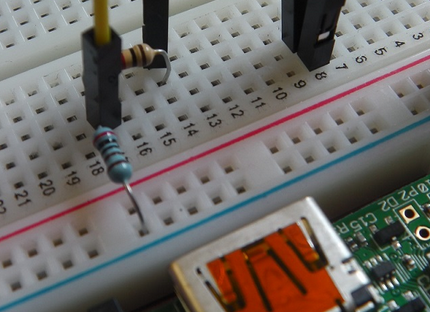
El diagrama de montaje es relativamente sencillo, necesitaremos dadas las características de los puertos GPIO dos resistencias que pondremos en según el siguiente esquema.

El calculo de la magnitud de las resistencias siguen el siguiente patrón

Ilustración 34: Circuito salida HC-SR04

así pues podemos concluir que aunque si que es importante el valor de las resistencias estas cumple unos características que relacionaran los dos valores, para nuestro caso usaremos resistencias de 1K y de 2K.

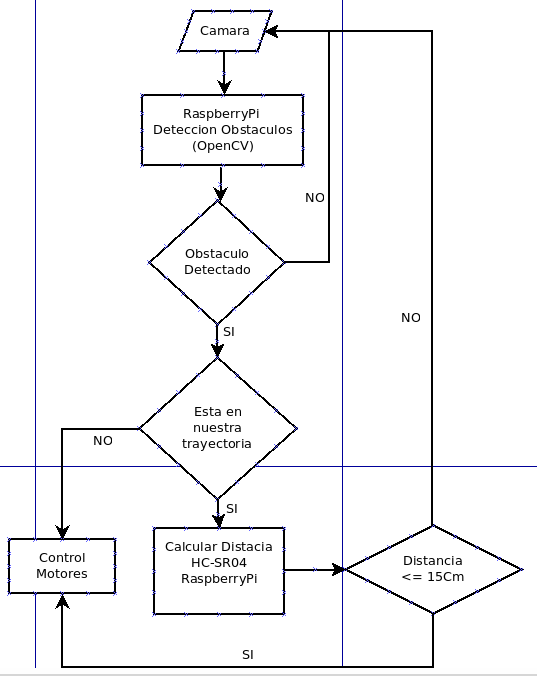
Ilustración 36: HC-SR04 Conectado a RaspberryPi

Ilustración 35: Detalle resistencias en la conexión de salida

1. Diagrama de procesamiento

Partiendo del diagrama anterior completaremos el proceso aplicando una nueva comprobación al proceso, esta es la distancia y según los datos que tengamos Actuaremos de una forma u otra.

Como podemos observar completamos el diagrama anterior con unos nuevos pasos en los que decidimos que vamos a hacer en caso de que nos encontremos un obstáculo.

Ilustración 37: Diagrama de proceso con cálculo de distancia

Como punto mas importante de todo este proceso es saber a que distancia estamos del obstáculo, definimos un limite de parada de 15cm aunque este sera configurable en puntos posteriores.

Como primer paso solo actuaremos de dos formas inicialmente continuar el recorrido y parada de motores.

El calculo de la distancia como hemos indicado anteriormente lo obtenemos por medio de la siguiente formula pero la pregunta es como obtenemos los datos para usar la formula, los obtendremos mediante el siguiente procedimiento.

Este es una versión simplificada del que usaremos en la versión final, puesto que en este no implementamos toda la funcionalidad de timeouts ni seguridad.

|  |
| --- |
| public double CalcularDistancia()  {  final GpioController gpio = GpioFactory.getInstance();  final GpioPinDigitalOutput trigPin = gpio.provisionDigitalOutputPin(RaspiPin.GPIO\_00, "Trig", PinState.LOW);  final GpioPinDigitalInput echoPin = gpio.provisionDigitalInputPin(RaspiPin.GPIO\_02, "Echo");  double tiempoInicio = 0d, TiempoFin = 0d, distanciaObstaculo = 0d;  trigPin.high();  try  {  Thread.sleep(0, 10000);  System.out.println(".");  }  catch (Exception ex)  {  ex.printStackTrace();  }  trigPin.low();  while (echoPin.isLow())  {  tiempoInicio = System.nanoTime();  }    while (echoPin.isHigh())  {  TiempoFin = System.nanoTime();  }    if (tiempoInicio > 0 && TiempoFin > 0)  {  double duracionPulso = (TiempoFin - tiempoInicio) / 1000d; // en MicroSegundos  distanciaObstaculo = duracionPulso \* 0.017;  }  return distanciaObstaculo;  } |

1. Detección de obstaculos basicos

Para el control del movimiento de nuestro vehículo crearemos un programa que nos permita movernos con libertad sobre el plano mandando un serie de ordenes básicas desde el control de de entorno.

Dentro de este punto tenemos varias alternativas las cuales tienen puntos buenos y malos y cada una tiene sus peculiaridades de implantación, nos hemos decidido por la alternativa de implantar un Puente H L298N.

Al igual que nos ha pasado con los sensores en el punto anterior tendemos que recurrir a una librería externa para el manejo del GPIO y con este el manejo de los interfaces de control de movimiento.

1. Puente H L298N

Ya hemos hablado de este componente en un punto anterior referente al Hardware utilizado, este componente se nos presenta como uno de los interfaces mas simples que podemos usar, se trata de un componente estándar y económico que nos permite controlar dos motores DC de forma sencilla.

Encontramos varios problemas en este interfaz :

* Control de PWM: Este interfaz proporciona una funcionalidad básica de PWM (Modulación por ancho de pulsos) pero esta funcionalidad esta supeditada al elemento superior de control, es decir si nuestro controlador, en este caso RaspberryPi, no dispone de Control de I/O por PWM tendremos que implementarlo por Software lo cual conducirá a una perdida de rendimiento que puede afectar a todo el sistema.
* Sobrecalentamiento : Este tipo de unidades es conocida por su calentamiento extremo y consecuentemente su perdida de funcionalidad.
* Regulador de tensión : Este interfaz proporciona un regulador de tensión auxiliar de 5v, suficiente para alimentar nuestro interfaz de control, pero esta es poco fiable cuando se calienta en exceso produciendo cortes de corriente que pueden llegar a dañar el controlador del sistema

1. Control de Giro

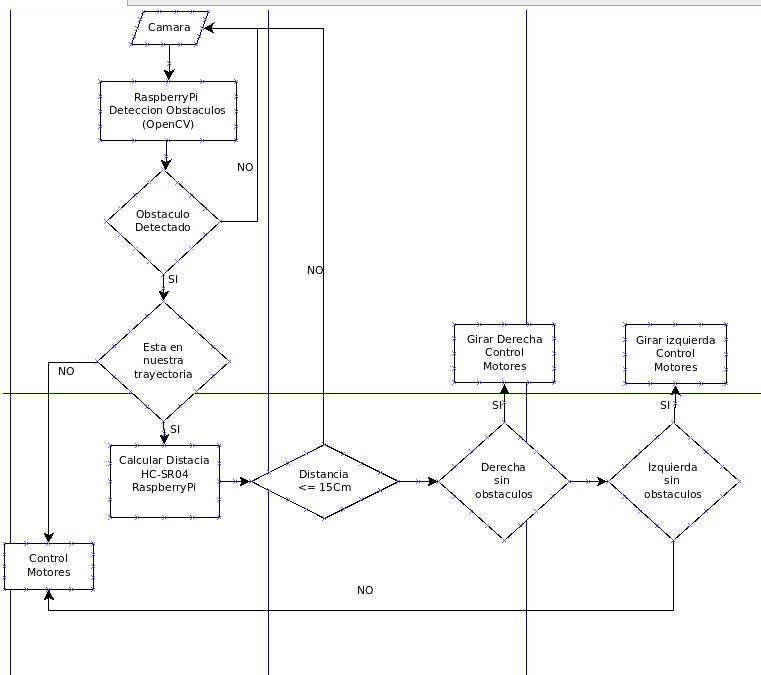
El control de giro que implementaremos en este proyecto es un sistema de dirección tipo oruga, este tipo de control de dirección se basa en el mismo sistema con el que se mueven la excavadoras que no disponen de ruedas direccionales.

Para girar a la derecha reducen la velocidad de rotación de las ruedas derechas, las paran o incluso invierten el sentido según la velocidad que quieran de giro.

Para girar a la izquierda hacen exactamente lo mismo que en el caso anterior pero invirtiendo el lado.

1. Diagrama de procesamiento

Como en los puntos anteriores vamos a mostrar los cambios implementados en el proceso desde este punto.

Ilustración 38: Control de giro

Como podemos ver implementamos en este punto un control de dirección básico con giro a la derecha y a la izquierda en caso de encontrar un obstáculo en nuestra ruta actual.

Ante el caso de no poder encontrar un ruta valida, tanto en la ruta actual como en la derecha y en la izquierda detendremos la marcha en pasos posteriores encontraremos una solución a este problema.

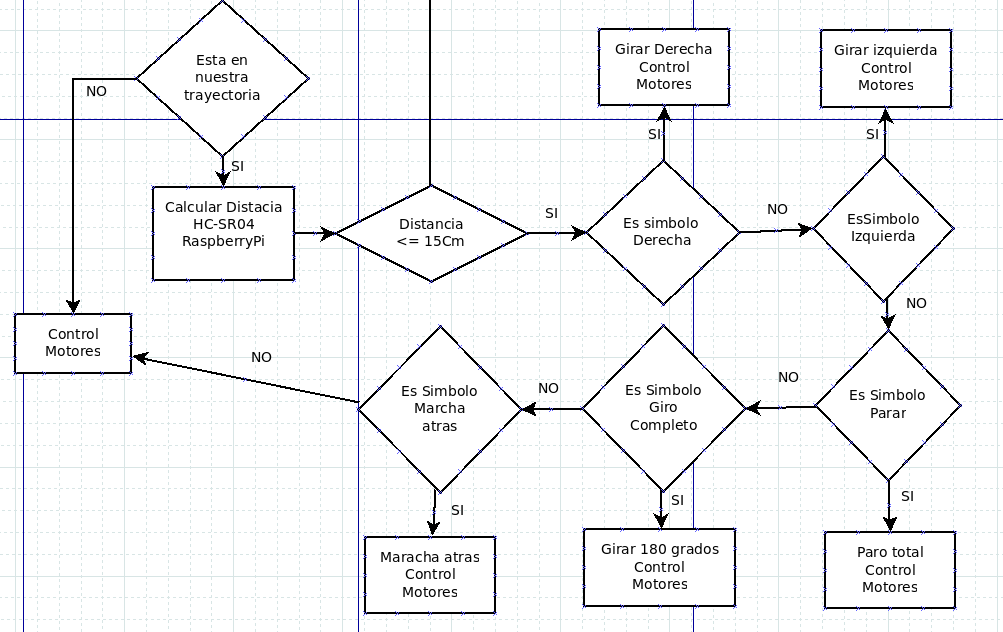
1. Control sentido de la marcha, giro completo

En puntos anteriores hemos definido un protocolo básico de movimiento, pero realmente se no queda insuficiente para ser completamente funcional por ello implementamos o modificamos el procedimiento para que nos sea mas útil y completo.

Nos basamos al igual que el anterior en los sensores y la cámara pero en este aso lo que utilizaremos serán los patrones que hemos entrenado, lo que nos permitirá saber con seguridad que tipo de movimiento queremos realizar .

1. Diagrama de procesamiento

Dado que en puntos anteriores ya hemos tratado parte del diagrama solo mostraremos la parte que cambia

Ilustración 39: modificación del control de movimiento

Como podemos observar en este caso compararemos si existen los obstáculos específicos que nos indicaran que movimiento tenemos que ejecutar, esto nos permitirá diseñar rutas de un forma sencilla únicamente colocando distintas señales en la habitación que el auto seguirá y ejecutara.

2. Control comunicaciones
3. Control Programa Principal
4. Desarrollo
5. Programa control RaspberryPi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Paquete duxmancar** | | | | |
|  | | | | |
| Este es el paquete inicial del proyecto, es decir contiene el método main las propiedades principales del programa y un pequeño modulo de test que nos permite pasas pruebas independientes de cada uno de los módulos de los que se compone el programa.  Como punto interesante a la vez importante es que la clase DuxmanCar es la encargada de instanciar todas los objetos referentes a Hardware , el puerto GPIO por WiringPi, el GestorI2C, la Cámara, el Puerto GPIO por JDK I/O, Todos estos gestores son clases de tipo Singleton.  Esto es importante de tener en cuenta puesto que a lo largo de todo el programa estamos trabajando con múltiples hilos, desgraciadamente muchos de estos objetos no son ThreatSafe y por o tanto no pueden ser accedidos desde distintos hilos al mismo tiempo, por esta razón el diseño de todos las clases controladoras de un elemento hardware están diseñadas como Singleton de esta forma podemos grantizar que no existen multiples instancias del mismo objeto en distintos Threats. | | | | |
| **Paquete duxmancar.datos** | | | | |
|  | | | | |
| Este paquete contiene todos los elementos referentes a los datos que se transfieren entre los distintos subsistemas . | | | | |
| **Clase** | **Descripción** | | | |
| Cdato  CDatoProvider | Estas son la clase básica de información dentro del sistema, todos los mensajes que se tratan dentro del sistema son instancias de estas clases.  Como punto importante a destacar:  CdatoProvider es una clase Abstracta que nos proporciona los datos básicos así como sus funciones de acceso, lo cual no permite extender y crear un nuevo tipo de datos compatible pero con información ampliada.  Cdato implementa los métodos abstractos de su padre CdatoProvider y realiza el trabajo de descodificar los mensajes. Por otro lado esta clase tiene un método estático que nos permitirá codificar un nuevo mensaje desde los parámetros aportados. | | | |
| CTelegramaAutomatico | Esta clase realiza invocaciones al método Estático cd Cdato para codificar mensajes, es usada desde el sistema de conducción automática y nos proporciona los datos necesarios para que el coche maniobre de forma autónoma. | | | |
| ClistaDatoProvider | Puesto que disponemos de varios métodos de conexión redes lan, redes Wan , y redes Pan , es decir internet, red local y Bluetooth , este hecho complica sobre manera el proceso del programa para ello se genera una única clase Singleton que permitirá aunar todos los mensajes que lleguen al sistema en una única cola FIFO que podremos tratar de forma mas sencilla.  Esta lista es utilizada tanto por los sistemas de comunicación para introducir y mandar mensajes como por los sistema de conducción tanto autónoma como guiada para obtener y generar las ordenes que los controladores de motor ejecutaran. | | | |
| **Paquete duxmancar.Datos.Procesadores** | | | | |
|  | | | | |
| Este paquete contiene los controladores de datos hacia la capa hardware que procesan los mensajes y posteriormente mandan ejecutar las ordenes a los controladores hardware propiamente dichos | | | | |
| Clase | | Descripción | | |
| CprocesadorDatosProvider | | Clase abstracta que implementa la cola de mensajes FIFO y los métodos sincronizados para acceder y a los datos. | | |
| CpuenteHControl | | Clase que implementa el controlador de puente H descrito en la sección hardware, nos permite tratar los mensajes específicos con destino el controlador Puente H para hace funcionar los motores DC | | |
| CservoControl | | Clase que implementa el controlador de motores Servo y permite al igual que la anterior procesar los datos que terminaran convirtiéndose en movimiento de de los Servo Motores. | | |
| **Paquete duxmancar.log** | | | | |
|  | | | | |
| Este paquete únicamente contiene la clase que nos cargara la configuración de log y no da acceso a las funcionalidades de escritura en los ficheros de log de la aplicación | | | | |
| **Paquete duxmancar.net** | | | | |
|  | | | | |
| Este se trata de uno de los paquetes principales de la aplicación, en este paquete diseñamos todo el sistema de comunicaciones asíncrono para por el cual recibiremos ordenes desde los distintos elementos de control y emitiremos información hacia ello de los eventos que sucedan en el sistema.  Cabe destacar que el sistema se basa en una comunicación asíncrona y multitrhreat en la cual separamos tanto el envío de como la recepción de datos los cuales se gestionaran en cola FIFO en la clase comentada anteriormente ClistaDatoProvider.  Como característica a destacar es la forma en la que aunamos todos los elementos de comunicación generales en una clase abstracta la cual es heredada por los distintos tipos de sockect que implementamos. | | | | |
| Clase | | Descripción | | |
| CsockectServer | | Clase que gestiona las colas de entrada y salida, así como los buffer de escritura y lectura.  Implementa internamente dos subclases para gestionar las comunicaciones de entrada y salida.  Implementa metodos generales genéricos que permiten obtener los buffer de lectura y escritura desde los distintos tipos de sockect | | |
| ChiloEntrada  ChiloSalida | | Clases internas que controlan el flujo de información hacia los dos sentidos gestionan la cola de datos ClistaDatoProvider para obtener y dejar mensajes. | | |
| CnetServer | | Clase Servidor Sockect LAN/WAN implementa el hilo de gestión de conexión de clientes | | |
| CbtServer | | Clase Servidor Sockect Bluetooth para redes PAN que implementa el hilo de gestión y descubrimiento de dispositivos Bluetooth. | | |
| CsockectMulticast | | Clase Sockect cliente servidor utilizada para el auto descubrimiento de dispositivos de control, esta clase permite realizar la conexión inversa haciendo de cliente y que sean los dispositivos Android de control los servidores de datos.  Falta implementar flujo de conexión, actualmente en desarrollo. | | |
| **Paquete duxmancar.Raspberry.hardware** | | | | |
|  | | | | |
| Este paquete únicamente contiene una clase de test que permite probar los sensores programados y los motores por medio de un menú de opciones. | | | | |
| Clase | | | | Descripción |
|  | | | |  |
| **Paquete duxmancar.Raspberry.hardware.ControlMotores** | | | | |
|  | | | | |
| En este paquete se definen las clases que nos permiten controlar a nivel físico los motores o elementos hardware de los que se | | | | |
| Clase | | | Descripción | |
| CmotorControlPuenteH | | | Esta clase controla el par de motores independientes aunando las ordenes de manera que los dos motores trabajen de forma conjunta para generar los distintos movimientos.  Contiene las funciones especificas para generar los movimientos de los dos motores de forma conjunta. | |
| CmotorDC | | | Clase de control de un motor DC permite activar y desactivar el motor así como girar en una dirección u otra y modificar la velocidad de rotación RPM de los motores mediante pulsos PWM  Cada uno de los motores esta gestionado por 3 pines uno para gestionar el movimiento hacia delante, otro para gestionar el movimiento hacia atrás y el ultimo es el pin de control que permite gestionar los pulsos PWM que controlan la velocidad. | |
| CGestor12CAdafruit | | | Esta clase permite controlar una interfaz i2c de adafruit para control de servo Motores, debido a las peculiaridades del interfaz i2c y específicamente los interfaces de adafruit es necesario codificar una clase de control de dicho interfaz que permita traducir las ordenes generadas en nuestro sistema a el sistema codificado del bus i2C de adafruit, aunque i2c es estándar algunos interfaces no pueden ser controlados directamente por los dispositivos, en nuestro caso raspberryPi con la librería WiringPi , es por esto que nos vemos en la necesidad de implementar una clase intermedia que nos haga de Wrapper de comunicaciones. | |

|  |  |
| --- | --- |
| Dusmancar.Raspberrry.Hardware.GPIO | |
|  | |
| Paquete de control de elementos hardware dependientes de GPIO | |
| Clase | Descripción |
| CgestorGPIO | Esta clase permite crea un sistema sincronizado y bloqueante del Puerto GPIO de la raspberry Pi para evitar el uso del dispositivo desde varios threat al mismo tiempo.  Implementa un seguro de bloqueo que impide que otros hilos actúen sobre el puerto GPIO mientras estamos trabajando con el. |
| CGestorI2CSincronizado | Igual que el anterior creamos una clase que permita comunicarnos con el bus i2c de forma sincronizada y bloqueante que impide el acceso a otros hilos procesos al bus i2C para evitar posibles problemas y core Dumps. |

|  |  |
| --- | --- |
| **Paquete duxmancar.Hardware.Raspberry.Sensores** | |
|  | |
| Paquete superior que contiene la implamanetación de la clase básica de sensores.  Se trata de una super clase que extiende de threat y únicamente implementa los métodos necesarios para identificar un sensor y el numero de mediciones que tiene que hacer para obtener un dato valido. | |
| Clase | Descripción |
|  |  |

|  |  |
| --- | --- |
| **Paquete duxmancar.Hardware.Raspberry.Sensores.distancia** | |
|  | |
| Paquete superior que contiene la implementación de la clase básica de sensores.  Se trata de una super clase que extiende de threat y únicamente implementa los métodos necesarios para identificar un sensor y el numero de mediciones que tiene que hacer para obtener un dato valido. | |
| Clase | Descripción |
| Csensor | Clase abstracta que contiene los métodos básicos a todos los sensores.  Implementa el tipo de sensor, para permitir extender el proyecto añadiendo distintos sensores.  Implementa el numero de pulsos a ejecutar, definimos pulso como una medición individual de un sensor, para poder realizar un media y asi obtener un resultado valido evitando las aberraciones o los datos erróneos. |
| CmedidorDistancia | Clase que implementa el control del dispositivo hardware HC-SR04, sensor de distancia por ultrasonidos que ya hemos detallado en puntos anteriores.  Esta clase implementa la medición de la distancia por medio de pulsos y realiza el calculo de estas mediciones. Retornando una media aritmetica de los datos obtenidos. |
| **Paquete duxmancar.Hardware.Raspberry.Sensores.distancia** | |
|  | |
| Paquete que implementa las clases necesarias para la gestión de la cámara, además en este paquete definimos una serie de clase adicionales necesarias para los procesos que tenemos que generar. | |
| Clase | Descripción |
| CCamara | Clase principal que nos ofrece una interfaz con la que obtener imágenes , hace la función de sistema de visión artificial controlado por el resto de clases.  Actualmente se han desarrollado los siguientes detectores:   * Formas Entrenadas HaarCascade * Colores Filtro HSV * Obstáculos genéricos con Bloobs * Formas fijas Círculos y cuadrados   Se trata de una clase singleton para evitar el acceso desde distintos hilos al mismo hardware.  Debido a las peculiaridades del BUS USB trabajando con WebCams y linux no hemos podido implementar un bloqueo que nos asegure que somos los únicos que estamos usando el dispositivo, por esta razón se implementa funciones sincronizadas que nos permitan asegurar que al menos este hardware no es accedido desde dos hilos distintos al mismo tiempo. |
| CDetectorObstaculos | Se trata de una clase abstracta utilizada para gestionar todos los detectores de obstáculos que hemos definido anteriormente. |
| CDetectorCirculos  CDetectorFormas | Estas dos clases contienen las llamadas a la la clase cámara y son las encargadas de gestionar las respuestas especificas para la conducción automática |
| Bloobs | Esta es una clase que hemos tenido que implementar, puesto que la versión de OpevCV que estamos utilizando no implementa detección bloobs, se puede definir bloob como formas genéricas, la usamos para permitir la detección de formas por color o simplemente de mediante threshold |
| Cobstaculo | Clase estática en la que guardaremos los resultados de las detecciones de obstáculos |

|  |  |
| --- | --- |
| **Paquete duxmancar.Hardware.Raspberry.Sensores.distancia** | |
|  | |
| Capa superior de programa de control de lasraspberryPi y del coche autónomo esta clase es la que contiene el hilo principal del programa y controla todos los gestores de los elementos conectados a raspberryPi | |
| Clase | Descripción |
|  |  |

1. Programa Control Android

El programa de control android es un evolución de una practica que realice el año pasado presentada para la asignatura de dispositivos móviles con David Chinarro, hemos aprovechado la interfaz y el core del programa para adaptarlo a nuestras necesidades actuales.

|  |  |
| --- | --- |
| **Paquete org.duxman,gui** | |
|  | |
| Este es el paquete principal del programa de control android, implementa todas las actividades, pantallas del programa, además implementa las clases clientes de los sockect de conexión.  Por otra lado implementa todos los eventos que se ejecutan en el programa. | |
| Clase | Descripción |
| CblueClient | Clase principal del programa android, es la interfaz principal del programa he implementa todos los mandos de control necesarios para controlar el vehículo.  También implementa los controles de conexión con nuestro coche permitiéndonos conectarnos mediante bluetooth o wifi , implementa un pequeño servidor de sockect que permitirá realiza la conexión inversa siendo el dispositivo android el servidor y el coche el cliente , aun se esta desarrollando esta funcionalidad, aunque la conexión es funcional es necesario realizar modificaciones en el sistema de control de la raspberryPi. |
| CtecladoTelegramas | Clase que implementa un sistema de control adicional pero mediante el envío exclusivo de telegramas individuales, hace las funciones de programa de test que no permite comprobar si la comunicación entre nuestro dispositivos es correcta. |
| CconectarNet | Activity que nos ofrece la pantalla de conexión a red , en esta pantalla podremos introducir la ip y el puerto de conexión de nuestro sistema. |
| DeviceListActivity | Lista en la cual mostraremos todos los medios de conexión disponibles y que podremos selecciona mediante un pulsación, lanzando la activity necesaria para completar la conexión deseada. |

|  |  |
| --- | --- |
| **Paquete org.duxman.net** | |
|  | |
| Paquete que implementa todo el sistema de conexión utilizado por el programa android.  Aunque Android esta basado en java y nuestro programa de control de la raspberryPi también esta desarrollado en java no se ha podido reutilizar el código debido a las peculiaridades existentes en android sobre los métodos de conexión. | |
| Clase | Descripción |
| CblueService | Clase principal de conexión por bluetooth, esta clase nos ofrece la pasarela necesaria para conectarnos por bluetooth a nuestra raspberryPi |
| CblueServiceConectThreat | Threat que implementa la logica de conecxión a un sockect bluetooth.  Este hilo cede el control a CblueServiceConectedThreat. |
| CblueServiceConectedThreat | Threat que mantiene y comprueba si la conexión con nuestro dispositivo continua abierta |
| CblueServiceAcceptThreat | Threat que esta escuchando para aceptar conexiones, este hilo cede el control a CblueServiceConectThreat. |
| CservidorNetSockect | Clase que implementa el servidor Sockect para redes Lan , la lógica del programa funcionara exactamente igual, con la salvedad que sera el dispositivo android quien controla comunicación. |
| CclienteNetSockect | Clase que implementa el cliente Socket de redes Lan |
| CTelegrama | Clase que nos ofrece una interfaz para crear Telegramas o datos que enviaremos desde nuestro dispositivo andoid a nuestra RaspberryPi. |

|  |  |
| --- | --- |
| **Paquete org.duxman.uitl** | |
| En este paquete simplemente implementamos la clase telegrama comentada anteriormente y las constantes utilizadas en todo el programa y comunicación entre nuestro dispositivo android y nuestra raspberryPi | |
|  | |
| Clase | Descripción |
|  |  |

|  |  |
| --- | --- |
| **Paquete org.duxman.util.list** | |
|  | |
| En este paquete implementamos todas las clases que hacen referencia a listas gráficas, son objetos que usaremos en las listas del interfaz gráfico | |
| Clase | Descripción |
| CbtDeviceItem | Representa un dispositivo bluetooth detectado, cuando lanzamos el descubrimiento de elemento bluetooth. |
| CarrayAdapterBtDevices | Adaptador de lista , para interfaz gráfico en android , nos permite seleccionar un item y nos retorna los valores deseados. |
| CconexiónItem | Items de elementos de conexión , nos permite terner los elementos de conexión como si una lista se tratara e interactuar con ellos en el entorno grafico. |

1. Análisis Coste del proyecto

El análisis de coste los vamos a dividir en varias secciones I+D, Análisis, Diseño software, Diseño Hardware, cada una de estas fases tiene una serie de gastos los cuales dividiremos en varias secciones según a que parte se impute el gasto.

Podremos observar que hemos realizado el estudio de gasto desde un punto de vista de divisón departamental en la cual entran en juego distintas fases de proyecto, I+D, Producción, Testing , Diseño ….

También se ha realizado un pequeña división para poder separar los costes humanos de los materiales , de esta forma nos podremos hacer una idea de cuanto es el coste real del proyecto y cuanto es el coste imputable a los materiales físicos que necesitamos.

1. I+D

Esta fase es una mezcla tanto de Diseño Software, como de diseño Hardware puesto que hemos tenido que implementar varias alternativas, diseñar software específico y construir varias maquetas de prueba, aunque muchos de los elementos son comunes, se han tenido que probar otros muchos para valorar, resultados y complejidad.

1. Coste elementos Hardware

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento** | **Observaciones** | **Coste** | **Unid.** | **Ok** |
| **Servo Motores Tower 995** | Servomotores de baja calidad, ha sido necesario hacerles un hack para permitir la rotación continua de 360º, es imposible su calibración | 4,10€ | 4 |  |
| **MiniServo 9G** | Servomotor potente aunque pequeño , dificulta en exceso la adaptación de las ruedas al ser tan pequeños, la potencia suministrada es escasa y se estropean con facilidad al girar demasiado rápido , la calibración es muy compleja y se estropea con facilidad | 3,90€ | 4 |  |
| **Servomotor de rotación continua SM-S4303R** | Servomotor de gran calidad y de fácil calibración, tenemos la dificultad añadida de adaptar las ruedas a este tipo de motores | 13 | 4 |  |
| **Motor DC TT Encoder** | Motor DC Pequeño pero potente, de fácil manejo , viene con una rueda adaptada de buena calidad que nos permite usarlo directamente en el proyecto | 4,60 | 4 |  |
| **PuenteH L298N** | Placa de control para motores DC que nos da la capacidad de usar PWM en motores DC que por norma general no permiten este tipo de uso. | 6.30 | 1 |  |
| **Adafruit 12-Channel 16-bit PWM LED Driver** | Controlador PWM para motores y Leds, como problema encontramos la necesidad de adaptar un fuente de alimentación externa para dar potencia a los motores. Solo es válido para ServoMotores | 7.5€ | 1 |  |
| **Adafruit 16-Channel 12-bit PWM/Servo Driver** | Controlador I2C para servomotores al contrario que el anterior está preparado para adaptar una alimentación externa, tenemos un problema a nivel de implementación software al tener que implementar una solución para controlar este dispositivo puesto que no esta soportado de forma estándar | 16€ | 1 |  |
| **Sensor de Ultrasonidos HC-SR04** | Sensor de ultrasonidos para medir distancias | 4.74€ | 1 |  |
| **Consumibles electrónica** | Para adaptar los y controlar los diversos elementos hardware necesitamos diversos componentes electrónicos , resistencias , condensadores , cableas estaño , placas de prototipo… | 20€ | 1 |  |
| **Interfaz BattBorg, transformador electrónicos 5V** | Con el fin de dar potencia a nuestra raspberry pi por medio del puerto GPIO, necesitaremos este componente | 5,80€ | 1 |  |
| **WebCamara USB Logitech C170** | Optamos por cámaras USB al darnos mayor calidad y rendimiento , Se compran dos cámaras para probar la posibilidad de Visión Estero | 8.90 | 2 |  |
| **Coste Montaje prototipos** | Se construyen y diseñan varios prototipos para comprobar las distintas configuraciones y elementos seleccionados , valoramos el coste medio de hacer un prototipo y el número de prototipos calculamos que el tiempo medio es de 10Horas y el coste por hora lo valoramos en 18€ | 180€ | 3 |  |
| **Total** |  | 720,54 € | | |

Tabla 3: Costes imputables a I+D Hardware

Explotamos un poco más el ítem de diseño de prototipos para analizar como dividimos los costes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento** | **Observaciones** | **Coste** | **Unid.** | **Ok** |
|  |  |  |  |  |
| **Documentación Diseño** | Búsqueda de información necesaria para el diseño de los prototipos | 18 | 2 |  |
| **Diseño placa pruebas** | Diseño mediante software de la placa y pruebas a nivel simulación | 18 | 3 |  |
| **Montaje Circuito** |  | 18 | 3 |  |
| **Pruebas** |  | 18 | 3 |  |
| **Subtotal Coste Montaje prototipos** | Se construyen tres prototipos | 180€ | |  |
| **Total** |  | 540€ | |  |

Tabla 4: Explotación Coste Prototipos

1. Coste Diseño Software

Dentro de este punto vamos a calcular el coste a nivel software

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento** | **Observaciones** | **Coste** | **Unid.** | **Ok** |
|  |  |  |  |  |
| **IDE NetBeans** | Entrono de desarrollo utilizado para la programación del software de control de raspberryPi | 0 | 1 |  |
| **IDE Eclipse** | Entrono de desarrollo para la aplicación Android de control | 0 | 1 |  |
| **Librerías de desarrollo Pi4J** | Librerías de desarrollo para el control de GPIO de RaspberryPi | 0 | 1 |  |
| **Librerías de desarrollo de Visión electrónica** | Se valoran varias librerías OpenCV, JavaCV y BOOTF | 0 | 3 |  |
| **Estudio y documentación** | Estudio y documentación de las distintas librerías desarrollo | 18€ | 120 |  |
| **Análisis Programas control prototipos** | Análisis de los programas de control de prototipos | 18€ | 40 |  |
| **Diseño Programas** | Diseño y programación de los programas | 18€ | 40 |  |
| **Pruebas Programas** | Pruebas realizadas para testear la validez de los prototipos | 18€ | 20 |  |
| **Total** |  | 3960€ | |  |

Tabla 5: Costes imputables a I+D Software

1. Diseño de Hardware

Un vez que nos hemos decidido por una tecnología y unos elementos de software y hardware llega el punto de analizar los costes de crear un prototipo final.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento** | **Observaciones** | **Coste** | **Unid.** | **Ok** |
|  |  |  |  |  |
| **Base Montaje** | Base de Metacrilato para el montaje del prototipo final | 25€ | 1 |  |
| **Tornillería** | Piezas de tornillería para el montaje del prototipo final | 5€ | 1 |  |
| **Cableado** | Cable y diverso material eléctrico | 5€ | 1 |  |
| **Montaje del prototipo** | Tiempo invertido en el montaje del prototipo y su adaptación | 18€ | 4 |  |
| **Baterías** | Baterías y porta-baterías utilizado en el prototipo final | 10€ | 1 |  |
| **Transformador Corriente** | Trasformador de corriente de 12V utilizado en las pruebas | 15€ | 1 |  |
| **Pruebas** | Pruebas realizadas para comprobar la funcionalidad | 18€ | 25 |  |
| **Total** |  | 582€ | |  |

Tabla 6: Costes imputables al hardware

1. Diseño de Software

Un vez que nos hemos decidido por una tecnología y unos elementos de software y hardware llega el punto de analizar los costes de crear un prototipo final.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elemento** | **Observaciones** | **Coste** | **Unid.** | **Ok** |
|  |  |  |  |  |
| **IDE NetBeans** | Entrono de desarrollo utilizado para la programación del software de control de raspberryPi | 0 | 1 |  |
| **IDE Eclipse** | Entrono de desarrollo para la aplicación Android de control | 0 | 1 |  |
| **Librerías de desarrollo Pi4J** | Librerías de desarrollo para el control de GPIO de RaspberryPi | 0 | 1 |  |
| **Librerías de desarrollo de Visión electrónica** | Valoramos en la etapa de de I+D la librería que mejor se adaptaba a nuestras necesidades era OpenCV | 0 | 3 |  |
| **Documentación de análisis funcional y antecedentes** | Documentación referente al proyecto | 18€ | 200 |  |
| **Análisis del software de control ( RaspberryPi)** | Análisis del programas de control del prototipo final | 18€ | 50 |  |
| **Análisis del Software de gestión (Android)** | Análisis del software de gestión del prototipo final, puesto que aprovechamos un software existente, creado anteriormente,  Practica 4 Aplicaciones Móviles realizada el año anterior , simplemente tendremos que adaptar la aplicación a las nuevas necesidades. | 18€ | 5 |  |
| **Diseño Software** | Diseño e implementación de los distintos programas que componen el sistema | 18€ | 175 |  |
| **Pruebas** | Pruebas realizadas para comprobar la funcionalidad | 18€ | 25 |  |
| **Total** |  | 8190€ | |  |

Tabla 7: Costes imputables al software

1. Análisis visual

En este apartado vamos a mostrar unas gráficas visuales sobre la distribución de los costes de esta forma podremos apreciar de una forma más sencilla cuanto es el coste total del proyecto y que coste tienen las diversas fases del proyecto

1. Gráfica distribución

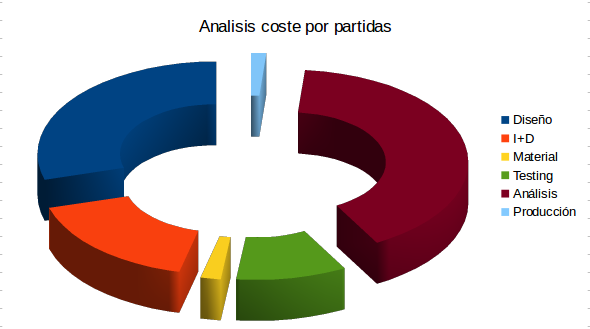
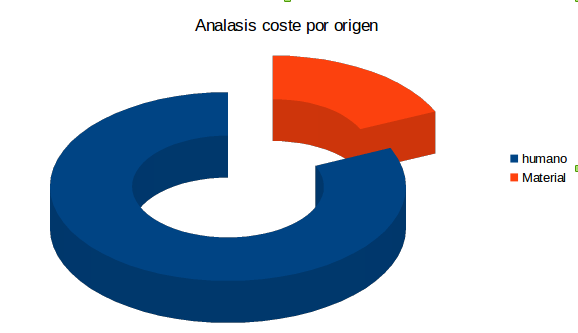
Ilustración 40: Gráfica distribución costes proyecto

Ilustración 41: Gráfica distribución costes por origen

1. Coste clasificado

|  |  |
| --- | --- |
| **Coste por partida** | **Suma de Base** |
| **Diseño** | **3870** |
| Diseño Programas | 720 |
| Diseño Software | 3150 |
| **I+D** | **2196** |
| Documentación Diseño | 36 |
| Estudio y documentación | 2160 |
| IDE Eclipse | 0 |
| IDE NetBeans | 0 |
| Librerías de desarrollo de Visión electrónica | 0 |
| Librerías de desarrollo Pi4J | 0 |
| **Material** | **240,54** |
| Adafruit 12-Channel 16-bit PWM LED Driver | 7,5 |
| Adafruit 16-Channel 12-bit PWM/Servo Driver | 16 |
| Base Montaje | 25 |
| Baterías | 10 |
| Cableado | 5 |
| Consumibles electrónica | 20 |
| Interfaz BattBorg, transformador electrónicos 5V | 5,8 |
| MiniServo 9G | 15,6 |
| Motor DC TT Encoder | 18,4 |
| PuenteH L298N | 6,3 |
| Sensor de Ultrasonidos HC-SR04 | 4,74 |
| Servo Motores Tower 995 | 16,4 |
| Servomotor de rotación continua SM-S4303R | 52 |
| Tornillería | 5 |
| Transformador Corriente | 15 |
| WebCamara USB Logitech C170 | 17,8 |
| **Testing** | **1314** |
| Pruebas | 954 |
| Pruebas Programas | 360 |
| **Análisis** | **5310** |
| Análisis del software de control ( RaspberryPi) | 900 |
| Análisis del Software de gestión (Android) | 90 |
| Análisis Programas control prototipos | 720 |
| Documentación de análisis funcional y antecedentes | 3600 |
| **Producción** | **180** |
| Diseño placa pruebas | 54 |
| Montaje Circuito | 54 |
| Montaje del prototipo | 72 |
| **Total general** | **13110,54** |

Tabla 8: Análisis del coste del proyecto

1. Evolución del proyecto

En este punto destacamos las posibles mejoras de este proyecto pues consideramos que en este momento no tenemos tiempo suficiente como para desarrollar todas las funcionalidades que hemos encontrado.

1. Sensor de distancia

Dentro del proyecto se ha implementado un sensor de distancia por ultrasonidos orientado a largo alcance y con rangos de 3cm a 3m posicionado en la parte dentera del vehículo, pero esto no lo consideramos suficiente puesto que evidentemente tenemos puntos ciegos en la parte trasera del vehículo y que según el ángulo en el que se encuentren los obstáculos pueden producir falsos negativos.

Existen otros sensores por ejemplo Sharp GP2Y0A02YK0F que se trata de un sensor de rango largo por infrarrojos con detección continua de proximidad, este sensor trabaja en rangos de 15cm a 1.50m.

Seria posible eliminar los puntos muertos de los sensores instalando mas sensores que complementen el sistema actual.

1. Visión electrónica

Dentro de este campo existen muchas posibles mejoras, la cuales detallaremos a continuación, que no se han podido implementar por falta de tiempo

1. Ángulos muertos cámaras

La solución implementada incluye un cámara en ella parte delantera del coche que nos permite tener constancia de los obstáculos que tenemos enfrente nuestro, pero que pasa si el obstáculo o el peligro aparece por los lados y si vamos marcha atrás.

Por lo tanto parece evidente e interesante pensar en la posibilidad de ampliar el sistema de cámaras por lo menos delante y detrás del vehículo las cuales nos den información del entorno en el que nos movemos.

En otro punto tenemos el tema de ángulos muertos, este tema se podría solucionar implementando un sistema de visión panorámica controlada por un motor o implementando un sistema de visión con lentes ojo de pez que nos de un ángulo de visión mayor.

1. Detección de objetos genéricos

Este es otro punto que desgraciadamente hemos tenido que dejar en el tintero por motivos evidentes, aunque también se ha investigado dentro del proceso de análisis no hemos podido llega a implementar todas las funcionalidades que nos hubiera gustado.

Este paso es interesante pues existen diversas formas de avanzar desde la detección de formas genéricas y detección de bordes a la posibilidad de construir un sistema experto que sea capaz e aprender y recordar diversos obstáculos que se ha encontrado anteriormente y actuar en consecuencia.

1. Visión 3d

Aunque se ha hecho un estudio bastante avanzado sobre este tema dentro del proyecto no se ha conseguido el objetivo de hacer un modelo funcional de esta tecnología que consideramos suficientemente importante como para ser la protagonista principal de un proyecto general entero.

Esta mejora permitiría junto con la ayuda de diversos sensores establecer un mapa tridimensional con el cual seriamos capaces de movernos y calcular trayectorias así como obtener resultados mas que interesantes, como mapas en 3d , proyecciones de objetos …

Dentro de este tema también tendremos que de capaces de gestionar el procesamiento de todos los cálculos necesarios para conseguir una proyección 3D desde dos imágenes planas, para lo cual la plataforma elegida en este caso, RaspberryPi, se nos queda un poco escasa de potencia.

1. Sensores de posición

Hemos hablando mucho de sistemas de sensores de proximidad y distancia de objetos pero nos quedamos cortos en un punto importante, ¿Cual es el sentido de muestra marcha? , ¿En que posición estamos sobre el plano? y lo mas importante de todo ¿Nos estamos moviendo?.

1. Acelerómetros

Parece interesante en este punto hablar de los acelerómetros los cuales nos darán información básica a las tres preguntas que nos hemos planteado y que creemos que a la hora de implementar un sistema real de conducción autónoma es suficientemente importante como para tenerlo encenta.

1. GPS

Otro punto a destacar dentro de posibles mejoras en el sistema es la implementación de un sistema de posicionamiento.

La ley mas básica del desplazamiento es cual es mi origen y cual es mi destino sin saber estas dos variables no es posible implementar un sistema de conducción autónoma verdaderamente eficaz.

1. Mejoras en el sistema de control de motores

Dentro de este punto no hemos podido estudiar todos los sistemas de control disponibles y hemos orientado el proyecto hacia una solución mas sencilla de implementar, pero dentro de este punto existen varias mejoras a tener en cuenta.

1. Control de dirección

Dentro del proyecto hemos implementado un sistema de dirección básica basado en el movimiento denominado como oruga hacer rotar solo los motores de un lado para permitir girar hacia el lado contrario, pero aunque este sistema es eficaz no es lo suficientemente bueno y puede ser mejorado.

Se podría integrar un sistema de dirección controlado por uno o varios servomotores que actuaran sobre las ruedas direccionales que nos permitirá girar de una forma eficaz y controlar de esta forma el ángulo de giro.

1. Control de motores

Hemos implementado el sistema con un Puente H , que nos da ofrece la capacidad de controlar dos motores DC pero en según que situaciones puede que este sistema se quede corto y queramos implementar mas motores.

Además tenemos la problemática del control de potencia por PWM que en el caso de los puente H tiene que ser controlado por software, existen diversas placas controladoras que permiten un control de PWM propio con el cual nos olvidamos del procesamiento de este.

1. Conclusiones

Después de desarrollar todo es proyecto y todo el estudio que ha implicado, vemos muchos campos interesantes de estudio y evolución tanto a nivel profesional como a nivel particular.

A nivel particular se ha tenido una especial motivación en el estudio de los elementos electrónicos puesto que es un campo en el cual siempre he estado interesado particularmente, por otro lado este es simplemente un proyecto de entrada para otros de mucha mayor envergadura puesto que a nivel personal estoy empezando a desarrollar un Dron que me permita reconocer de forma aérea el terreno.

A nivel profesional veo muchas alternativas de evolución en el campo de la visión computerizada, puesto que cada vez existen mas dispositivos con cámaras, móviles sistemas de vigilancia, coches, consolas. Estos sistemas implementan ya métodos de control por sensores de visión y creo pero creo que es posible avanzar muchísimo en este aspecto y combinando camaras con dispositivo de controla cada vez mas pequeños y potentes tenemos un mundo de posibilidades.

Por otro lado me ha permitido profundizar en el conocimiento de elementos de control como raspberryPi, el cual tiene muchas posibilidades, un hecho que me ha permitido evolucionar en mi trabajo por lo conocimientos obtenidos con raspberry pi es un nuevo proyecto de estudio para adaptar RaspberryPi a un sistema de comunicaciones VOIP para completar nuestra gama de productos en la empresa.

1. Referencias y bibliografia

* Especificacion tecnica bus I2C
  + <http://www.semiconductors.philips.com/acrobat_download/literature/9398/39340011.pdf>
* Definición del Bus i2C en
  + Wikipedia [http://es.wikipedia.org/wiki/I%C2%B2C](http://es.wikipedia.org/wiki/I²C)
* DataSheet del controlador I2C PCA9685
  + <https://www.adafruit.com/datasheets/PCA9685.pdf>
* Controlador de Servo motores de Adafruit, especificación y guia de uso
  + <http://www.adafruit.com/products/815>
* Especificaciones Tecnicas Dispositivo de control de motores DC de PiBorg
  + <https://www.piborg.org/picoborgrev/specs>
* Datasheet del chip mcp23017, expansor de puertos i2C
  + <http://ww1.microchip.com/downloads/en/DeviceDoc/21952b.pdf>
* Referencia de programacaión del chip MCP23017 con wiringPi
  + <http://wiringpi.com/extensions/i2c-mcp23008-mcp23017/>
* Descripción de RaspberryPi
  + <http://es.wikipedia.org/wiki/Raspberry_Pi>
* Página oficial de RaspberryPi <http://www.raspberrypi.org/>
* Referencia de la librería de programación Pi4J
  + <http://pi4j.com/>
* Implemantación del modulo de adafruit de control de servoMotores por lediouris
  + http://www.lediouris.net/RaspberryPI/servo/readme.html
* implementación de un cocher RC con raspberryPi
  + http://www.sistemasorp.es/2013/03/23/televigilancia-con-un-coche-rc-arduino-y-la-raspberry-pi/
* calibración estero de camaras con openCV
  + http://www.cnblogs.com/wangyaning/p/4236976.html
* Calibración estero de camaras
  + http://www.jayrambhia.com/blog/stereo-calibration/
* como reconocer objetos con visión artificial
  + http://es.wikipedia.org/wiki/Reconocimiento\_de\_Objetos
* Como realizar un haar training
  + http://www.memememememememe.me/training-haar-cascades/
* Haar Training
  + http://www.trevorsherrard.com/Haar\_training.html
* Como entranar a openCV
  + http://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html
* Opencv haartraining
  + http://note.sonots.com/SciSoftware/haartraining.html
* Como compilar bluecove para raspberry pi
  + https://sites.google.com/site/opengaragedoor1/home/stages/raspberry-pi
* como instalar bluecove en raspberry pi
  + http://lukealderton.co.uk/blog/posts/2015/january/raspberry-pi-bluetooth-using-bluecove-on-raspbian.aspx
* Android rover controller http://www.pezzino.ch/android-rover-controller/
* Calibración de camaras en opncv
  + http://computervisionandjava.blogspot.com.es/2013/10/camera-cailbration.html
* Busqueda de objetos con opencv
  + https://code.google.com/p/find-object/wiki/FindObjectsWithWebcam
* Introducción al desarrollo con opencv y java
  + http://docs.opencv.org/doc/tutorials/introduction/desktop\_java/java\_dev\_intro.html#java-dev-intro
* reconocimiento de señales de trafico
  + http://es.wikipedia.org/wiki/Reconocimiento\_de\_se%C3%B1ales\_de\_tr%C3%A1fico
* Vehiculo rc controlado por raspberrypi
  + http://www.raspberryshop.es/wp/vehiculo-con-raspberry-pi-controlado-distancia-por-bluetooth/
* RC car over web
  + http://blog.kaazing.com/2013/04/01/remote-controlling-a-car-over-the-web-ingredients-smartphone-websocket-and-raspberry-pi/
* Pibot-B
  + http://www.retas.de/thomas/raspberrypi/pibot-b/index.html
* Raspberrypi y los pines implementación de una casa domotica
  + http://www.peatonet.com/raspberry-pi-y-los-pines-gpio-implementando-domotica-asequible-parte-i/
* librería de programación de visión artifical en java
  + https://github.com/sarxos/webcam-capture
* protocolos y java
  + https://unpocodejava.wordpress.com/category/protocolos/
* Raspberry pi y bluetooth
  + http://www.diverteka.com/?p=1880
* Modelo del color
  + HSV http://es.wikipedia.org/wiki/Modelo\_de\_color\_HSV
* Definición de Surf
  + <http://es.wikipedia.org/wiki/SURF>

1. Anexos
2. Anexo 1.- FDS Control Puerto GPIO RaspberryPi

Documento FDS Control Puerto GPIO RaspberryPi.pdf

Especificación funcional del programa de control de android entregado en la practica final de dispositivos moviles

1. Anexo 2.- DataSheet del Chip MCP23017

Documento 21952b.pdf

1. Anexo 3.- DataSheet del Chip Regulador de corriente OKI-78SR

Documento battborg.pdf

Datasheet del regulador de corriente utilizado en el componente battborg usado para alimentar la raspberryPi por medio del GPIO

1. Anexo 4.- DataSheet Sensor HC-SR04

Documento HCSR04-datasheet-version-1.pdf

Datasheet del funcionamiento del sensor HC-SR04

1. Anexo 5.- DataSheet PuenteH L298N

Documento L298N datasheet.pdf

DataSheet del puenteH utilizado para el control de motores en nuestro Coche teledirigido

1. Anexo 6.- Schematic L298N

Documento L298N\_schematic.jpg

1. Anexo 7.- Especificaciones tecnicas chip i2C PCA9685

Documento PCA9685.pdf

1. Dada la cantidad de información sobre este tema es preferible no detallar en exceso este punto. Y hacer referencia a sus estudios en la bibliografia. [↑](#footnote-ref-2)
2. HSV modelo de color en termino de sus componentes [↑](#footnote-ref-3)
3. Surf Speeded-Up Robust Features [↑](#footnote-ref-4)
4. Haar-like características para la detección de oibjetos [↑](#footnote-ref-5)
5. ARM : es una arquitecturas de procesadores RISC de 32 Bits orientada a sistemas de baja potencia y consumo [↑](#footnote-ref-6)
6. JNI:Java Native Interface permite acceder desde java a librerias desarrolladas en otros lenguajes como C/C++ y ensamblador [↑](#footnote-ref-7)
7. CrossCompiling: estilo de compilación que permite generar un binario de una arquitectura desde otra. [↑](#footnote-ref-8)