

Universidad San Jorge

Escuela Politécnica Superior

**GRADUADO EN INGENIERÍA INFORMÁTICA
(ITINERARIO DE ADAPTACIÓN)**

Proyecto Final

Coche teledirigido con RaspberryPi

**Autor del proyecto: Antonio Duce Gimeno
Director del proyecto: David Chinarro Vadillo
Zaragoza, 12 de junio de 2015**



Dedicatoria

“El aprendizaje es el camino y la meta”

Stephen Covey

Gracias a mis padres por facilitarme el camino

Gracias a quienes me acompañaron durante el trayecto

Y gracias a los que me enseñaron a encontrar la meta

Referencias y acrónimos

1. ADAS (Advance driver assistance systems)
2. LIDAR (Ligth Detection And Ranging)
3. RADAR (Radio Detection And Ranging)
4. GPIO (General Purpose Input/Output)
5. PWM (Pulse Width Modulation)
6. GPS (Global Positioning System)
7. HSV (Hue Saturatio Value)
8. RGB (Red Gren Blue)
9. SURF (Speeded Up Robust Features)
10. DARPA (Defense Advanced Research Projects Agency)
11. I2C (Inter-Integrated Circuit)
12. SPI(Serial peripheral interface)
13. ADT (Android Development ToolKit)

Tabla de contenido

Referencias y acrónimos	4
Introducción	13
Antecedentes.....	14
Estado del arte.....	15
Sistemas ADAS	15
Radio Control	19
Drones	20
Percepción del entorno	21
Visión artificial	21
Visión Artificial Estéreo	23
LIDAR	24
RADAR	26
Objetivo	27
Metodología.....	28
Etapas del método con Prototipos	29
Diseño de un prototipo	29
Esquema de proyecto por prototipos	31
Diseño software en espiral.....	33
Requisitos funcionales	35
DISEÑO.....	36
Elementos Fijos.....	36
Elementos Hardware	36
Sistema controlador	36
Elementos de comunicación.....	38
Conexión Wifi	38

Conección Bluetooth	39
Hardware de visión	40
Elementos Software	41
Entornos de desarrollo.....	41
NETBEANS.....	41
Eclipse.....	41
Librerías de desarrollo	42
librería de conexión bluetooth BLUECOVE.....	42
Elementos a estudiar.....	43
Libreras de control de GPIO	43
Pi4J.....	43
OPENJDK Device I/O	44
Librerías de Visión artificial	45
Diseño de la aplicación	46
Visión Artificial.	46
Detección de obstáculos.....	46
Diagrama de proceso	47
Detección de la distancia al objeto	49
Diagrama de proceso	49
Control de movimiento	51
Diagrama de procesamiento	52
Control de comunicaciones básico	54
Diagrama de procesamiento	54
Desarrollo	55
Prototipo 1. Conexión directa motores a bus GPIO	55
Prototipo 2. Extensor de puerto MCP23017	57

Prototipo 3 I2C Adafruit PCA9685.....	60
Prototipo 4 Puente H L298N.....	62
Prototipo 4 A Comunicaciones.....	65
Prototipo 4 B Calculo de distancia	66
Integración del modulo HC-SR04.....	67
Prototipo 4 C Alimentación Raspberry Pi	69
Prototipo 4 E Detección avanzada de objetos	69
Reconocimiento de objetos por color	69
Reconocimiento de objetos por forma	71
Prototipo 4 F Programa de control Android	72
Prototipo 4 G Conducción autónoma.....	74
Diseño final de la aplicación.....	76
Programa control RaspberryPi	76
Programa Control Android.....	94
Análisis Coste del proyecto	100
Coste material.....	100
Coste Diseño	102
Análisis visual	103
Explotación de los costes	104
Reflexiones	105
Sensor de distancia	105
Visión electrónica	105
Ángulos muertos cámaras.....	105
Detección de objetos genéricos.....	106
Visión 3d	106
Sensores de posición.....	106

Acelerómetros.....	106
GPS.....	107
Control de dirección	107
Control de motores	107
Conclusiones.....	108
Referencias y bibliografía	109
Anexos	112
Anexo 0 Propuesta proyecto	112
Anexo 1. Compilar librería BlueCove en RaspberryPi	113
Anexo 2. Instalación y uso de librería Pi4J	114
Anexo 3. Compilación OpenJDK DIO.....	116
Anexo 4. Compilación de OpenCV en RaspberryPi	117
Anexo 5. Explicación simplificada del Bus i2C	118
Anexo 5. Calculo de distancia con trigonometría.....	120
Anexo 6. Calculo de distancia con Geometría de Estereoscópica	121
Anexo 7. Sensor de distancia HC-SR04	122
Anexo 8 BattBorg	125
Anexo 9. Clasificadores Haar	127
Anexo 10 DataSheet L298N	129
Anexo 11. Contenido del CD	133

Ilustración 1Google self-driving	14
Ilustración 2Control de crucero a adaptativo	15
Ilustración 3Navegador de a bordo.....	15
Ilustración 4Asistencia cambio carril	16
Ilustración 5Advertencia abandono de carril.....	16
Ilustración 6Sistema anticolisión.....	16
Ilustración 7Adaptación velocidad inteligente	16
Ilustración 8Reconocimiento de señales.....	17
Ilustración 9Ayuda de estacionamiento.....	17
Ilustración 10Control luces adaptativo	17
Ilustración 11Visión nocturna	17
Ilustración 12Modelo borde ideal.....	22
Ilustración 13Sistema visión estéreo.....	23
Ilustración 14Tipos de LIDAR.....	24
Ilustración 15Calculo posición por triangulación	25
Ilustración 16Red NavStar posicionamiento GPS.....	25
Ilustración 17Diagrama diseño de prototipo	30
Ilustración 18Proyecto por prototipos	33
Ilustración 19Diseño software por prototipos	34
Ilustración 20Diseño Software en espiral	34
Ilustración 21Placa RaspberryPi, modelo B+	36
Ilustración 22Asus USB-N10.....	38
Ilustración 23CSL – Adaptador Bluetooth USB nano V4.0	39
Ilustración 24Logitech C170	40
Ilustración 25Ejemplo detección obstáculos	47
Ilustración 26DFD detección obstáculos básica.....	47

Ilustración 27DFD detección distancia	50
Ilustración 28Esquema movimiento tipo oruga	51
Ilustración 29DFD Control giro	52
Ilustración 30DFD Comunicaciones básicas	54
Ilustración 31Prototipo1 conexión directa GPIO	55
Ilustración 32Controlador MCP23017	57
Ilustración 33Esquema Controlador MCP23017.....	57
Ilustración 34Placa de prototipo MCP23017	58
Ilustración 35Prototipo2 Controlador MCP23017	58
Ilustración 3616-channel, 12-bit PWM Fm+ I2C-bus LED controller	60
Ilustración 37PCA9685.....	60
Ilustración 38Conexionado de Placa controladora a RaspberryPi.....	61
Ilustración 39Placa controladora con ServoMotor y rueda	61
Ilustración 40Detalle pines l298N	62
Ilustración 41Controlador L298N	62
Ilustración 42Detalle conexionado motores con ruedas	63
Ilustración 43Detalle Conexionado Motores a L298N.....	63
Ilustración 44Detalle conexionado L298N a raspberryPi	63
Ilustración 45Sensor HC-SR04 (www.modmypi.com)	66
Ilustración 46Conexión de Hc-sr04 a raspberryPi	67
Ilustración 47Detalle conexión hc-sr04 en protoboard.....	67
Ilustración 48Circuito Divisor de corriente final.....	67
Ilustración 49Esquema circuito divisor corriente	67
Ilustración 50Detección canales por canal RGB	70
Ilustración 51REsultado detección objeto por canal RGB	70
Ilustración 52Filtro HSV sobre imagen	70

Ilustración 53Detección objetos con Surf	71
Ilustración 54Ejemplo de objeto detectado con harrcascade	71
Ilustración 55Detalle pantalla principal aplicación android	73
Ilustración 56Detección de dos objetos.....	74
Ilustración 57Detección de un único objeto.....	74
Ilustración 58DFD Control de dirección rediseñado	75
Ilustración 59Extracto del convenio del metal aplicado a informática	102
Ilustración 60Distribución coste por sección	103
Ilustración 61Distribución Coste Origen	103
Ilustración 62Puerto GPIO de raspberryPi B+	114
Ilustración 63Diagrama de puertos lógicos de Pi4J	115
Ilustración 64Representación Bus i2C	118
Ilustración 65Esquema trigonometría	120
Ilustración 66Geometria Epipolar.....	121
Ilustración 67Vista inferior Battborg	125
Ilustración 68vista superior BattBorg	125

Introducción

La Real Academia de las Ciencias Físicas y Exactas define la automática como el conjunto de métodos y procedimientos para la substitución del operario en tareas físicas y mentales previamente programadas. De esta definición original se desprende la definición de la automatización como la aplicación de la automática al control de multitud de elementos físicos llevados a cabo desde un ordenador. Existe una gran variedad de proyectos que recurren a la automática, desde casas demóticas, dispositivos robóticos, procesos industriales y vehículos autónomos. Éstos últimos son coches que pueden conducir sin necesidad de un conductor, al menos se les dota de capacidades tecnológicas de la ingeniería de control e inteligencia artificial para facilitar o imitar las acciones humanas de manejo y control del vehículo, por lo tanto un vehículo autónomo es capaz de percibir todos los eventos del medio que le rodea que inciden en la ruta optimizada mediante diversos sensores y técnicas de navegación y actuar en consecuencia.

Los centros de I+D que tratan de incorporarse a este campo emergente con productos innovadores, necesitan prototipos simplificados, desarrollados en sucesivas fases y con funciones simplificadas, para llegar a la consolidación y verificación de un sistema de control más complejo. Además, los centros de formación demandan unos sistemas entrenadores despiezados al más bajo nivel de componentes para que el alumno tenga más versatilidad de acoplos y mayor la posibilidad de expresar su facultades innovadoras. Este proyecto es una experiencia de diseño, construcción y experimentación como posible guía de estudio de una iniciativa empresarial o contribución didáctica.

Antecedentes

Últimamente se está hablando mucho de proyecto de las técnicas avanzadas de conducción con objetivo de la conducción autónoma total. Sin embargo, 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 presentó un coche eléctrico que conducía gracias a un circuito integrado en el pavimento.

En 1980 DARPA (DefenseAdvancedResearchProjects Agency) hizo que un vehículo condujese más 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 más de 1000 kilómetros con tráfico intenso a una velocidad de 130km/h solo con pequeñas intervenciones humanas.

En 1995 El equipo Dickmans aplicó un sistema de visión computerizada a su coche y permitió realizar un viaje Munich- Copenague ida y vuelta a más 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 ganador de DARPA Grand challenge 2005 (Stanley) coche autónomo creado por el equipo de SebastianThruningerniero de Google y Director de Stanford Artificial IntelligenceLaboratory.



Ilustración 1 Google self-driving

(www.androidheadlines.com)

El dotar de inteligencia a los automóviles es una línea 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 en algunas de sus funciones 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.

Estado del arte

Sistemas ADAS

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 vehículos inteligentes de forma comercial está 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 vehículo, ayudando al conductor a hacer la conducción más cómoda y segura.

Entre los sistemas ADAS existentes encontramos las siguientes:

Sistema de navegación a bordo (In-vehiclenavigationsystem). Con GPS y TMC para ofrecer posicionamiento en carretera e información del tráfico

Control de crucero adaptativo (Adaptativecruise control). Adapta automáticamente la velocidad del vehículo para mantener una distancia seguridad con los vehículos que le anteceden.

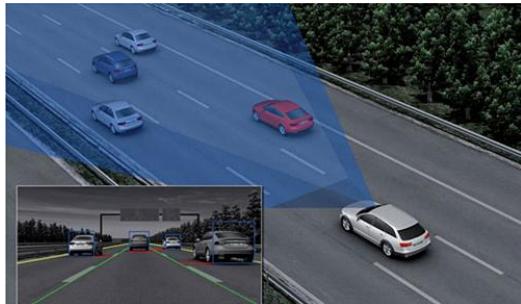


Ilustración 2 *Control de crucero a adaptativo*
(www.opel.com)



Ilustración 3 *Navegador de a bordo*
(www.opel.com)

Asistencia de cambio de carril (Lanechangeassistance). Son sistemas que cubren elángulo muerto de los vehículos avisando de la presencia de otros en sus inmediaciones.

Sistema de advertencia de abandono de carril (Lanedeparturewarningsystem). advierte al conductor cuando el vehículo comienza a salirse de su carril en autopistas y carreteras principales.

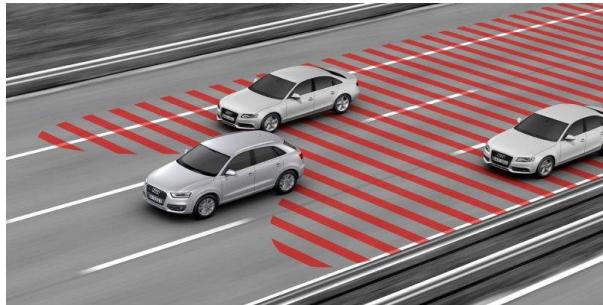


Ilustración 4Asistencia cambio carril
(www.opel.com)

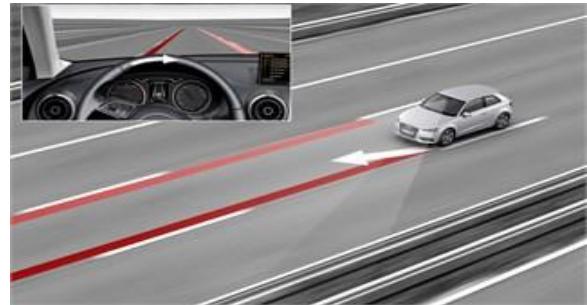


Ilustración 5Advertencia abandono de carril
(www.opel.com)

Sistema anticolisión (Collisionavoidancesystem). 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 (Intelligentspeedadaptation). 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.

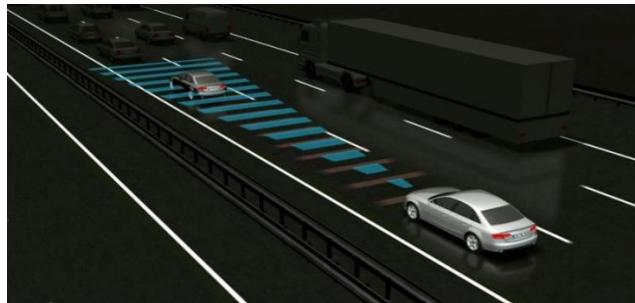


Ilustración 6Sistema anticolisión
(www.opel.com)



Ilustración 7Adaptación velocidad inteligente
(www.opel.com)

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
(www.opel.com)*



*Ilustración 9 Ayuda de estacionamiento
(www.opel.com)*

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 (Automotivenight 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 luces adaptativo
(www.opel.com)*



*Ilustración 11 Visión nocturna
(www.opel.com)*

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 drowsinessdetection). Dispone de una cámara que monitoriza la cara del conductor y detecta cuando éste está somnoliento.

Sistemas de comunicación Vehicular (Vehicular communicationsystems). 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

más protagonismo en la conducción en busca de la meta de la conducción segura y autónoma que irá 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 está en vías de desarrollo una solución única y cada modelo está 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.

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.

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 ParrotAR.Drone, que aunque no fue ni mucho menos el primero, si que lo popularizo este tipo de vehículos.

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 más a fondo a continuación

Visión artificial

Sin duda esta es una de las técnicas más importantes dentro de la automatización de un vehículo, podríamos 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 podríamos destacar la siguientes:

Detección y localización de objetos (caras , coches..).

Registro y evaluación de resultados

Seguimiento de objetos entre distintas imágenes.

Mapeo de un escenario para conseguir un modelo tridimensional , permitirá moverse por el escenario capturado.

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

Uno de los aspectos más 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 más avanzados como la detección de bordes.

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

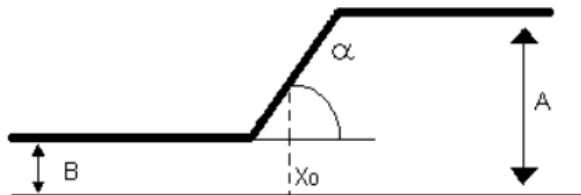


Ilustración 12 Modelo borde ideal

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 algoritmo 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 menor que el umbral bajo es descartado, si el pixel está entre los dos umbrales solo se acepta si está en contacto con un pixel de umbral alto.

Transformadas de Hough. Con este cálculo es posible detectar todo tipo de figuras que sea posible expresarlas mediante una fórmula matemática, deben su nombre al su inventor (Paul Hough), para este cálculo se usa una dimensión igual al número de parámetros desconocidos del problema.

Revisar Anexo 5. Calculo de distancia con trigonometría para más información sobre el cálculo de distancias mediante trigonometría básica con un cámara

Visión Artificial Estéreo

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

Una definición más 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.

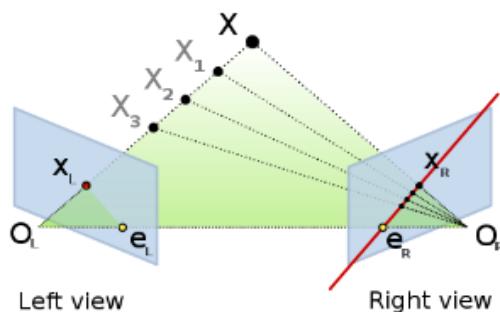


Ilustración 13Sistema visión estéreo

Estos datos podemos obtenerlos mediante una serie de cálculos automatizados , usando los algoritmos de Zhang¹ , los cuales mediante un proceso de hacer varias fotografías a un cuadro de ajedrez en distintas posiciones espaciales nos permite obtener estos datos.

Revisar Anexo 6. Calculo de distancia con Geometría de Estereoscópica para obtener más información sobre como calcular distancias mediante visión estereoscópica.

¹Dada la cantidad de información sobre este tema es preferible no detallar en exceso este punto. Y hacer referencia a sus estudios en la bibliografía.

LIDAR

Es el acrónimo 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 más 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.

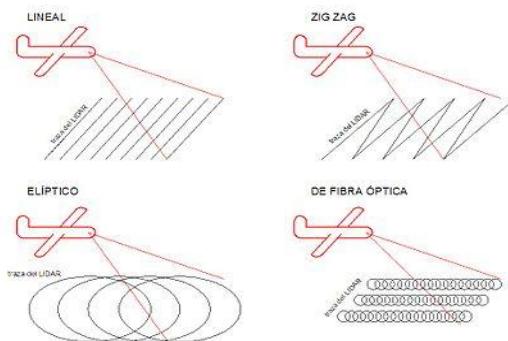


Ilustración 14 Tipos de LIDAR

(es.wiki.com)

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.

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.

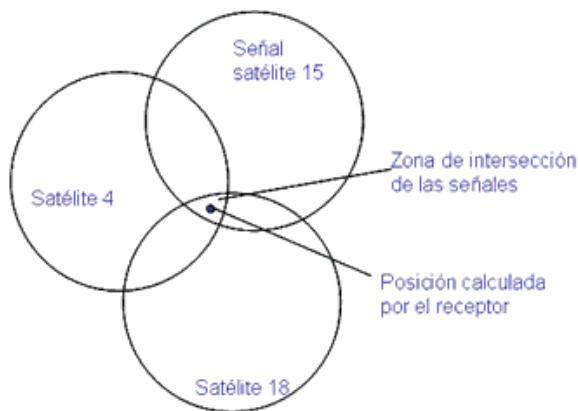
Fibra óptica: En es caso el sistema monta múltiples espejos más pequeños con múltiples haces lo que nos da mayor resolución pero un ángulo menor de escaneado .

Elíptico: en este caso el haz es desviado por dos espejos produciendo un escaneado elíptico 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.

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 órbita que ubican el objeto gracias a la trilateración, dándonos la ubicación en el globo con una precisión de metros.



*Ilustración 15 Calculo posición por triangulación
(es.wiki.com)*



*Ilustración 16 Red NavStar posicionamiento GPS
(es.wiki.com)*

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

El mensaje que nos mandan (efeméride) 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. .

RADAR

acrónimo de Radio detection and ranging, es un sistema que usa ondas 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 ondas electromagnéticas de distintas frecuencias nos puede aportar distintos tipos de información.

Objetivo

El objetivo de este Proyecto Fin de Grado es el diseño, desarrollo y construcción de un prototipo robot que experimente algunas de las técnicas avanzadas de ayuda a la conducción como las basadas en Visión por Computador; así como estudiar la capacidad para controlar elementos físicos desde dispositivos móviles próximos, o desde computadores remotos por medio de aplicaciones.

Especificamente, el trabajo se centra en lo siguiente :

- Construir un coche miniaturizado cuyo sistema de control está basado en un dispositivo (RaspberryPi) que nos permitirá interactuar con el medio físico mediante los puertos GPIO.
- Estudiar API existentes para desarrollar funciones de visión por computar que emulen la localización, seguimiento y análisis de la visión del conductor bajo condiciones simplificadas.
- Construir algoritmos eficaces y fiables que sean capaces de procesar la información inferidas de las imágenes en tiempo real.
- Desarrollar aplicaciones móviles en el entorno local para recepción de alertas y órdenes de control.
- Desarrollar aplicaciones servidoras web que utilicen como pasarela el dispositivo móvil en las proximidades del robot para, de forma remota, recibir alertas y emitir órdenes de control.

El alcance del proyecto está determinado por el prototipo robot, focalizando el desarrollo tanto de la ingeniería mecánica y electrónica de control como de la informática. Se trata de estudiar la capacidad de controlar los movimientos de un coche teledirigido, dotado de diversos sensores (Video, Proximidad) que le permita tener un conocimiento del entorno en el que circula, procesar las imágenes y extraer la información necesaria de control. Además se estudia distintas formas de comunicación con el dispositivo para transmitir órdenes y recibir información.

Metodología

La metodología está basada en Desarrollo en Espiral de Prototipos de producto, donde se integra el diseño mecánico y electrónico con el software de control y tratamiento de la información

En la metodología por prototipos se busca probar varias suposiciones formuladas por analistas y usuarios con respecto a las características requeridas del sistema, con la finalidad de aclarar los requerimientos de los usuarios o verificar la factibilidad del diseño del sistema. Por lo tanto, el prototipo debe ser un producto o aplicación que debe funcionar sin errores, aunque con limitaciones definidas y focalizado en las funciones esenciales del problema planteado.

Los prototipos se crean con rapidez, evolucionan a través de un proceso interactivo y tienen un bajo costo de desarrollo ya que se crean consecutivamente y pueden desecharse en caso de estar fuera de requerimientos

Esta metodología está justificada en este proyecto por:

- No conocer suficientemente los requerimientos que determinan el objetivo a alcanzar.
- La dificultad en evaluar los requerimientos.
- Los costos altos de inversión que suponen el riesgo de implementar y estudiar sobre un producto final.
- Se evitan altos riesgos que suponen experimentar con el vehículo autónomo
- Es una oportunidad para ensayar nuevas tecnologías que optimicen la función y uso del producto

Etapas del método con Prototipos

Dividimos el diagrama en dos partes diferenciadas, una primera referente a la creación de un prototipo específico y los pasos que seguimos desde su definición hasta su verificación y posteriormente una segunda que englobaría todo el proyecto y es donde se revisa la viabilidad del prototipo.

Diseño de un prototipo

Presentamos el modelo a seguir utilizado en el diseño de prototipos individuales.

Una primera fase de este proceso es identificar las funciones que queremos conseguir y las pruebas que tendremos que seguir para verificar si el prototipo es válido o no.

De este proceso tendremos que extraer varias conclusiones cuando desechemos un prototipo, las cuales se tendrán que documentar o por lo menos indicar el motivo de rechazo y validación.

Un prototipo puede ser rechazado por diversas causas, ya sean por problemas de hardware, de software o simplemente porque el prototipo no se adapta perfectamente a nuestras necesidades.

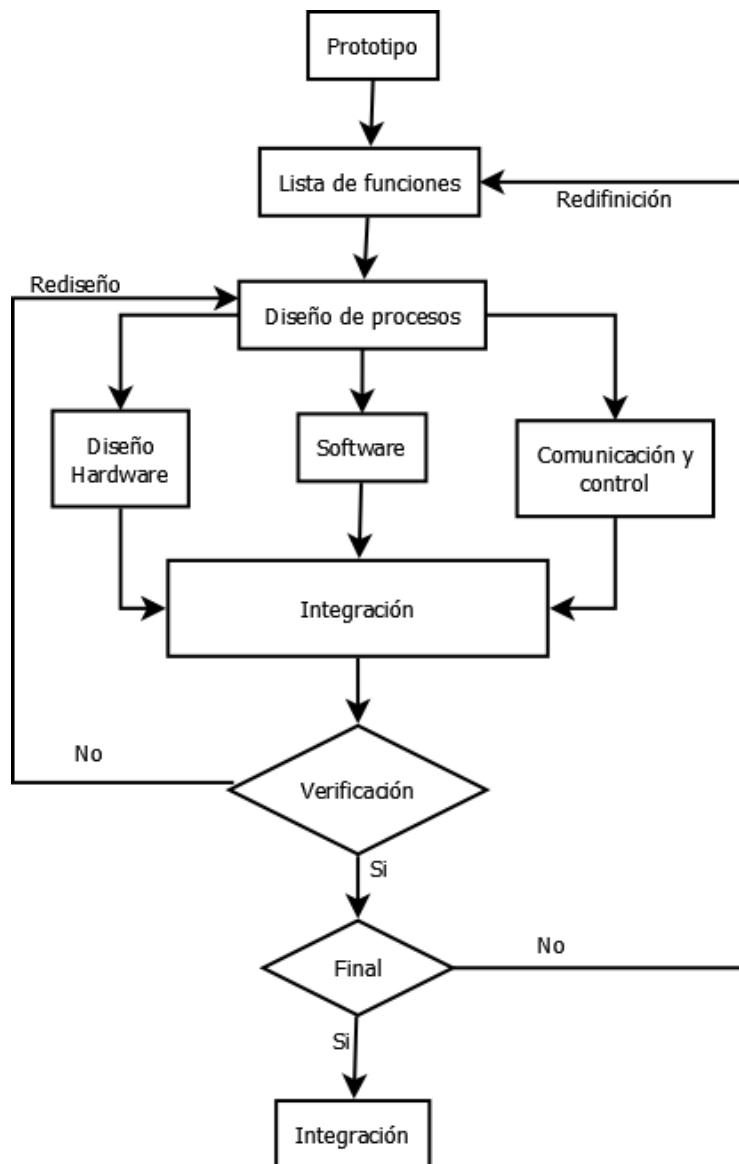


Ilustración 17 Diagrama diseño de prototipo

Esquema de proyecto por prototipos

Anteriormente hemos mostrado como creamos un prototipo y cuáles son las fases por las que tenemos que pasar, pero este esquema se queda incompleto a la hora de trabajar con un proyecto puesto que solo se centra en los prototipos.

A continuación presentamos un pequeña mejora sobre este proceso dentro del cual integraremos las fases de creación de prototipos.

Como podemos observar unas primera fase de este proceso, como en todos los proyectos, es la identificación de los objetivos que queremos conseguir. Estos objetivos nos llevan directamente a la especificación de una serie de requisitos que queremos cumplir y a la identificación de si existe alguna tecnología que cumpla con nuestros objetivos o la tendremos que crear.

En este punto empezaremos a trabajar en nuestros prototipos hasta encontrar uno que cumpla con todas nuestras necesidades o al menos que sea capaz de cumplir con todos nuestros requisitos funcionales, los cuales hemos definido en una fase anterior.

Después de encontrar un prototipo y que este haya pasado todas las verificaciones continuaremos con el proyecto, dándole forma y perfilando todos los detalles para ultimarlo.

Evidentemente no existe forma de garantizar ni el tiempo ni el dinero que se puede invertir en un diseño por prototipos, si que está claro que posiblemente se mucho más económico que enfrentarnos directamente al desarrollo de un producto final sin conocer exactamente toda la tecnología, capacidades y requerimientos.

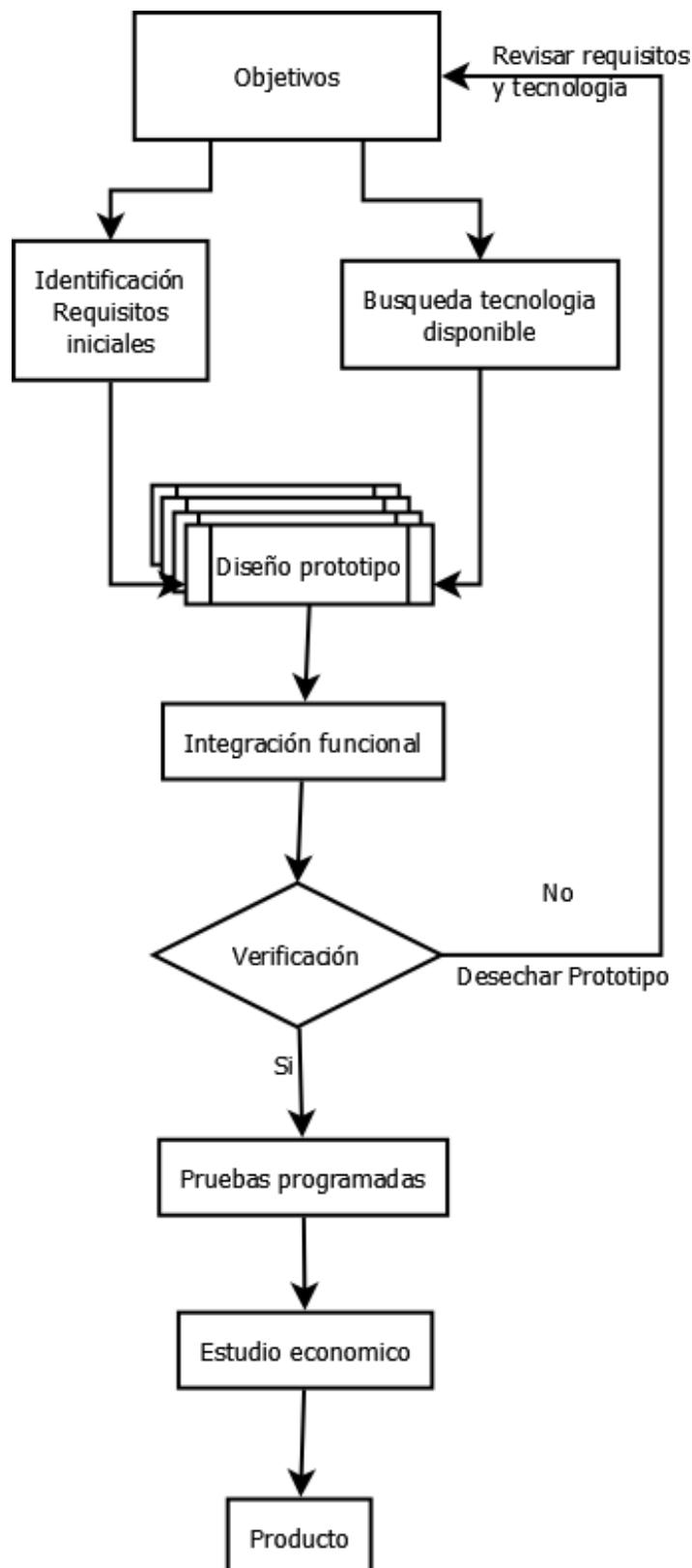


Ilustración 18 Proyecto por prototipos

Diseño software en espiral

Dentro del campo de diseño de software ,trabajaremos en un diseño espiral el cual nos permitirá realizar pequeños avances funcionales sobre el prototipo final seleccionado.

Aunque en la fase de prototipos realizaremos bastante desarrollo de software este se utilizara en una fase final en la cual desarrollaremos todos el sistema.

Podemos identificar dos diagramas clásicos del desarrollo de software por prototipos y desarrollo de software en un modelo espiral nos valdrán tanto para una primera fase de desarrollo de prototipos como para la etapa final donde iremos evolucionando poco a poco para integrar pequeños avances.

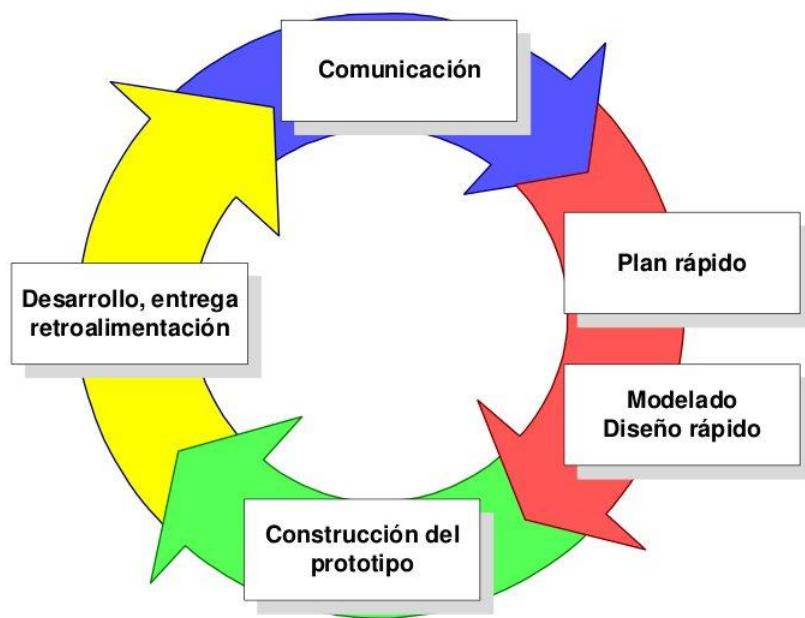


Ilustración 19Diseño software por prototipos

(<http://www.codejobs.biz>)

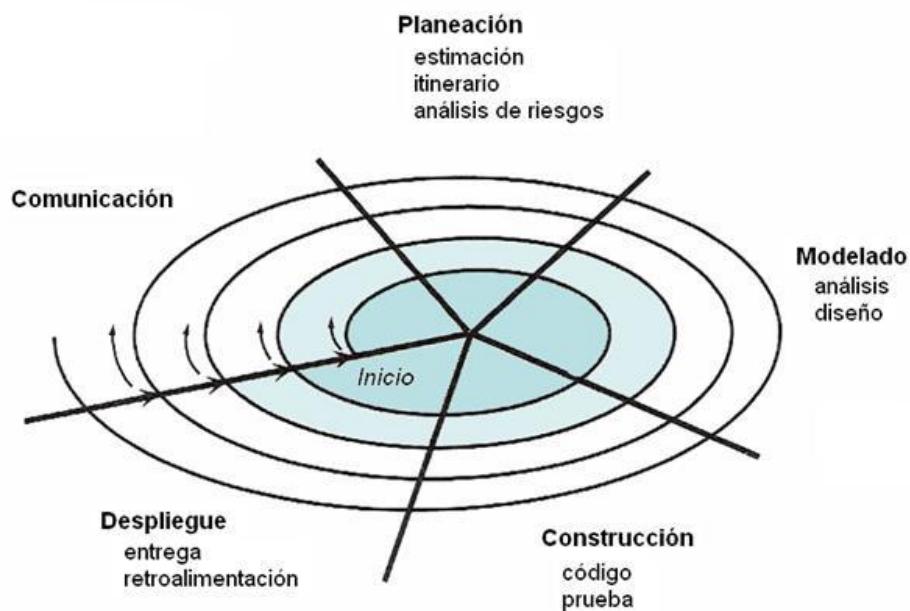


Ilustración 20Diseño Software en espiral

(<http://www.codejobs.biz>)

Requisitos funcionales

Construir un dispositivo que tenga capacidad de movimiento.

- 1) El dispositivo creado tiene que ser capaz de obtener información del medio que le rodea
- 2) Construir un sistema multi-plataforma que nos permita comunicarnos con el dispositivo
- 3) El sistema tiene que ser capaz de comunicarse con el dispositivo para obtener y mandar información
- 4) El sistema tiene que ser capaz de establecer una conexión con el dispositivo por medio de una red PAN
- 5) El sistema tiene que ser capaz de establecer una conexión con el dispositivo por medio de una red LAN
 - i. Requisito optativo, El sistema tiene que ser capaz de establecer la conexión con el dispositivo por medio de una red WAN.
- 6) El sistema tiene que ser capaz de controlar los movimientos del dispositivo creado
- 7) El dispositivo tiene que ser capaz de mandar información del medio que le rodea la sistema controlador.
- 8) El dispositivo tiene que ser capaz de detectar obstáculos
- 9) El dispositivo tiene que ser capaz de calcular trayectorias para esquivar obstáculos.

DISEÑO

Dentro de la fase de diseño tenemos una serie de elementos que no podemos variar y otros sobre los cuales realizaremos un estudio de viabilidad de la tecnología para comprobar si esta se adapta a nuestras necesidades.

A continuación detallamos tanto los elemento fijo como las tecnologías que vamos a estudiar y sobre las cuales realizaremos nuestros prototipos.

Pasamos a detallar aquellos elementos que son inamovibles o directamente no tiene un relevancia funcional específica por tratarse de componentes comunes basados en estándares del mercado y sobre los cuales tendremos que desarrollar nuestros prototipos

Elementos Fijos

Elementos Hardware

En este punto definiremos los elementos Hardware sobre los que desarrollaremos nuestra tecnología

Sistema controlador

Todo nuestro sistema va a estar basado sobre una raspberryPi, existen diversos modelos de esta placa de desarrollo pero nos centraremos en los dos más populares hasta el momento.

Este elementos es el principal de sistema, encargado de controlar todos los dispositivos y donde residirá la lógica del proyecto los modelos B/B+ puesto que nos ofrece una serie de características que consideramos necesarios y son los más sencillos de conseguir en este momento.

Pasamos a describir rápidamente este tipo de dispositivo.



Ilustración 21 Placa RaspberryPi, modelo B+
(www.modmypi.com)

RaspberryPI : 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.

Es un elemento ideal para el prototipado puesto que ofrece una serie de características avanzadas para la creación de pequeños proyectos de robótica además de una potencia razonable y un precio muy ajustado.

Este dispositivo nos ofrece las siguientes características:

- 4 puertos USB 2.0
- Procesador de ARM11 1176JZF-S a 700 MHz overclockable 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 600ma
- SO basado en Linux
- Puertos GPIO (General Purpose Input/Output) de Entrada Salida Digital
- Capacidad PWM (pulse-width modulation) sobre varios de sus puertos
- Interfaz Bus I2C (Inter-Integrated Circuit).
- Interfaz Bus SPI (Serial Peripheral Interface).
- Interfaz UART(Universal Asynchronous Receiver-Transmitter).

Otro punto muy importante sobre la elección de este dispositivo es la gran cantidad de información existente en la red acerca de este elemento , además de la gran variedad de proyectos ya existente que pueden servirnos de ayuda o guía en la fase de prototipado.

Elementos de comunicación

Sobre este tipo e elementos poco tenemos que hablar puesto que la mayoría de ellos se basa en estándares del mercado.

Los elementos que usemos deben de cumplir una serie de características mínimas que detallamos a continuación.

Conexión Wifi

Este punto está relacionado con el requisito funcional 5) sobre la conexión a redes LAN

Es un mecanismo de conexión de dispositivos electrónicos de forma inalámbrica las características principales que buscamos son las siguientes:

- Conexión mínima B/G
- Seguridad inalámbrica basada en WPA
- Conexión USB 2.0
- Tamaño reducido a ser posible nano receptor

El modelo que hemos seleccionado ha sido el siguiente

Este modelo cuenta con la siguientes características:



*Ilustración 22Asus
USB-N10*

(PcComponentes.com)

- Estándar de red IEEE 802.11 b/g/n
- Interfaz USB 2.0
- Frecuencia de funcionamiento 2.4 GHz
- Canal de funcionamiento 11 para N. América, 13 Europa (ETSI)
- Segmento de producto N150 Acceso instantáneo fácil; 150Mbps
- Modulación 64QAM, 16QAM, QPSK, BPSK, CCK, DQPSK, DBPSK, OFDM
- Seguridad 64-bit WEP, 128-bit WEP, WPA2-PSK, WPA-PSK
- Certificados CE, FCC, C-Tick, IC, NCC
- Peso 2 g (sólo el dispositivo)

Conexión Bluetooth

Este punto está relacionado con el requisito funcional 4) sobre conexión a redes PAN.

Bluetooth es una especificación industrial para redes inalámbricas de área personal que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia.

Las principales características que buscamos en este dispositivo son las siguientes:

- Bluetooth V2.0 o superior
- Conectividad serie
- Bajo consumo
- Tamaño reducido, a ser posible nano receptor
- Interfaz USB 2.0

El modelo seleccionado ha sido el siguiente

Este modelo cuenta con las siguientes características



- Miniadaptador USB Bluetooth® 4.0
- Interfaces: USB 2.0 (compatible con la versión anterior USB 1.1) / Bluetooth® 4.0 (compatible con versiones anteriores)
- Gama de frecuencias: 2,402 – 2,480 GHz
- Altas velocidades de transmisión de hasta 3 MBit/s
- Establecimiento rápido de la conexión e instalación sencilla
- Se comunica sin problemas con todos los equipos Bluetooth
- Consumo mínimo de corriente

*Ilustración 23CSL –
Adaptador Bluetooth USB
nano V4.0*

(www.compraschulas.net)

Hardware de visión

Este punto está relacionado con el requisito funcional 7) y 8) referentes a la percepción del entorno.

Otro de los elementos importantes del proyecto es la cámara que usaremos para reconocer el entorno, puesto que existe multitud de webcams y todas cumplen con una serie de características mínimas deseadas, al igual que en los anteriores casos buscaremos cualquier elemento económico que cumpla con una serie de características principales marcadas.

- Conexión Usb 2.0
- Captura de jpeg
- Distancia Focal fija
- Resolución de captura mínima de 320X240
- Procesado de imagen por hardware.

El elemento que hemos seleccionado en este caso es el siguiente



*Ilustración 24Logitech
C170*

(www.logitech.com)

Las características principales de este dispositivo son las siguientes:

- Conexión USB 2.0
- UVC Support Yes
- Tipo lentes y sensor: Plástico
- Tipo Enfoque fijo
- Resolución 640x480
- Ángulo de captura (FOV) 58°
- Distancia Focal 2.3 mm

Elementos Software

Dentro de los elementos Software fijos existen una serie de elementos los cuales no podemos variar ya sea por restricciones del entorno en el que estamos basando nuestro proyecto o por propias restricciones que nos imponemos a la hora de desarrollar, en este caso se ha decidido desarrollar todo el sistema en Java, aunque existen posibilidades de desarrollo en otro lenguajes de programación como Python se ha desecharido este lenguaje principalmente por la falta de conocimiento avanzado como para desarrollar todo el proyecto en el.

Entornos de desarrollo

Para el desarrollo de este proyecto hemos seleccionado dos entornos de programación distintos, aunque basados en el mismo lenguaje de desarrollo, en nuestro caso JAVA, cada entorno esta centrado y tiene una serie de características apreciables para determinadas tareas.

NETBEANS

Netbeans es un entorno de desarrollo integrado libre, principalmente orientado para el lenguaje de programación java, dispone de un gran número de módulos que permiten extender su funcionalidad de forma sencilla.

Hemos seleccionado este entorno para el desarrollo del programa de control que instalaremos en nuestra RaspberryPi , puesto que tiene una integración nativa con el API de desarrollo grafico JavaSwing que nos permitirá crear una interfaz grafica sencilla con la que podremos interactuar.

Eclipse

Se trata de un programa, al igual que el anterior , de un IDE(Entorno de desarrollo integrado) que nos ofrece múltiples opciones para desarrollar programas en distintas plataformas, en este caso nos interesa la capacidad de integrarse con ADT (Android Development Toolkit) entorno para el desarrollo de aplicaciones android.

Hemos seleccionado este entorno debido a esta peculiaridad esencial puesto que nuestro sistema de control remoto estará basado en esta plataforma y por lo tanto debemos tener una herramienta para desarrollarlo.

Librerías de desarrollo

Existe una serie de librerías de desarrollo que debido a restricciones del sistema no podemos variar puesto que son la única alternativa válida existente en el mercado actualmente para abordar una serie de tareas que tenemos que desarrollar.

librería de conexión bluetooth BLUECOVE

Es una implementación de JSR-82-J2SE multiplataforma que nos permite el manejo y control de dispositivos bluetooth desde java.

Inicialmente esta librería está preparada para funcionar sobre sistemas operativos basados en x86 y por lo tanto para poder usarla dentro de nuestro sistema android cuyo sistema operativo está basado en ARM es necesario seguir una serie de pasos para adaptar la librería a nuestro entorno.

A la hora de preparar nuestro entorno será necesario recompilar esta librería para que funcione en ARM por este motivo nos veremos obligados a compilar dentro de la raspberryPi para que los binarios generados sean compatibles con esta última.

Ver Anexo 1. Compilar librería BlueCove en RaspberryPi para obtener información detallada de cómo compilar la librería en RaspberryPi.

Elementos a estudiar

En este punto estudiaremos las distintas tecnologías que tenemos a nuestra disposición para alcanzar cada una de los requisitos funcionales del proyecto.

Librerías de control de GPIO

GPIO (General purpose Input/Output) Es un pin genérico en un chip cuyo comportamiento, incluyendo si es un pin de entrada o de salida, se puede controlar por el usuario en tiempo de ejecución.

Dado que Raspberry dispone de un puerto GPIO de 40 pines y que estos serán necesarios para poder interactuar con el medio físico mediante motores, sensores, leds..., necesitaremos una librería para el control de dicho puerto.

Llegados a este punto descubrimos que tenemos dos alternativas que pasamos a relatar a continuación.

Pi4J

PI4J es una librería de desarrollo que provee de un interfaz orientado a objetos para el control de I/O y una implementación de las librerías para desarrollar en java que dan acceso a todas las capacidades de control del puerto GPIO de la plataforma RaspberryPi.

Inicialmente basada en la librería de desarrollo WiringPi provee una interfaz basada en java pero igual de potente.

Esta librería posee las siguientes características:

- Exportar y liberar GPIO pins
- Configurar la dirección de los puerto GPIO (Entrada y/o Salida)
- Auto detectar la dirección de los pines GPIO
- Controlar/escribir los estado de los pines GPIO
- Leer los pines GPIO
- Usar pwm por software y hardware.
- Escuchas en base a interrupciones de los pines GPIO
- Establece estados predeterminados a la salida del programa.
- Eventos basados en los estados de los pines
- Enviar y recibir información vía RS232.
- Comunicación I2C.
- Comunicación SPI
- Acceso a la información del sistema de RaspberryPi
- Wrapper para acceso a WiringPi.

Aunque se trata de un librería muy completa y muy útil tiene la problemática que en según qué situación puede ser excesivamente lenta, para situaciones en las que no se requiera una gran precisión, es la alternativa a elegir.

Otro problema que tenemos que tener en cuenta ante esta librería es que se trata de un librería de programación específicamente desarrolladle para controlar una raspberryPi y por lo tanto no es compatible con otras plataformas.

OPENJDK Device I/O

Esta es otra de las alternativas que tenemos disponibles para el control del puerto GPIO de raspberryPi.

Es un API genérica para el acceso a los pines GPIO de diversos dispositivos integrados, no está orientado específicamente a ningún dispositivo sino que se orienta a un sistema multiplataforma en sistemas basados en JavaSE.

Tiene las siguientes características:

- Acceso a pines GPIO en entrada salida.
- Conexión al Bus I2C
- Conexión al puerto RS232
- Conexión SPI

Esta librería al contrario de la anterior y al tratarse de una librería genérica, no dispone de un interfaz tan amigable de desarrollo como Pi4J pero a su favor tiene que es mucho más rápida para tareas que requieran de una gran precisión.

No se han encontrado muchas referencias a esta librería salvo pequeño ejemplos desarrollados por la comunidad, con el fin de demostrar su potencia.

El acceso a los puertos al contrario que en Pi4J sigue la nomenclatura con la que han sido denominados y por lo tanto no es necesaria ninguna tabla para traducir su numeración.

Librerías de Visión artificial

Dentro de este campo existen multitud de librerías para el control de la visión computarizada pero nos centraremos en la librería más importante y la más usada en este campo, se trata de OpenCV.

OpenCV es un librería libre de visión artificial originalmente desarrollada por Intel, se ha usado en infinidad de aplicaciones, desde sistemas de seguridad hasta aplicativos de control de calidad.

Esta librería es multiplataforma existiendo versiones para casi todos los sistemas operativos del mercado.

OpenCV nos ofrece un librería de desarrollo basada en C/C++ optimizada para aprovechar al máximo las capacidades del sistema, esto nos plantea un problema, si la librería esta en C++ como podremos acceder desde Java.

Para resolver este problema tenemos que recurrir a un interfaz JNI, actualmente existen dos alternativas o usamos el interfaz JNI integrado desde hace relativamente poco dentro de la misma librería OpenCV o hacemos uso de un interfaz de desarrollo externo en este caso JAVACV el cual no ofrece una serie de librerías para el acceso a OpenCV.

En este caso hemos seleccionado directamente el interfaz de java que viene con la librería de desarrollo openCV puesto que es considerablemente más rápido y estable que javaCV, además solo será necesario distribuir una única librería.

Diseño de la aplicación

A lo largo de este capítulo vamos a describir las funciones de software que vamos a implementar sin entrar en detalles de la tecnología utilizada, describiremos desde un punto de vista analítico como deberíamos ir implementando el software y los distintos avances funcionales para encarar posteriormente el desarrollo de una forma más sencilla.

Visión Artificial.

Unos de los puntos más 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.

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 el análisis del proyecto y continuar desde este punto hacia objetivos más complejos.

Para ello definiremos que todos los obstáculos que vamos a encontrar tendrán un círculo 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 más adelante será el cálculo de la distancia hasta dicho obstáculo o la detección de diversos tipos de obstáculos.



Ilustración 25 Ejemplo detección obstáculos

Diagrama de proceso

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.

Inicialmente partiremos del siguiente diagrama :

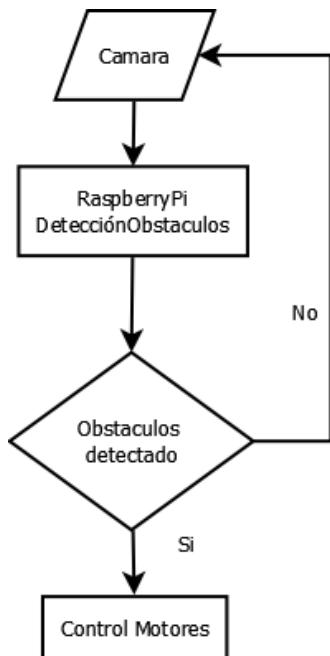


Ilustración 26 DFD detección obstáculos básica

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

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

Como podemos observar en la ilustración 30 por medio de la cámara capturamos una imagen, buscamos posibles círculos que se encuentren en la imagen y los marcamos dibujando el círculo sobre esta.

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

A continuación mostramos una pequeña función de cómo deberíamos diseñar el sistema.

```
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 sería 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 pues mejora sustancialmente nuestra tasa de detección.

Detección de la distancia al objeto

Como en el punto anterior definimos un nuevo avance funcional con el que iremos completando el desarrollo de nuestro sistema.

En este caso se plantea el siguiente problema, hemos detectado un obstáculo pero ¿a qué distancia esta?, ¿Tengo que actuar ya?.

Para resolver estas dudas tendremos que conocer la distancia al obstáculo detectado, no entramos en este momento en detalles de cómo calcularemos dicha distancia pues, existen diversas técnicas para hacerlo, por esta razón no quedamos un punto más arriba y solo planteamos como tendremos que actuar ante dicha situación.

Diagrama de proceso

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.

Como punto más importante de todo este proceso es saber a qué distancia estamos del obstáculo, definiremos un límite de parada que será configurable en puntos posteriores.

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

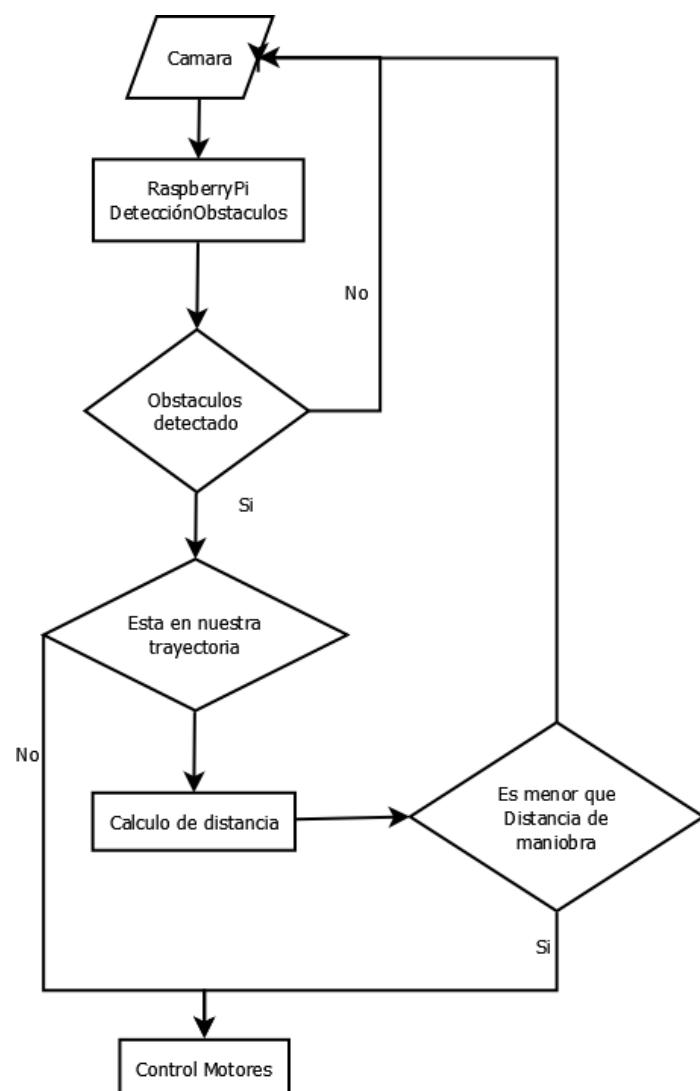


Ilustración 27DFD detección distancia

Control de movimiento

En este punto detallaremos como actuaremos en base al movimiento que queremos generar como en otros ejemplos vamos a definir el proceso desde un punto de vista funcional que más tarde abordaremos de forma más específica entrando en detalles en los diversos prototipos que hagamos.

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 las 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.

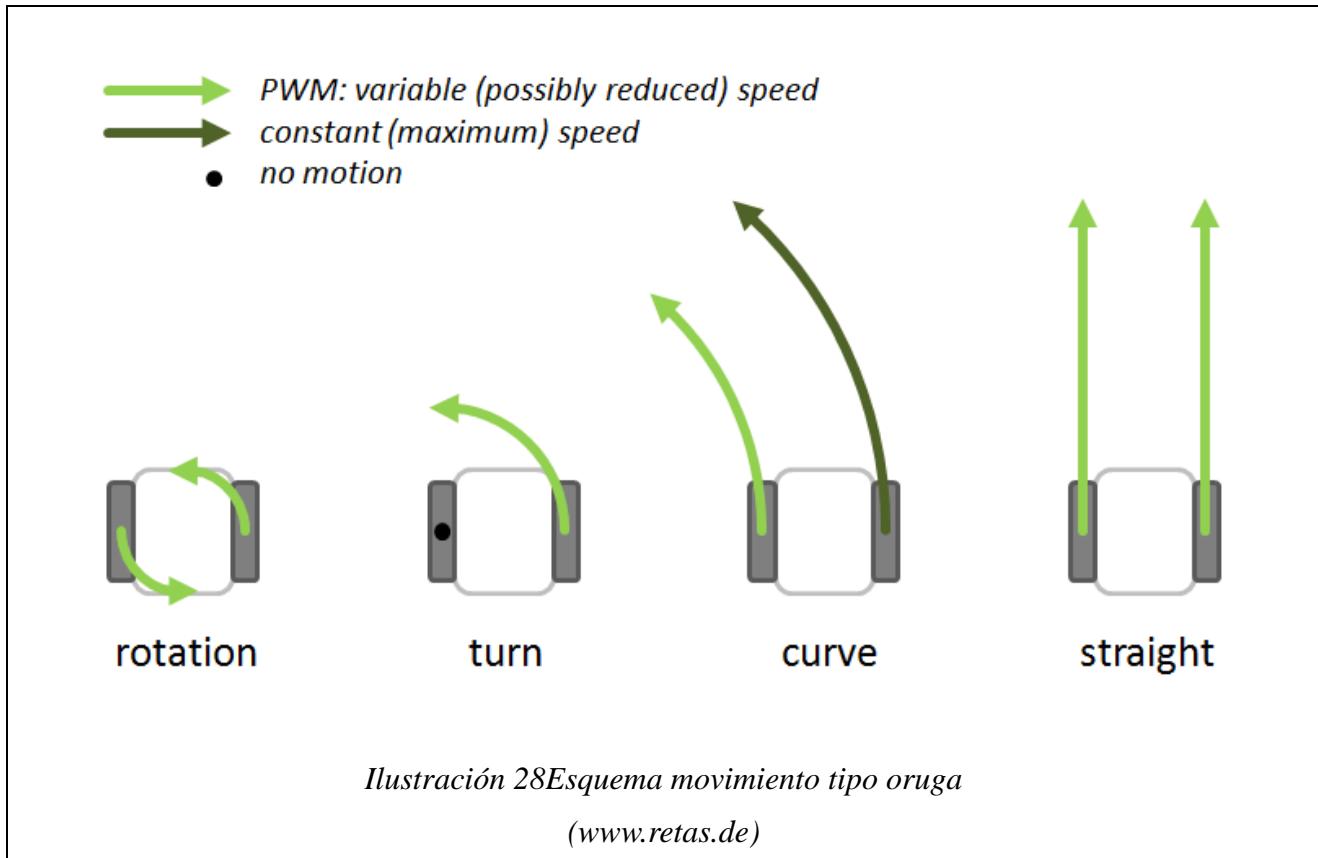
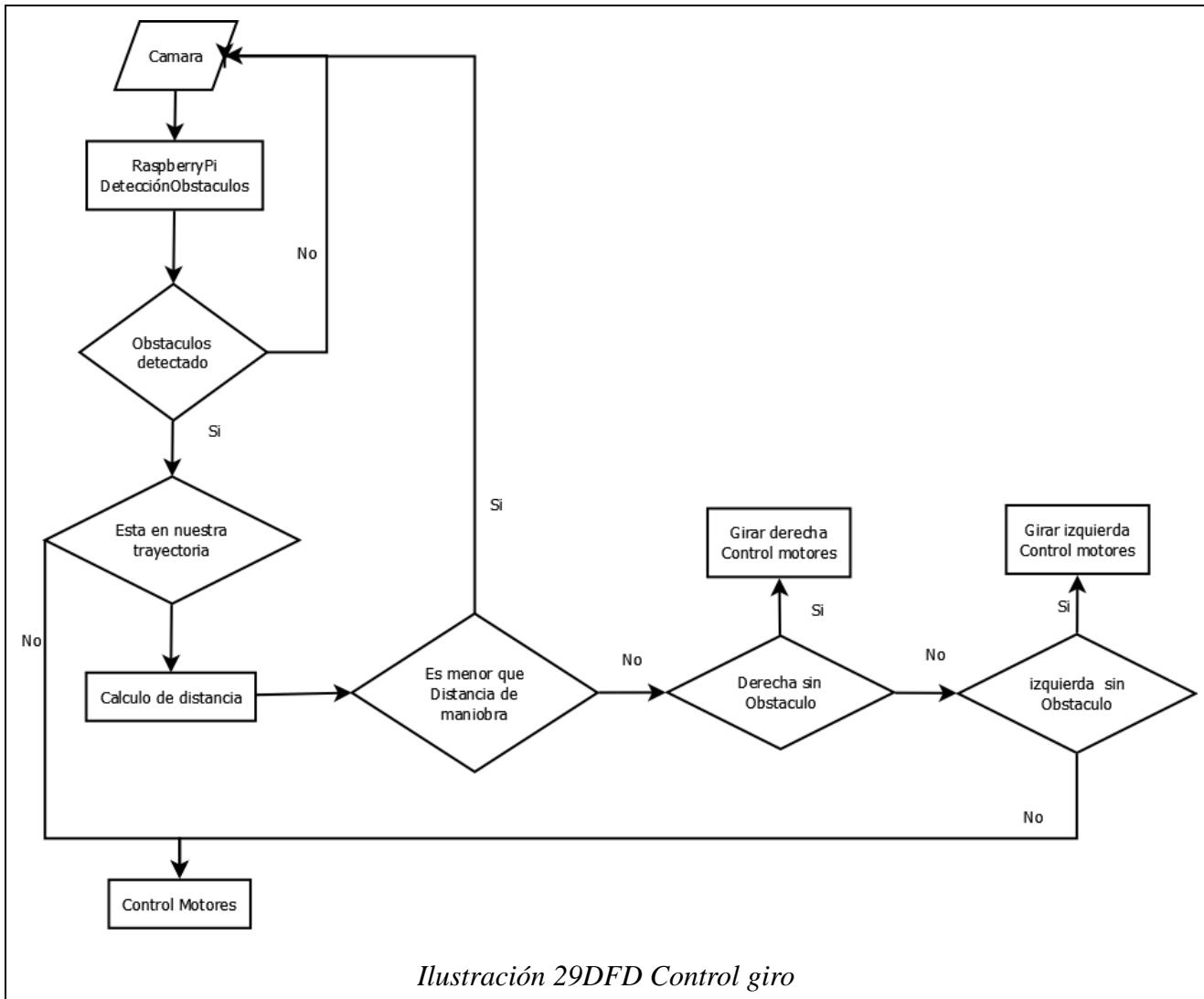


Diagrama de procesamiento

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

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 una 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.



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 una 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.

Control de comunicaciones básico

En este punto nos salimos un poco del proceso actual he implementamos un modulo distinto en este caso comunicaciones este modulo nos permitirá establecer un comunicación activa entre los elementos controlados y los elementos controladores.

En esta fase simplemente implementaremos un modelo básico de comunicación basado en socket y controlaremos el estado de este así como los datos que nos envían desde el elemento controlador.

Diagrama de procesamiento

Como podemos observar en el diagrama planteamos un sistema en el que o estamos conectados o estamos a la espera de conexión, de esta misma forma controlamos si tenemos datos en la entrada o en la salida y actuaremos en consecuencia con el socket.

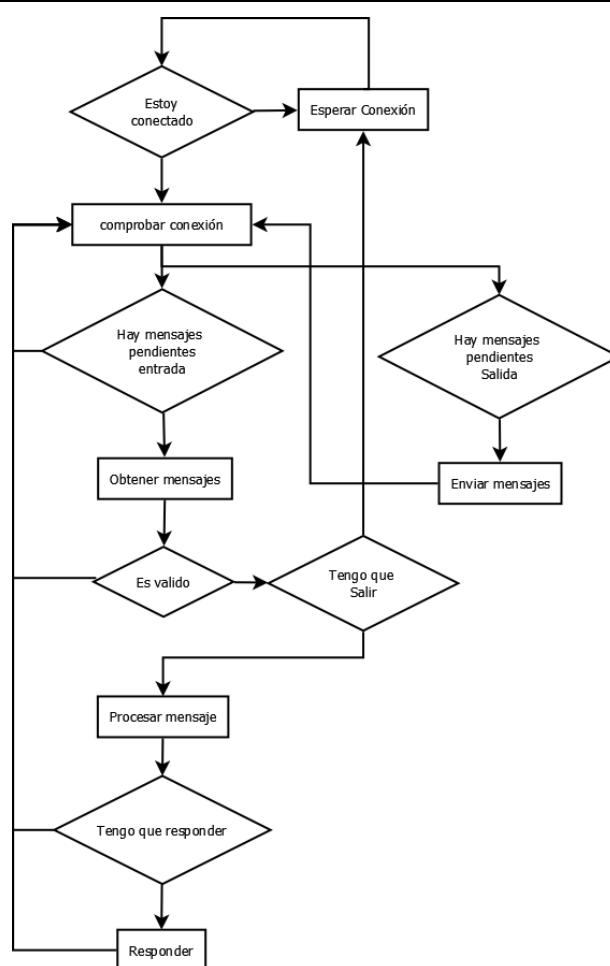


Ilustración 30 DFD Comunicaciones básicas

Desarrollo

Dentro de esta fase estudiaremos las distintas alternativas de enfocar los problemas mediante el desarrollo de varios prototipos para comprobar si se ajustan a nuestras necesidades.

Enfocamos los prototipos sobre todo a nivel de control de movimiento puesto que es en esta categoría donde más alternativas encontramos y donde nos es más complicado decidir una solución válida.

Prototipo 1. Conexión directa motores a bus GPIO

El primer prototipo que desarrollamos lo hacemos en base a la conexión directa de los motores DC al puerto GPIO.



Ilustración 31 Prototipo 1 conexión directa GPIO

Se desarrolla un programa de control mínimo que usando la librería de desarrollo Pi4J que permita controlar los motores simplemente activando los pines a los que están conectados los motores.

Para facilitar las conexiones utilizamos un extensor de puerto GPIO que nos permite conectar de forma sencilla distintos componentes sin necesidad de soldar.

Nos damos cuenta en seguida que este prototipo no es viable por los siguientes motivos:

- Únicamente podemos controlar un sentido de la marcha
- Si realizamos la conexión de forma que permita marcha adelante y marca atrás la raspberry entra en cortocircuito y se resetea.
- En determinadas circunstancias los motores exigen demasiada potencia y la raspberry al no poder asumirla se sobrecalienta y se apaga.

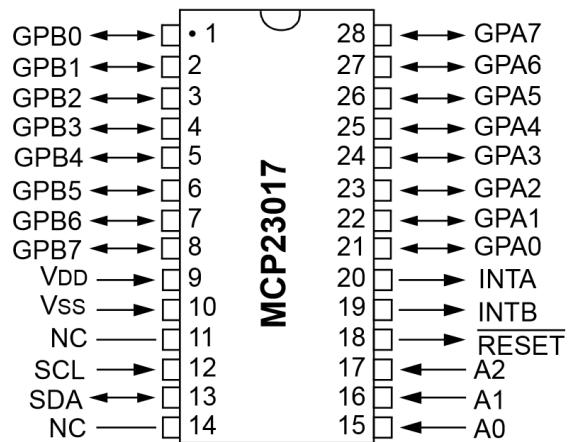
Debido a los resultados infructuosos decidimos descartar el prototipo por las razones presentadas.

Prototipo 2. Extensor de puerto MCP23017

Este componente es un extensor de puerto I2C, realizamos un pequeño estudio sobre esta tecnología para saber afrontar los problemas y entender un poco más como funciona.



*Ilustración 32 Controlador MCP23017
(www.adafruit.com)*



*Ilustración 33 Esquema Controlador MCP23017
(www.adafruit.com)*

Para más información acerca del bus I2C revisar Anexo 5. Explicación simplificada del Bus i2C

Dentro de este prototipo tendremos que crear un pequeño circuito electrónico que nos permitirá conectar de forma sencilla los distintos componentes al controlador MCP 23017.

Este Circuito lo hemos creado sobre una base de prototipado , donde hemos soldado una alimentación a 5v y hemos extendido las conexiones de los puertos a un sistema de pines similar a la que dispone RaspberryPi.

En este caso hemos implementado una alimentación externa que nos permitirá suministrar la potencia requerida por los motores,

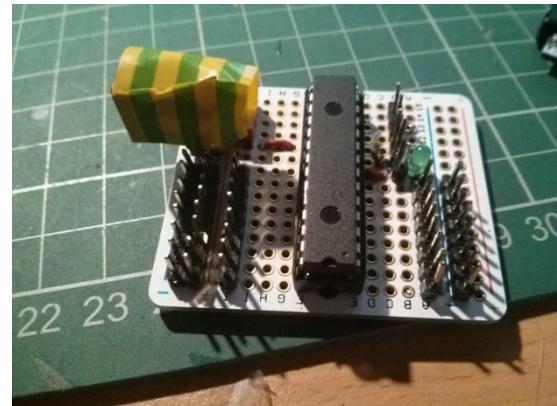
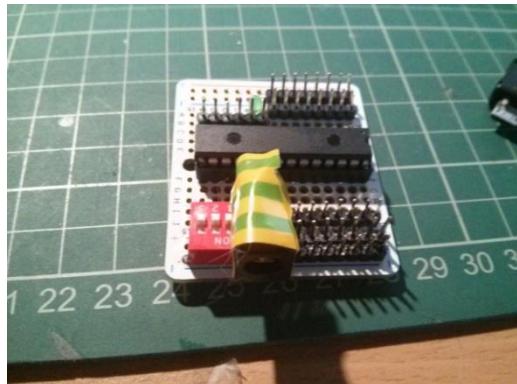


Ilustración 34 Placa de prototipo MCP23017

Para el control de los motores mediante este prototipo se ha tenido que desarrollar una clase que Permita transformar comandos I2C a la activación de los puertos que dispone el controlador.

Desarrollamos un pequeño programa de pruebas que nos permite testear el desarrollo y la viabilidad de este.

Creamos la clase CGestorI2CAdafruit que permite controlar puertos individuales por medio de bus I2C.

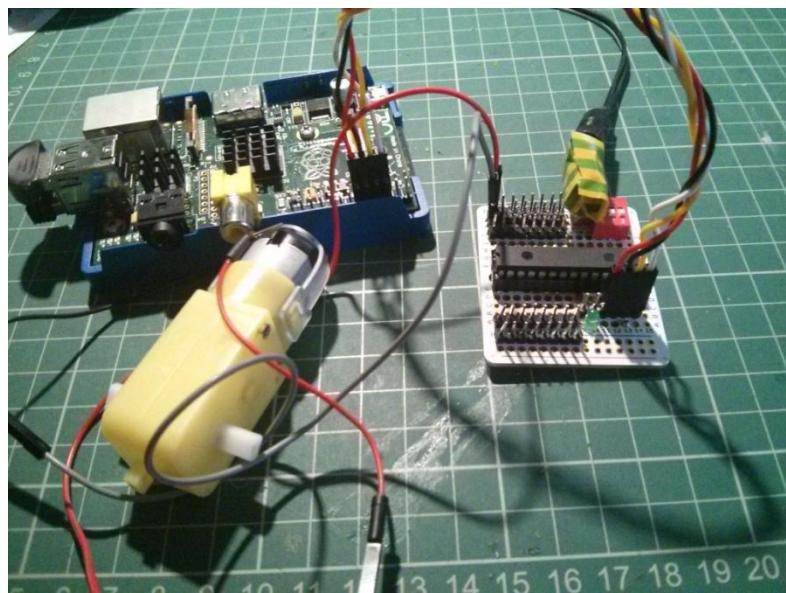


Ilustración 35 Prototipo 2 Controlador MCP23017

Tenemos que desechar este prototipo por las siguientes razones:

- No es posible el control de marcha adelante y marcha atrás en un solo motor.
- Es necesario usara al menos 4 motores para el control de movimiento.
- Complica demasiado la lógica del programa controlador.
- Al no tener regulador de corriente los motores funcionan siempre a 5v.
- No es posible hacer uso de PWM, los motores siempre funcionan la misma velocidad.

Debido a los malos resultados decidimos rechazar el prototipo y seguir estudiando nuevas posibilidades.

Prototipo 3 I2C Adafruit PCA9685

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

Para más información acerca del bus I2C revisar Anexo 5. Explicación simplificada del Bus i2C

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

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

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

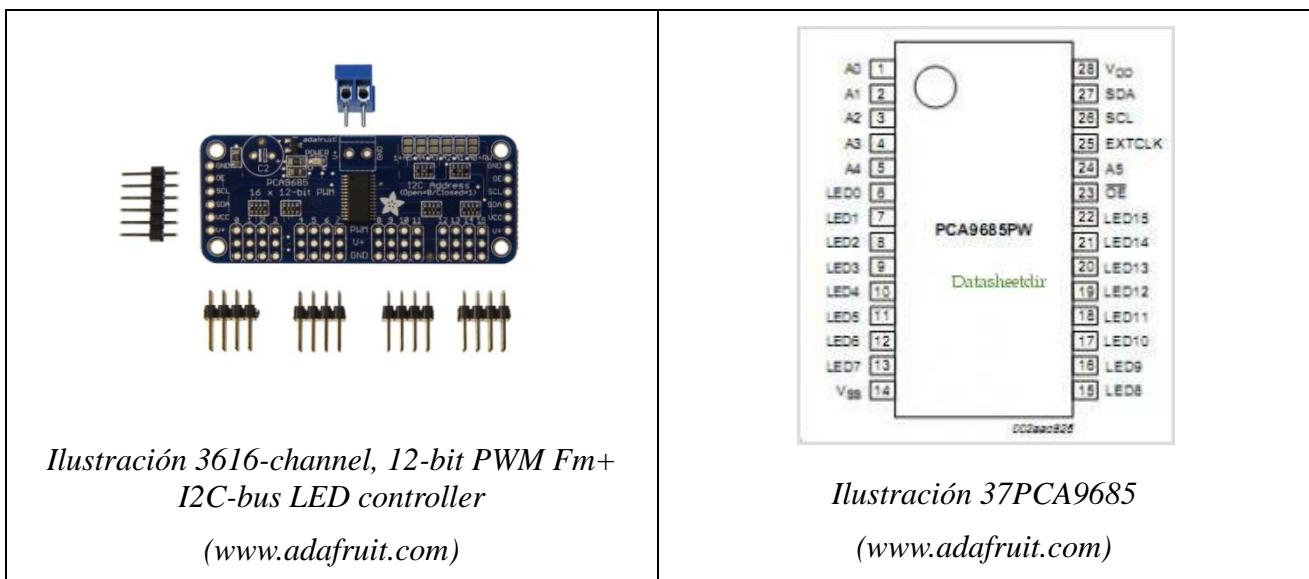


Ilustración 36 16-channel, 12-bit PWM Fm+
I2C-bus LED controller
(www.adafruit.com)

Ilustración 37 PCA9685
(www.adafruit.com)

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 una 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 configurar 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 automática escribiendo en una única posición de memoria.

Este prototipo al igual que el anterior es controlado por i2C , por esta razón modificamos ligeramente la clase creada anteriormente lo que nos permitirá controlar este nuevo dispositivo.

En este caso al dejar de trabajar con motores DC y comenzar a trabajar con motores Servo hemos creado una pequeña clase que nos permitirá gestionar estos últimos de manera sencilla ServoTest.



Ilustración 38Conexionado de Placa controladora a RaspberryPi

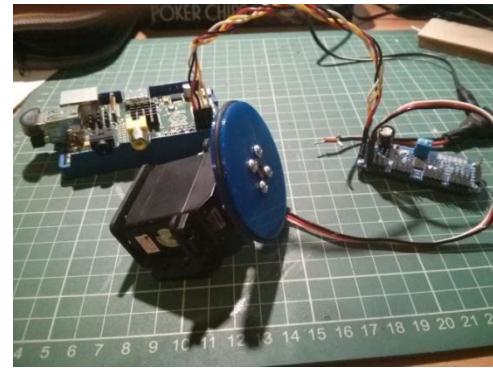


Ilustración 39Placa controladora con ServoMotor y rueda

Como puntos negativos de anteriores prototipos destacamos la dificultad tanto nivel hardware como a nivel software de ejecutar movimientos con un solo motor adelante y atrás, en este caso podemos comprobar como con este prototipo tenemos la posibilidad de ejecutar movimientos hacia adelante y hacia atrás de forma sencilla.

También tenemos el beneficio del control de potencia por PWM que ofrece directamente este dispositivo.

El dispositivo además nos ofrece la posibilidad de una alimentación externa variable de 3V a 6V lo cual nos facilita enormemente el control de motores sin recurrir a la alimentación directa desde el bus GPIO que en anteriores prototipos nos ha dado problemas.

Aunque la mayoría de los aspectos han sido positivos en este prototipo nos hemos encontrado ante una problemática a la hora de obtener otros elementos necesarios para finalizar el proyecto.

En este caso hemos encontrado un problema con la ruedas, en la imagen anterior vemos como se ha adaptado una rueda impresa en 3D al servomotor, hemos podido comprobar como esta no es capaz de generar la suficiente tracción como para mover con relativa soltura el robot comprobando como están llegan a derrapar.

Se ha buscado alternativas a ruedas y no hemos encontrado ninguna que fuera capaz de adaptarse correctamente al servomotor por un precio módico.

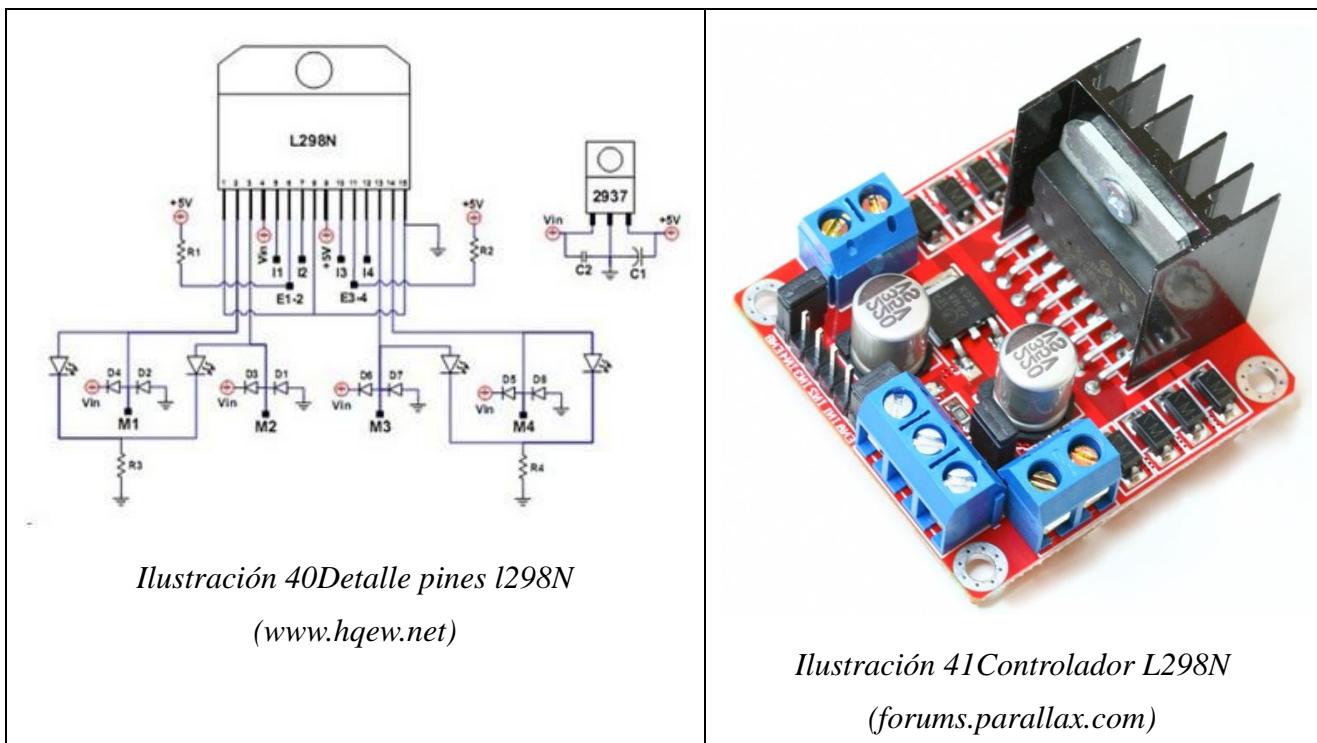
Decidimos rechazar este prototipo por las causas evidentes detalladas anteriormente.

Prototipo 4 Puente H L298N

Dados los anteriores fracasos decidimos reorientar de nuevo el proyecto a la utilización de motores DC que tiene mayores alternativas de repuestos.

En este caso vamos a utilizar un controlador puente H L298N. Este elemento nos permite controlar dos motores DC o un motor de pasos, ejecutando instrucciones desde el bus GPIO de RaspberryPi.

Este elemento es un componente muy económico, 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.



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 está 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 pérdida de rendimiento que puede afectar a todo el sistema.
- Sobrecalentamiento : Este tipo de unidades es conocida por su calentamiento extremo y consecuentemente su pérdida 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

En el caso de este prototipo hemos tenido que crear diversas clases que nos permitan controlar este elemento así como los elementos que le conectemos, para eso creamos las siguientes clases disponible en el código fuente entregado:

- CMotorDC (clase controladora de motores DC).
- CMotorControlPuenteH (Clase controladora del puente H)
- L298NTest(clase de test que nos permitirá controlar los movimientos de los motores desde una sencilla interfaz de usuario).

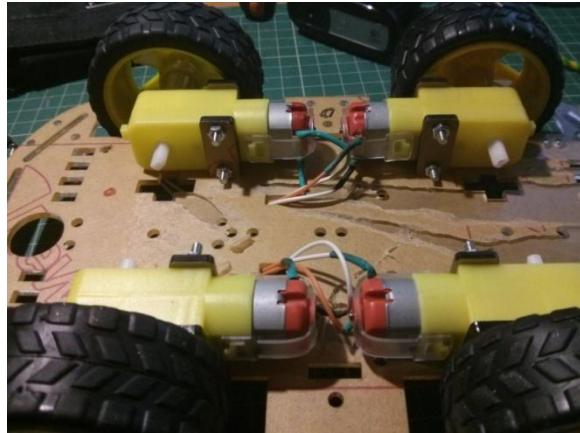


Ilustración 42Detalle conexiónado motores con ruedas



Ilustración 43Detalle Conexiónado Motores a L298N



Ilustración 44Detalle conexiónado L298N a raspberryPi

En este caso Hemos podido comprobar cómo al igual que en el anterior hemos podido controlar la dirección de la rotación sin problemas tanto adelante como hacia atrás.

Aunque este elemento solo nos ofrece la conexión a dos motores se ha creado un puente directo entre estos los cual nos permite controlar 4 motores de forma sencilla en parejas de dos.

El prototipo ha sido capaz de arrancar el movimiento sin que sus ruedas patinen , causa por la que desechamos el anterior prototipo.

Aunque hemos podido comprobar que efectivamente el regulador de potencia se calienta en exceso cuando alimenta los motores y la RaspberryPi este hecho no es de mayor importancia puesto que buscaremos alternativas viables más adelante para darle corriente.

Para obtener más información sobre este controlador ver Anexo 10 DataSheet L298N

Prototipo 4 A Comunicaciones

En una fase anterior de diseño "

Control de comunicaciones básico” consideramos un estilo de comunicación básico, en este momento redefinimos este proceso para darle una funcionalidad avanzada.

Tenemos que tener en cuenta que en la sección requisitos funcionales definimos como requisito funcional que la aplicación fuera capaz de conectarse a redes PAN, LAN y como requisito Optativo a redes WAN.

En este punto nos encontramos con un problema tenemos que ser capaces de controlar varias entradas de datos y que todas ellas se sincronicen desde un único punto, es por esta razón que rediseñamos el diagrama anterior para completarlo y darle un enfoque distinto a la funcionalidad.

Para este fin implementamos un sistema que permite controlar varios socket los cuales va encuestando, con el fin de comprobar la conexión.

Por otro lado se implementa un lista de mensajes general, en realidad es una instancia singleton que permite que todos los socket o elementos de comunicación trabajen con la misma lista de esta forma podremos alternar comandos utilizando cualquiera de los socket o desde dentro del sistema con la comunicación autonómica.

Prototipo 4 B Calculo de distancia

Uno de los puntos que dejamos abierto en el diseño es el cálculo de la distancia donde simplemente indicábamos que tendríamos que buscar un medio para calcular la distancia al obstáculo de forma precisa y detallada.

Este punto hace referencia al requisito funcional 8)El dispositivo tiene que ser capaz de detectar obstáculos hacemos referencia a los anexos (Anexo 5. Calculo de distancia con trigonometría y Anexo 6. Calculo de distancia con Geometría de Estereoscópica) donde se explica el modo de calcular distancias mediante imágenes obtenidas con las cámaras conectadas al sistema.

El cálculo de distancias mediante imágenes tuvo un resultado infructuoso por distintos motivos que pasamos a enumerar a continuación:

- El cálculo de distancia mediante trigonometría resultó ser poco fiable dando resultados demasiado aleatorios como para ponerlos en producción
- El cálculo de distancia por trigonometría era únicamente aplicable a obstáculos más pequeños que el robot y que estuvieran localizados a una distancia corta.
- El vehículo era incapaz de maniobrar en la distancia detectada.
- El cálculo de distancia con Geometría epipolar (Visión estéreo) requería una gran cantidad de procesamiento, que no podría ser soportado por la raspberry Pi.
- Requería imágenes de alta resolución para obtener valores aceptables.
- No permitía el cálculo de distancia a un único punto o a un rango de la imagen.
- Requería un entrenamiento realizar un entrenamiento bastante extenso para que los resultados fueran óptimos

Dado que las soluciones probadas para el cálculo de distancia desde las imágenes obtenidas con la cámara no han sido validas buscamos otras alternativas adoptando elementos externos.

Tras varias búsquedas no decidimos por el siguiente componente ver anexo Anexo 7. Sensor de distancia HC-SR04, Este modulo permite medir la distancia mediante ultrasonidos, es económico y sencillo de usar.

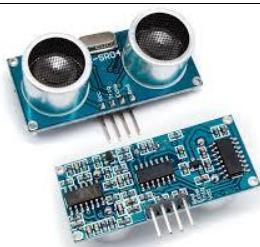


Ilustración 45Sensor HC-SR04 (www.modmypi.com)

Integración del modulo HC-SR04

Para poder controlar este modulo con raspberryPi es necesario realizar un pequeño circuito eléctrico, un divisor de corriente mediante dos resistencias.

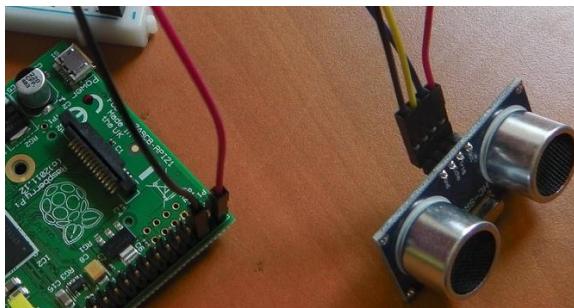


Ilustración 46 Conexión de Hc-sr04 a raspberryPi

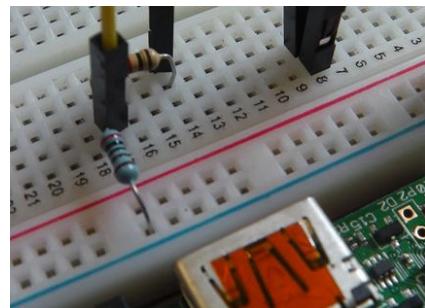


Ilustración 47 Detalle conexión hc-sr04 en protoboard

En las imágenes anteriores podemos observar cómo se realiza el conexionado del sensor a nuestra raspberryPi, pero esto solo nos vale a un nivel de test puesto que evidentemente no vamos a montar los prototipos mediante una protoboard, por este motivo creamos un circuito con el que integraremos nuestro prototipo de robot.

El cálculo de la magnitud de las resistencias siguen el siguiente patrón $V_{sal} = V_{ent} \cdot \frac{R_2}{R_1+R_2} \Rightarrow \frac{V_{sal}}{V_{ent}} \cdot \frac{R_2}{R_1+R_2} = \frac{3,3v}{5v}$.

$\frac{R_2}{R_1+R_2} = 0,66 \cdot R_1 = 0,34 \cdot R_2 \Rightarrow R_2 = 1,94 \cdot R_1$ así pues podemos concluir que aunque si que es importante el

valor de las resistencias estas cumplen unos características que relacionaran los dos valores, para nuestro caso usaremos resistencias de 1K y de 2K.



Ilustración 48 Circuito Divisor de corriente final

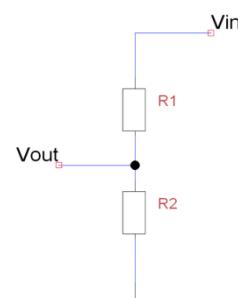


Ilustración 49 Esquema circuito divisor corriente

Para realizar el cálculo de la distancia haremos uso de la librería Pi4J, una vez que hemos detectado que tenemos un obstáculo delante nuestro ejecutaremos la rutina para que nos diga la distancia hasta dicho obstáculo.

La fórmula que aplicaremos será la siguiente $D_{cm} = T_{\text{d}s} \cdot ((340_{m/s} \cdot 100/10^{-6})/2)$ 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 $D_{cm} = T_{\text{d}s} \cdot 0,017$, 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();
    Thread.sleep(0, 10000);
    System.out.println(".");
    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;
}
```

Nos Encontramos un problema al integrar la función con la librería Pi4J, esta librería tiene un bug en su versión más reciente estable, que aun no está solucionado que impide que el sensor funcione de forma correcta.

Encontramos una solución a este último problema haciendo uso de la librería OpenJDK Device I/O comentada anteriormente en OPENJDK Device I/O y en el Anexo 3. Compilación OpenJDK DIO, hemos podido comprobar cómo las dos librerías son capaces de funcionar al mismo tiempo siempre y cuando el proceso se lance desde un función sincronizada y aseguremos que solo existe un objeto controlador del puerto GPIO, es decir tenemos que diseñar el la clase controladora como threatSafe, para permitir trabajar con múltiples hilos.

Prototipo 4 C Alimentación Raspberry Pi

Uno de las problemáticas que encontramos en el punto anterior era que el Controlador de Puente H tenía un regulador de corriente que se sobrecalentaba en exceso cuando alimentaba a la vez la RaspberryPi y los motores.

Para evitar este problema implementamos un regulador de corriente adicional externo a el controlador L298N.

El regulador de corriente que usamos es battborg ver Anexo 8 BattBorg que nos permitirá alimentar nuestra raspberryPi desde el puerto GPIO sin necesidad de usar un alimentador microusb.

Prototipo 4 E Detección avanzada de objetos

En la fase de diseño se planteo la capacidad de detectar obstáculos mediante distinto procesos, se diseño un ejemplo mediante el cual se detectaban obstáculos con círculos inscritos, pero esta técnica es escasa para las capacidades que planteamos.

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.

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.

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 más 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 sería 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 50 Detección canales por canal RGB
(<http://blog.electricbricks.com>)

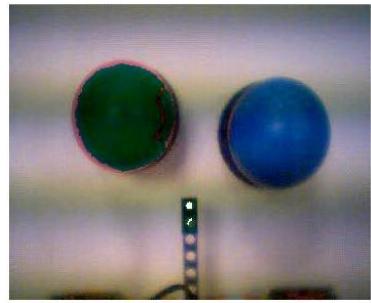


Ilustración 51 R E sultado detección objeto por canal RGB
(<http://blog.electricbricks.com>)

Otra forma más óptima sería analizar la imagen y aplicando diversos filtros llegar al punto de poder diferenciar sin lugar a dudas un color de otro, esto se puede realizar de varias funciones sobre el modelo de color HSV1, 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.

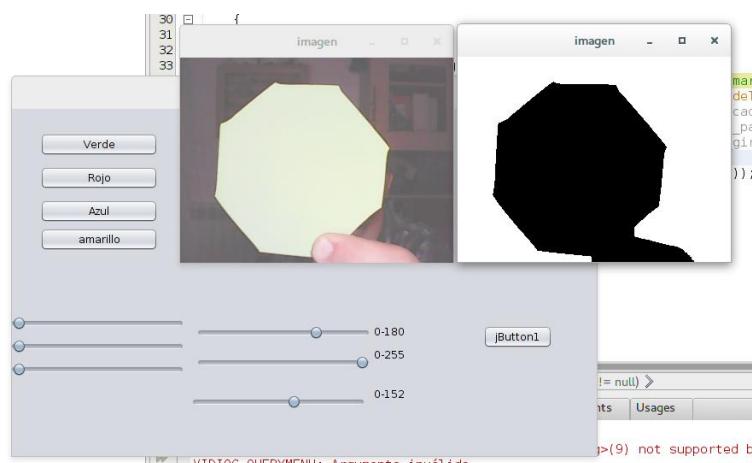


Ilustración 52 Filtro HSV sobre imagen

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.

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 más, algo que nos permita identificar un objeto de forma unívoca, 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.

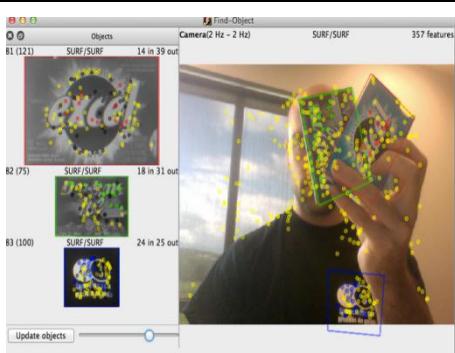


Ilustración 53 Detección objetos con Surf
(<https://code.google.com/p/find-object/wiki/FindObjectsWithWebcam>)



Ilustración 54 Ejemplo de objeto detectado con harrcascade

Una de las formas de detección de objetos por forma es hacer uso de SURF, es un algoritmo de visión por computador, capaz de obtener una representación visual de una imagen y obtener información detallada de esta. Esto nos permitirá reconocer objetos que anteriormente hayamos visto y seleccionado, entraría dentro del campo de la inteligencia artificial.

El problema principal de esta técnica es la cantidad de procesamiento necesario para detectar y analizar todos los puntos de interés de una 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, este método nos permite localizar objetos dentro de una imagen por medio de descriptores pre cargados en el sistema.

La localización de objetos por Haar es extremadamente rápida puesto que busca objetos que concuerden con las características del fichero que hemos recargado y que anteriormente hemos pre calculado.

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.

Para más información sobre cómo realizar un entrenamiento de Haar-Cascade ver Anexo 9. Clasificadores Haar.

Prototipo 4 F Programa de control Android

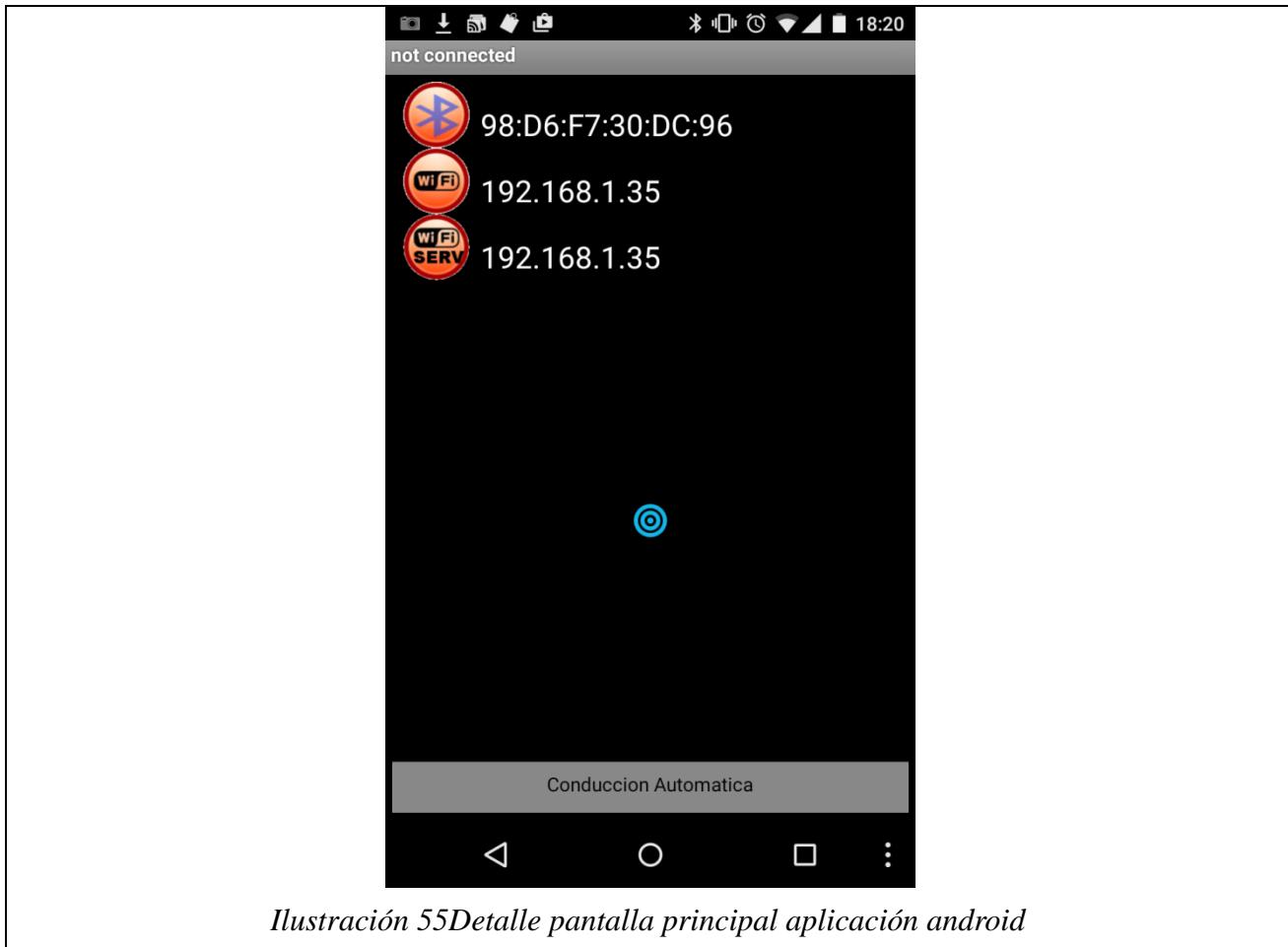
En esta fase haremos uso de una programa que tenemos creado de una práctica anterior realizada el año pasado en la asignatura aplicaciones móviles con David Chinarro como profesor docente.

Simplemente se ha readaptado la aplicación a los nuevos mensajes de control que utilizamos en este proyecto para ejecutar los movimientos.

También hemos rediseñado la ventana principal de la aplicación con el fin de tener un jostick táctil que nos permitirá ejecutar los movimientos en el coche de manera manual.

Debido a la extensión del documento que se genero en su momento no se ha puesto en anexos por no considerarse necesario pero puede consultarse en el CD adjunto en este proyecto en la carpeta "Anexos no incluidos" el documento "FDS Control Puerto GPIO RaspberryPi.PDF"

En la imagen adjunta podemos observar como ha quedado finalmente la pantalla de control del programa.



Prototipo 4 G Conducción autónoma

Como último punto del proyecto implementamos la capacidad de conducir autónomamente, es decir el coche es capaz de detectar diversos obstáculos que hemos creado mediante Entrenamientos HaarCascade ver Anexo 9. Clasificadores Haar, y actuar en consecuencia.

El proceso de conducción autónoma varía del diseñado al principio del proyecto en donde inicialmente detectábamos obstáculos, ahora detectamos distintos objetos que nos informan de cómo debemos actuar en consecuencia.

De esta forma completamos la funcionalidad permitiendo ruta en base a señales predefinidas en el sistema, Este sistema puede ser completado con diversos cascades que nos envíen alertas por ejemplo detección de personas, detección de caras...

En las imágenes vemos como el sistema es capaz de detectar varias señales de distintos tipos, para las cuales hemos realizado un entrenamiento exhaustivo de más de 120h de procesamiento, ver Anexo 9. Clasificadores Haar para mas información



Ilustración 56 Detección de dos objetos

Ilustración 57 Detección de un único objeto

El diagrama queda como el presentado a continuación

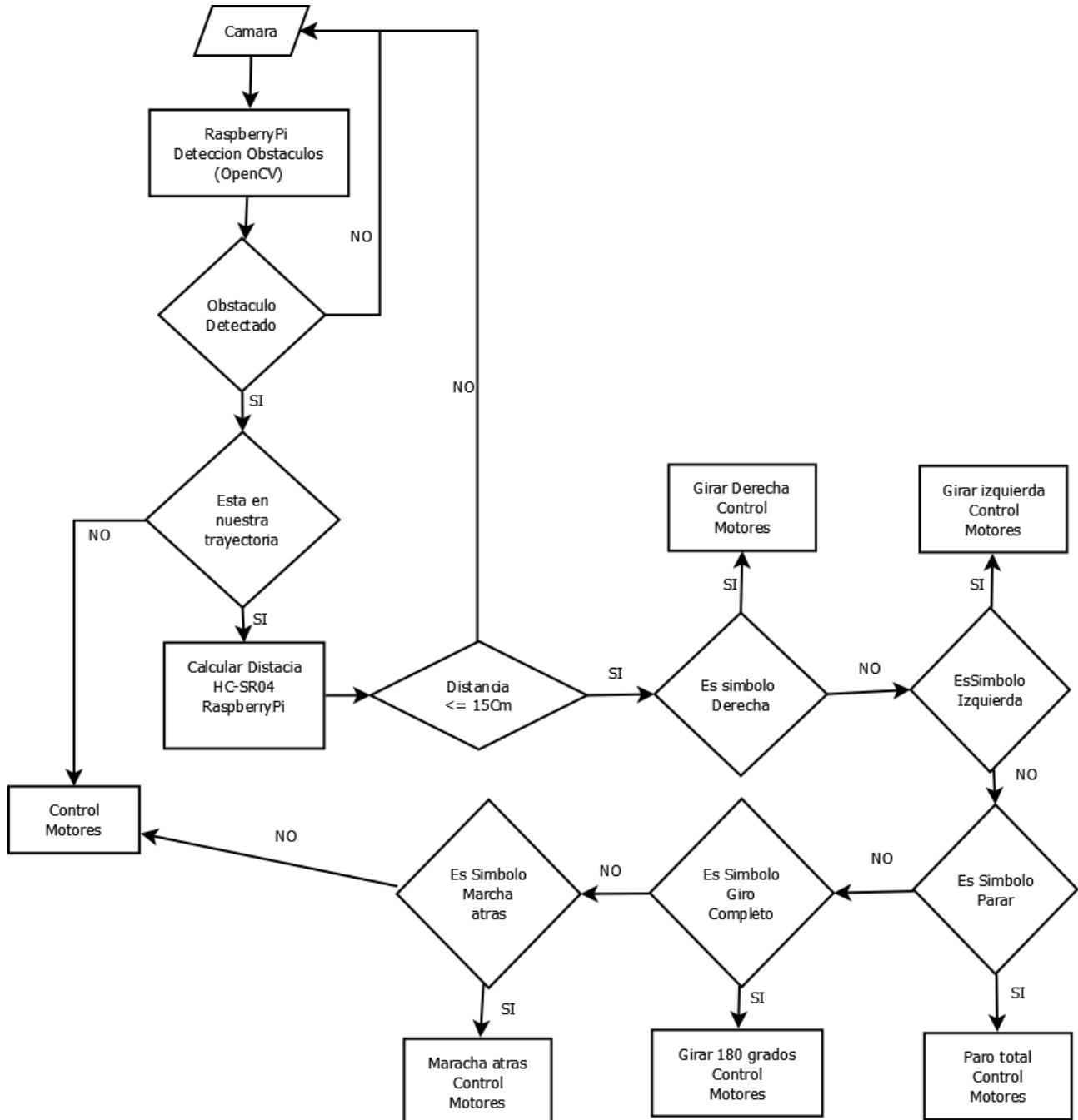
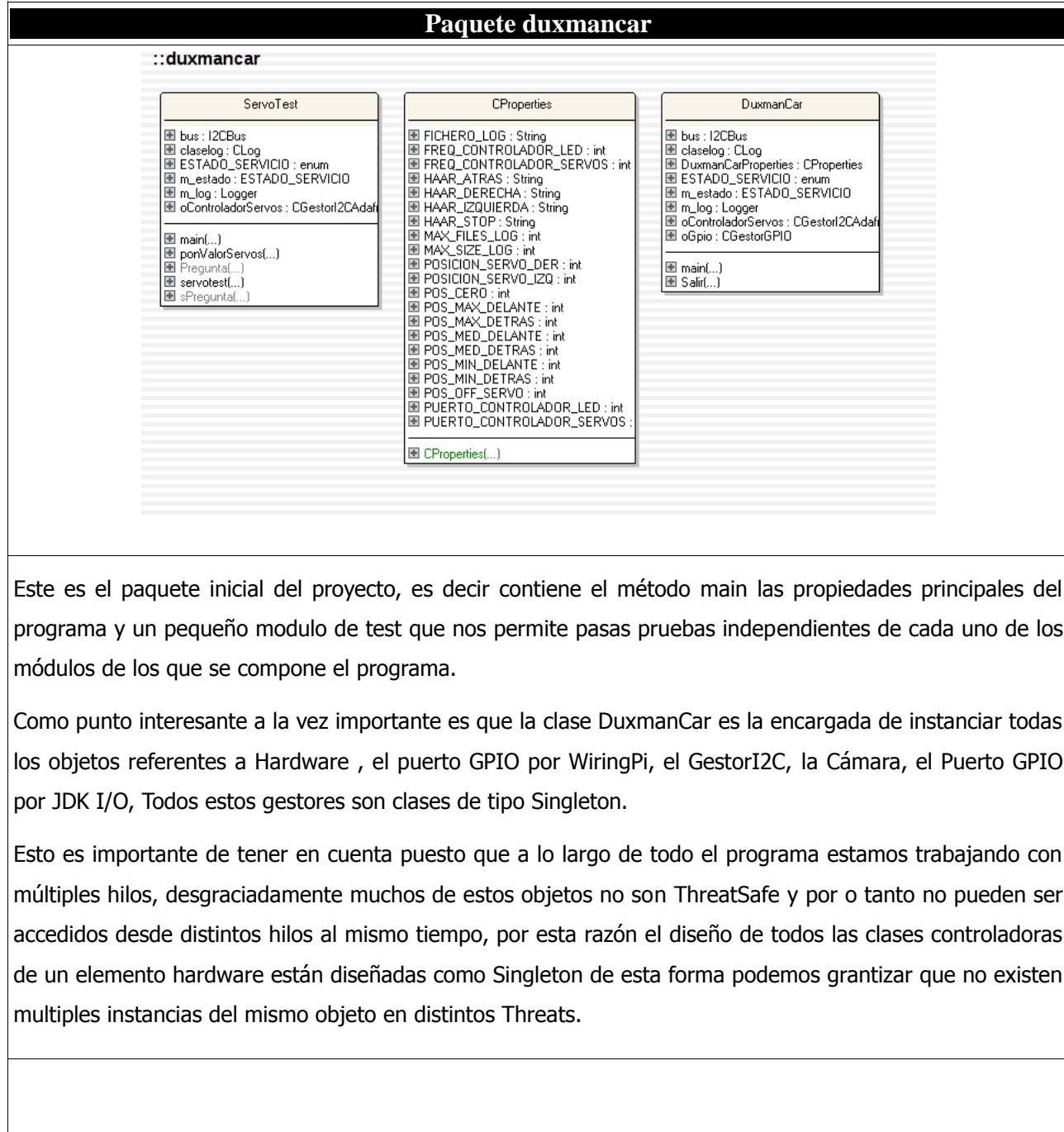
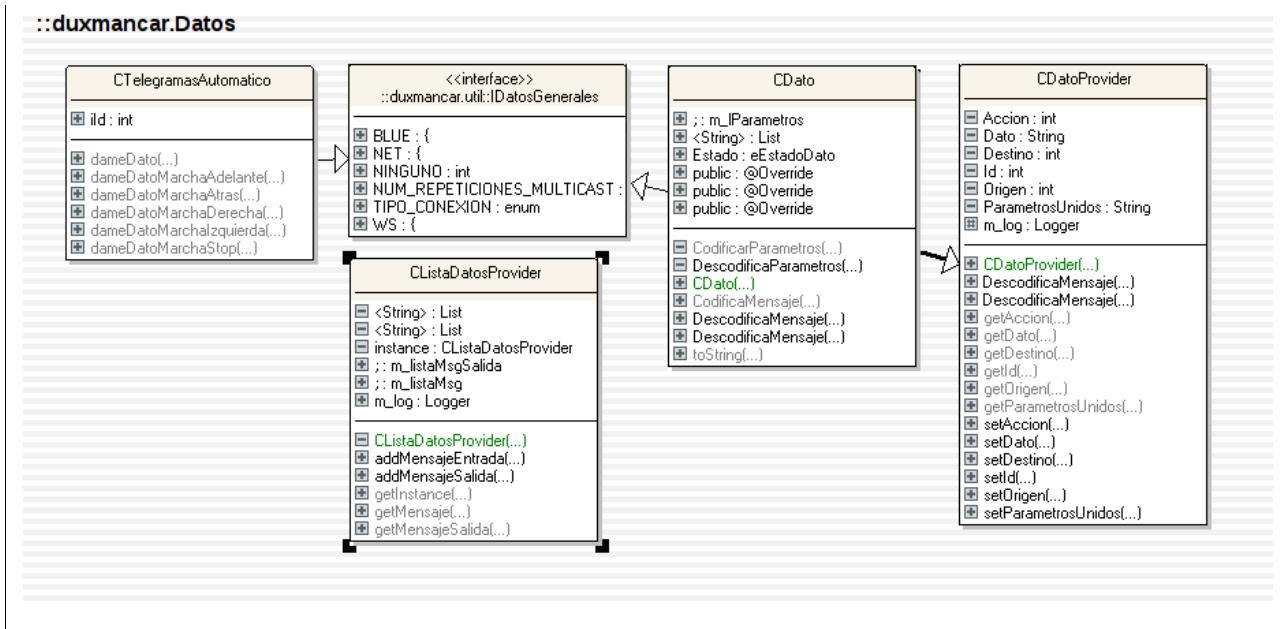


Ilustración 58DFD Control de dirección rediseñado

Diseño final de la aplicación

Programa control RaspberryPi

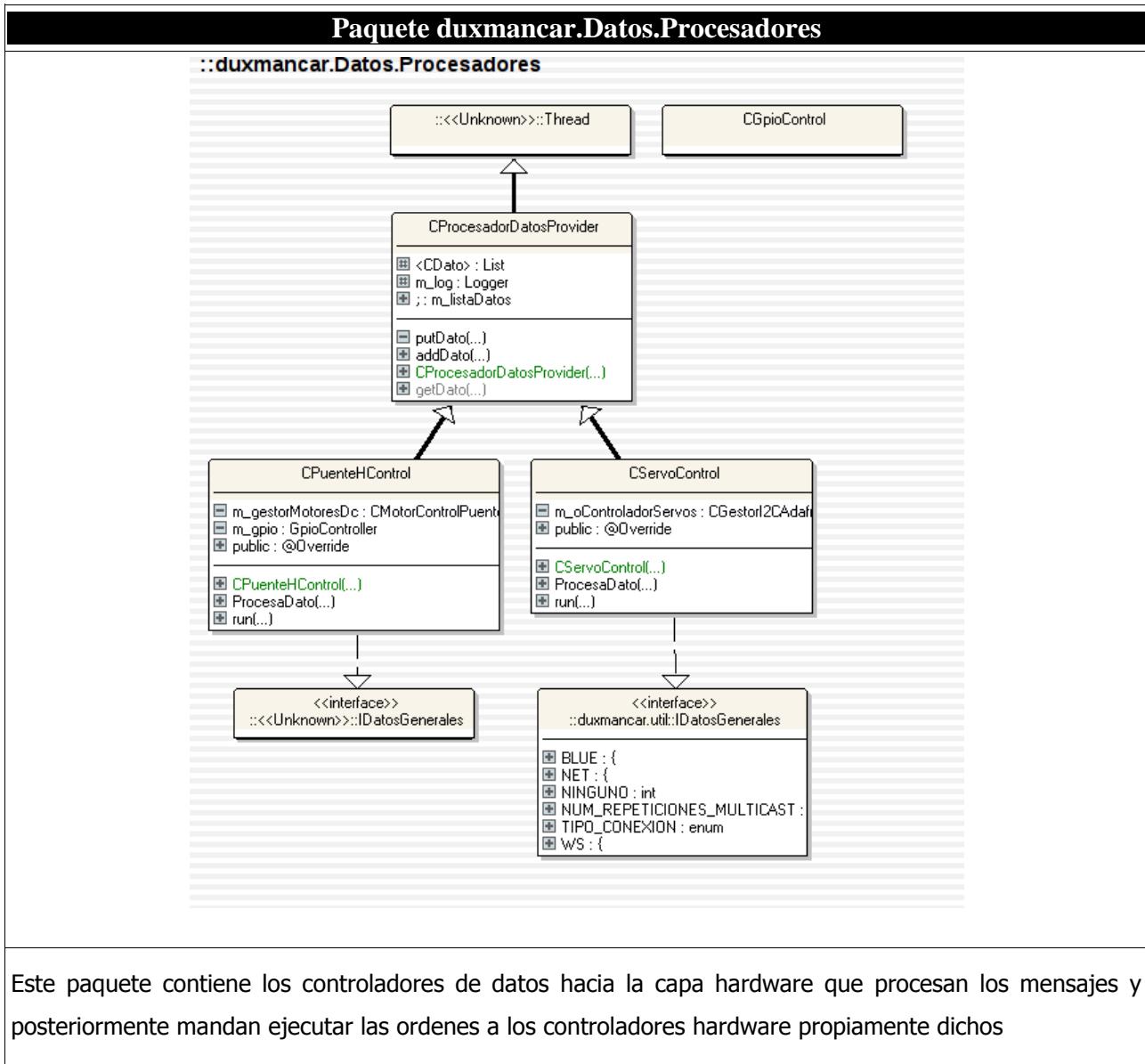




Este paquete contiene todos los elementos referentes a los datos que se transfieren entre los distintos subsistemas .

Clase	Descripción
Cdato	Estas son la clase básica de información dentro del sistema, todos los mensajes que se tratan dentro del sistema son instancias de estas clases.
CDatoProvider	<p>Como punto importante a destacar:</p> <p>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.</p> <p>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.</p>

CTelegramaAutomatico	<p>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 manobre de forma autónoma.</p>
ClistaDataProvider	<p>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.</p> <p>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.</p>

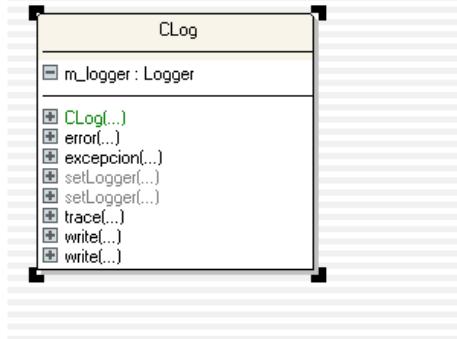


Este paquete contiene los controladores de datos hacia la capa hardware que procesan los mensajes y posteriormente mandan ejecutar las ordenes a los controladores hardware propiamente dichos

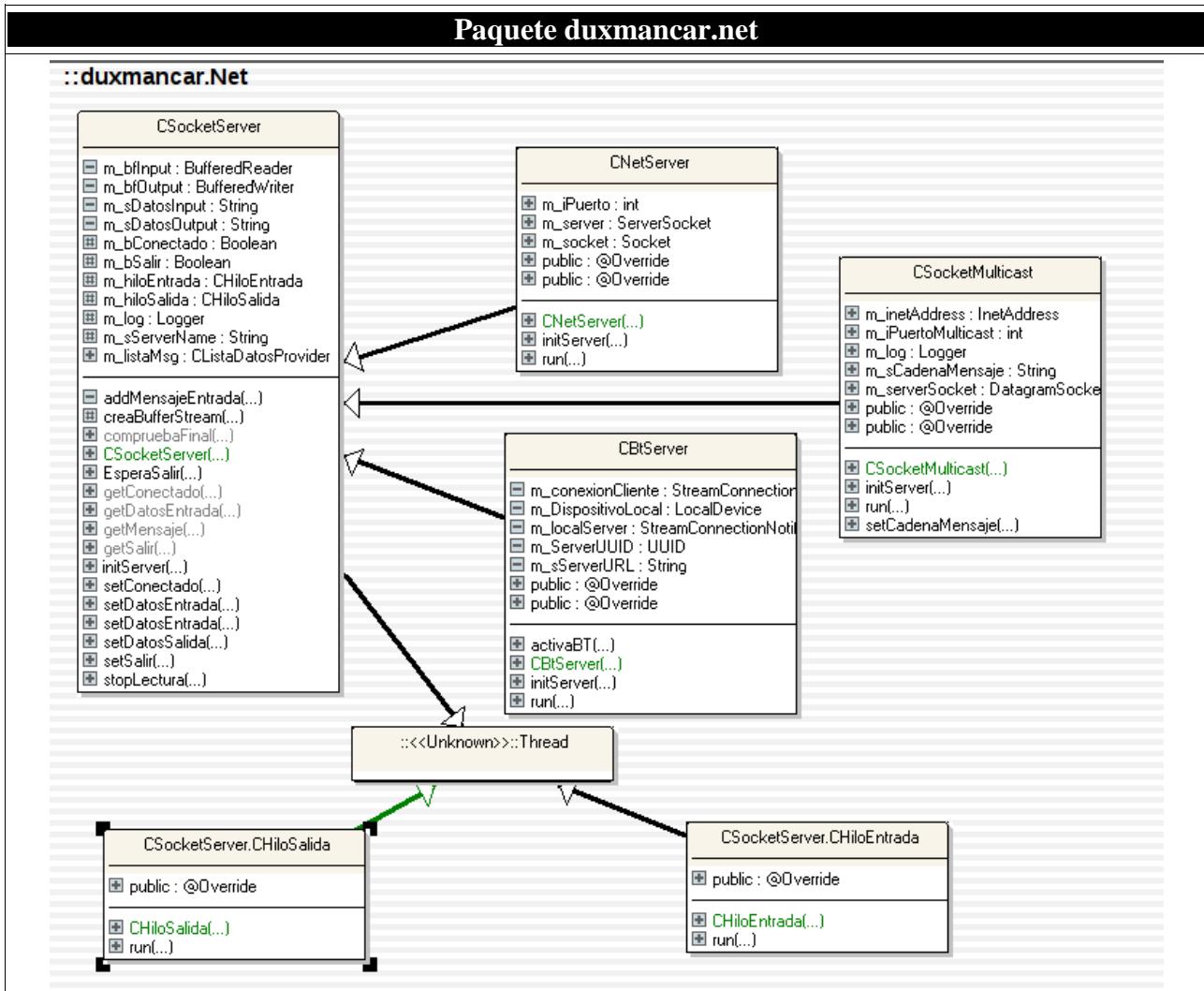
Clase	Descripción
CprocesadorDatosProvider	Clase abstracta que implementa la cola de mensajes FIFO y los métodos sincronizados para acceder y a los datos.
CpuenteHControl	Clase que implementa el controlador de puente H descrito en la sección hardware, nos permite tratar los mensajes específicos con destino el controlador Puente H para hacer 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 los Servo Motores.

Paquete duxmancar.log

::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



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 multithread 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 `ClistaDataProvider`.

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 socket que implementamos.

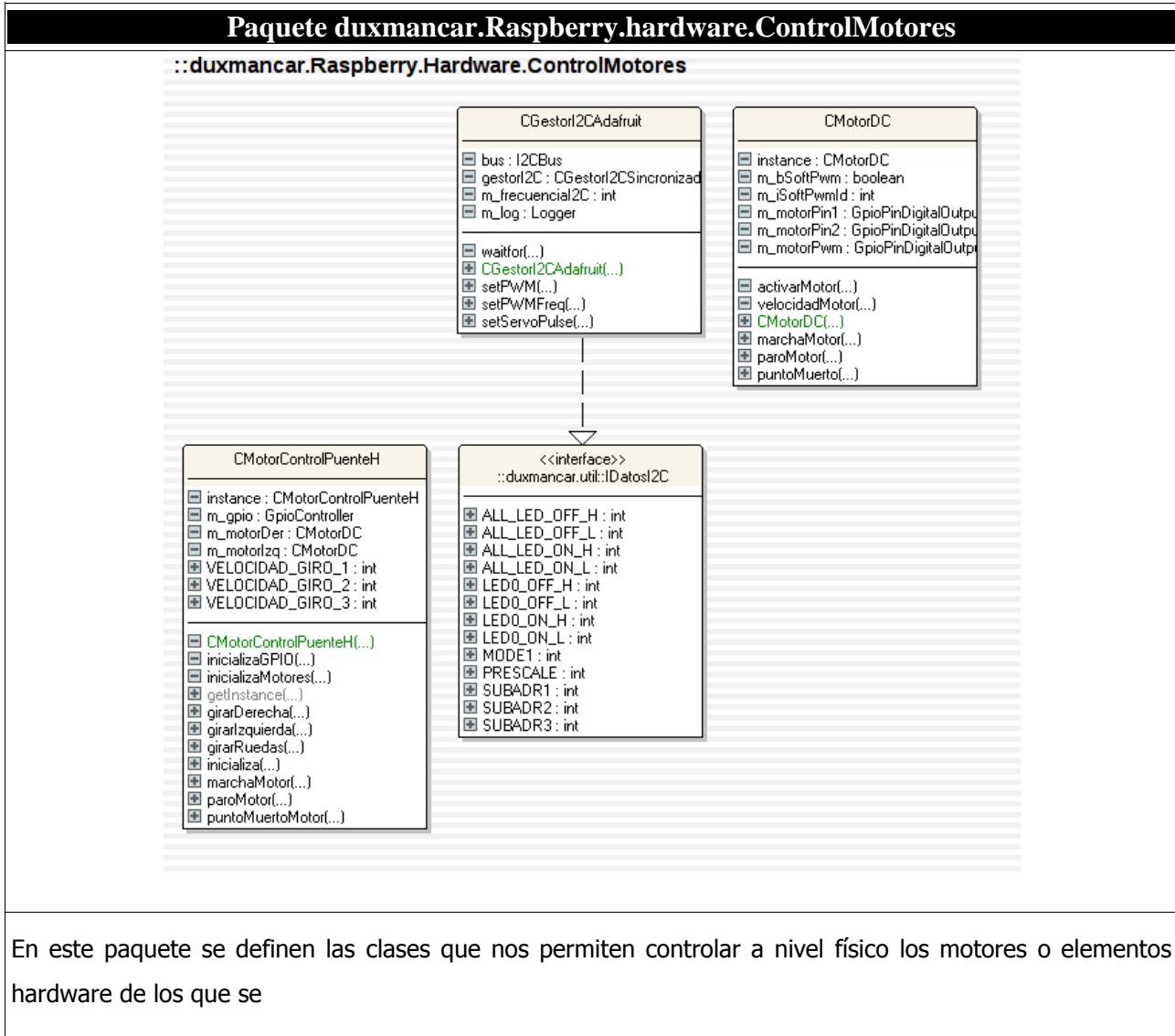
Clase	Descripción
CsocketServer	<p>Clase que gestiona las colas de entrada y salida, así como los buffer de escritura y lectura.</p> <p>Implementa internamente dos subclases para gestionar las comunicaciones de entrada y salida.</p> <p>Implementa métodos generales genéricos que permiten obtener los buffer de lectura y escritura desde los distintos tipos de socket</p>
ChiloEntrada ChiloSalida	Clases internas que controlan el flujo de información hacia los dos sentidos gestionan la cola de datos ClistaDataProvider 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.
CsocketMulticast	<p>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.</p> <p>Falta implementar flujo de conexión, actualmente en desarrollo.</p>

Paquete duxmancar.Raspberry.hardware

::duxmancar.Raspberry.Hardware

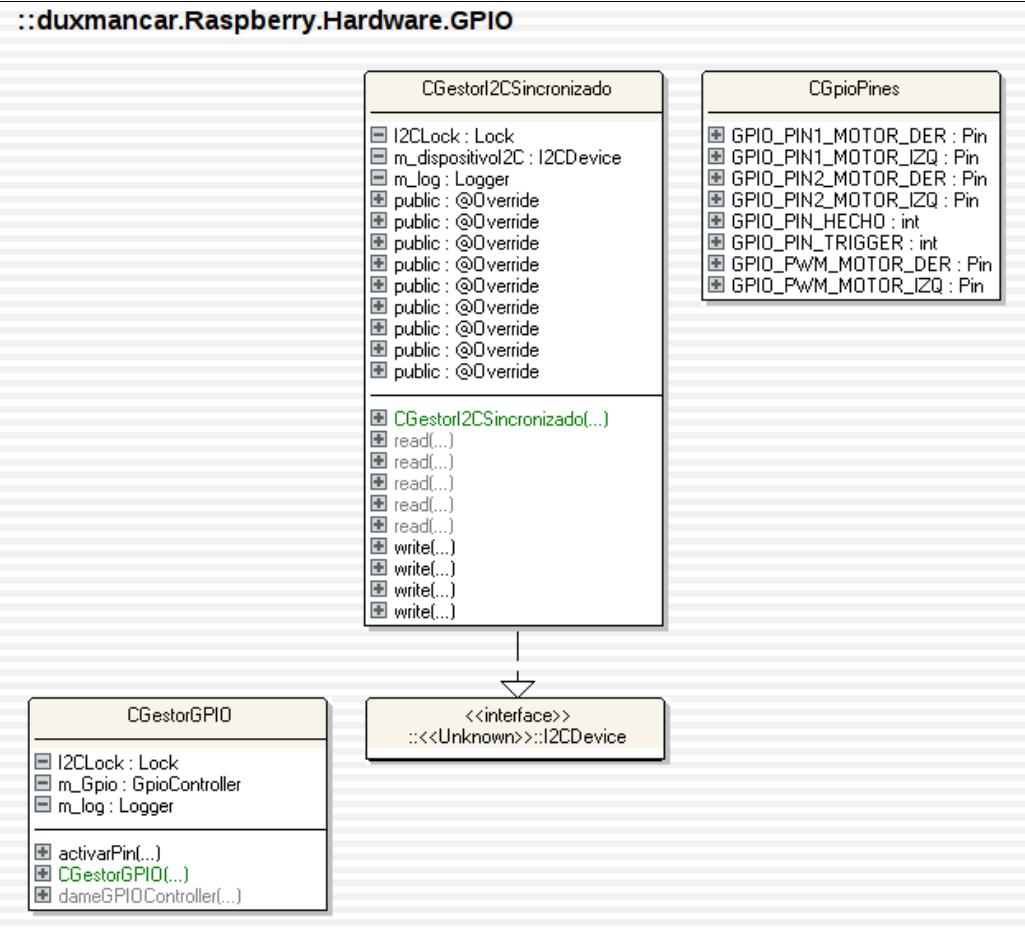


Este paquete únicamente contiene una clase de test que permite probar los sensores programados y los motores por medio de un menú de opciones.



Clase	Descripción
CmotorControlPuenteH	<p>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.</p> <p>Contiene las funciones específicas para generar los movimientos de los dos motores de forma conjunta.</p>
CmotorDC	<p>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</p> <p>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.</p>
CGestor12CAdafruit	<p>Esta clase permite controlar una interfaz i2c de adafruit para control de servos 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.</p>

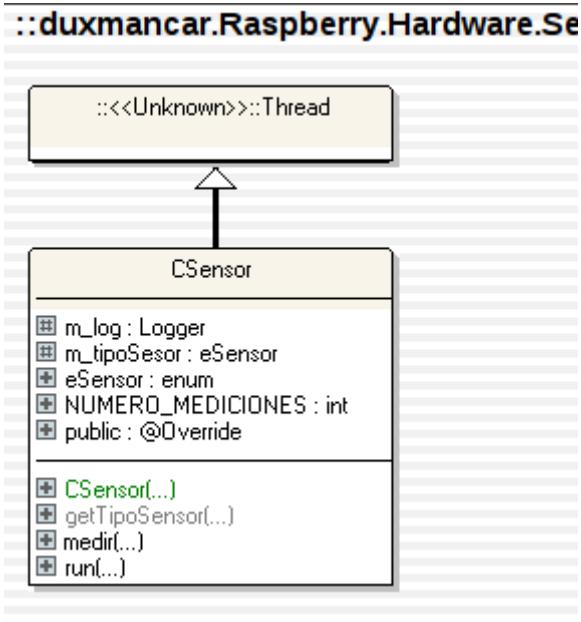
Dusmancar.Raspberry.Hardware.GPIO



Paquete de control de elementos hardware dependientes de GPIO

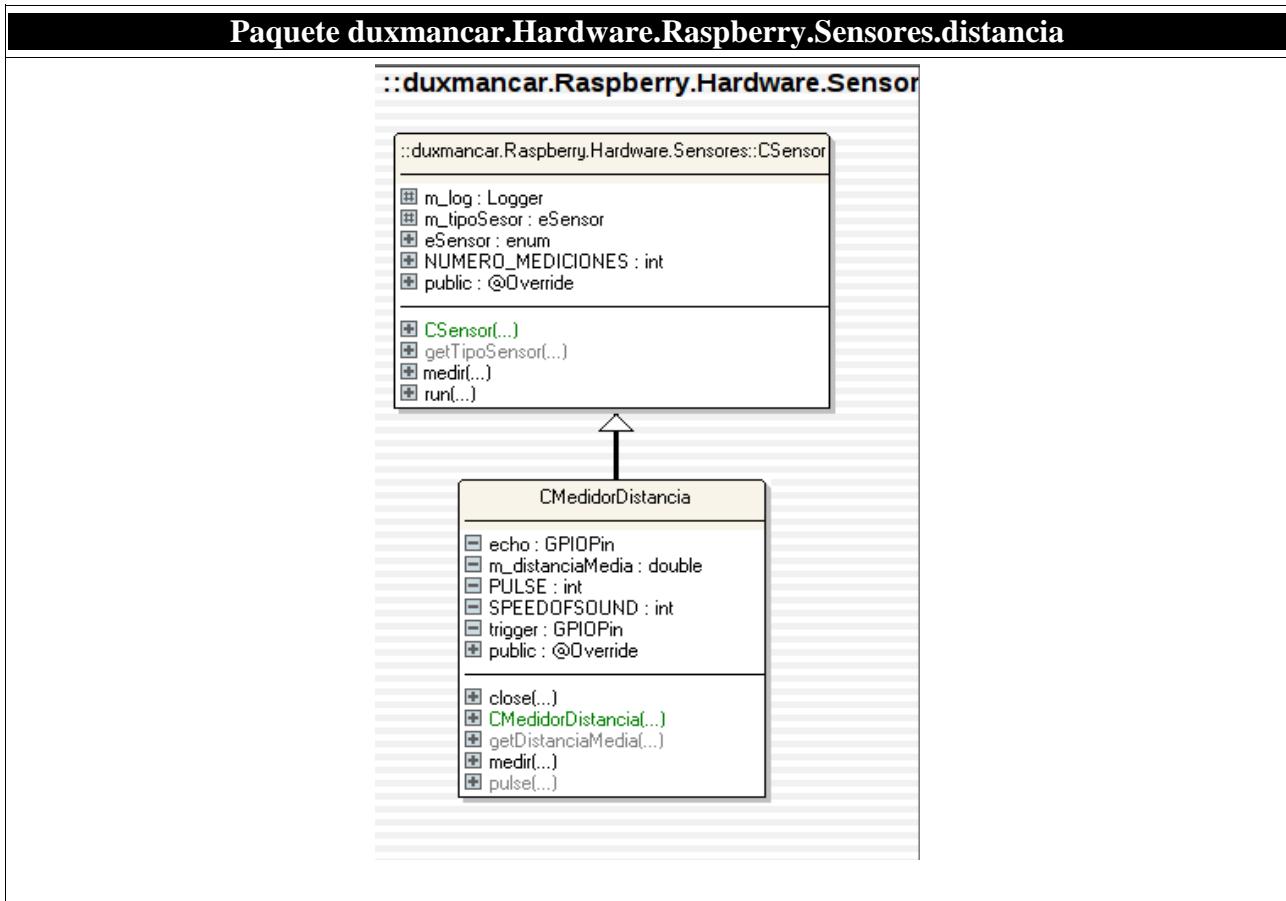
Clase	Descripción
CgestorGPIO	<p>Esta clase permite crear un sistema sincronizado y bloqueante del Puerto GPIO de la raspberry Pi para evitar el uso del dispositivo desde varios threads al mismo tiempo.</p> <p>Implementa un seguro de bloqueo que impide que otros hilos actúen sobre el puerto GPIO mientras estamos trabajando con él.</p>
CGestorI2CSincronizado	<p>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.</p>

Paquete duxmancar.Hardware.Raspberry.Sensores



Paquete superior que contiene la implementación de la clase básica de sensores.

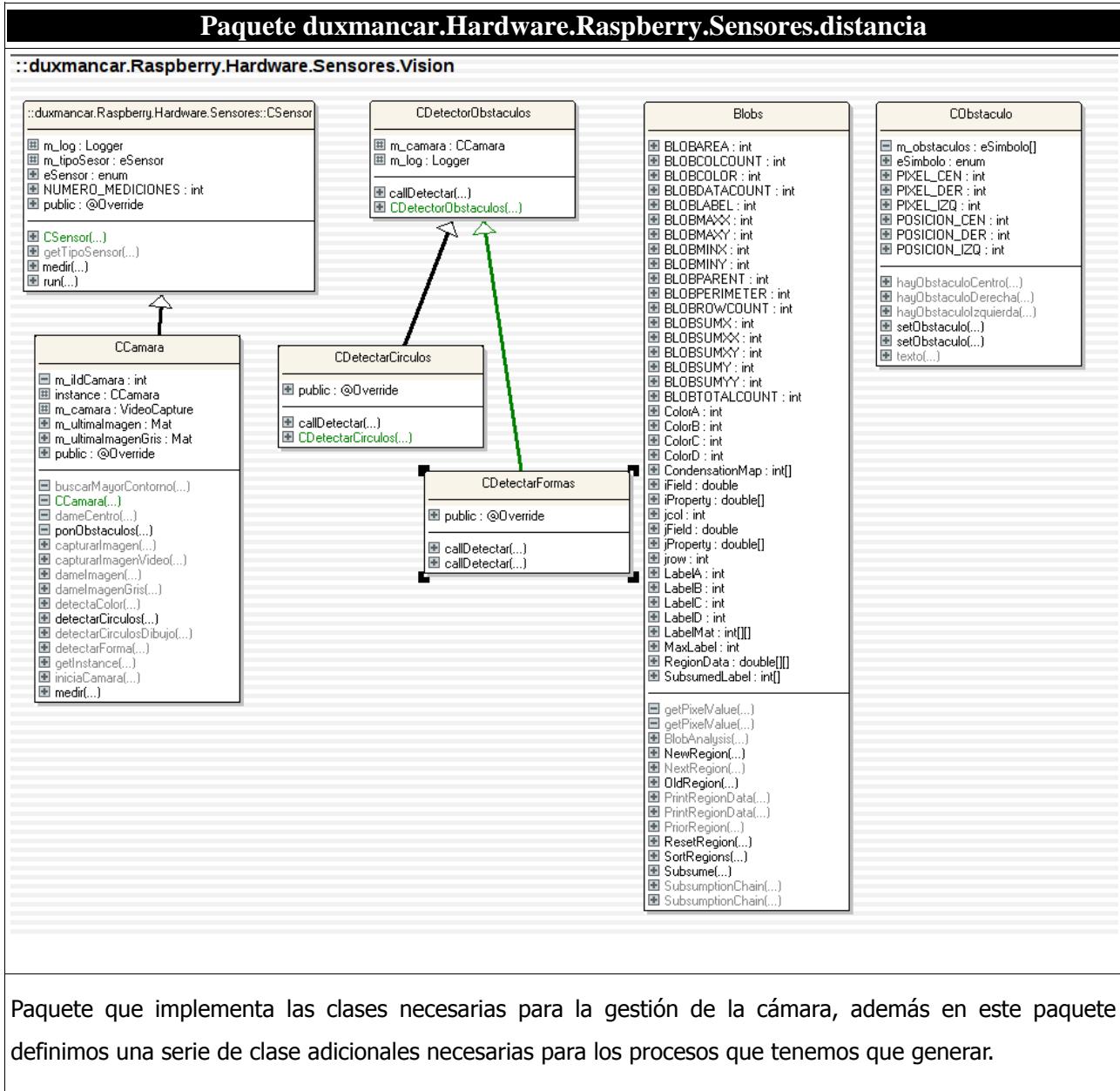
Se trata de una super clase que extiende de threat y únicamente implementa los métodos necesarios para identificar un sensor y el numero de mediciones que tiene que hacer para obtener un dato valido.



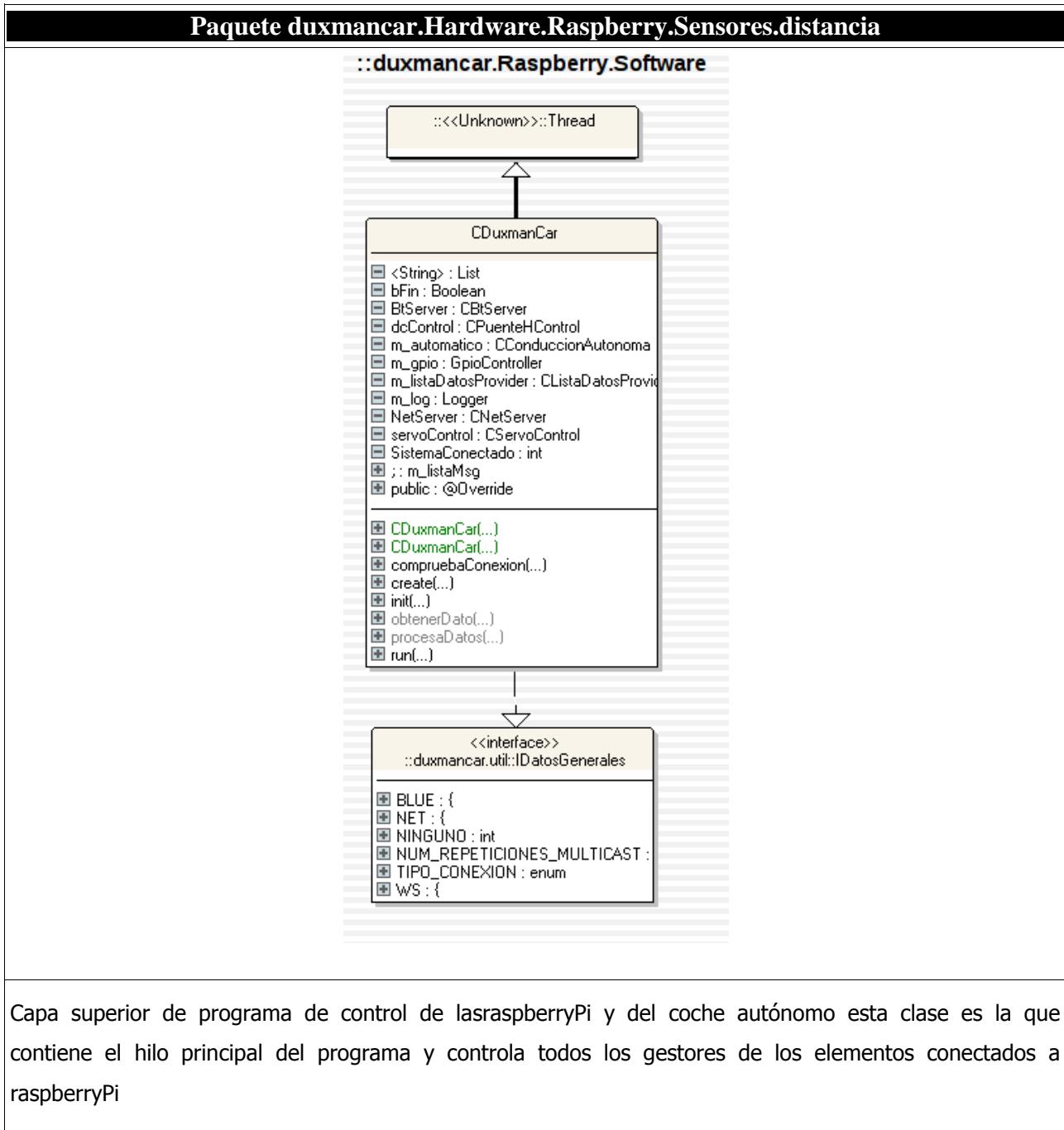
Paquete superior que contiene la implementación de la clase básica de sensores.

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	<p>Clase abstracta que contiene los métodos básicos a todos los sensores.</p> <p>Implementa el tipo de sensor, para permitir extender el proyecto añadiendo distintos sensores.</p> <p>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.</p>
CmedidorDistancia	<p>Clase que implementa el control del dispositivo hardware HC-SR04, sensor de distancia por ultrasonidos que ya hemos detallado en puntos anteriores.</p> <p>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.</p>

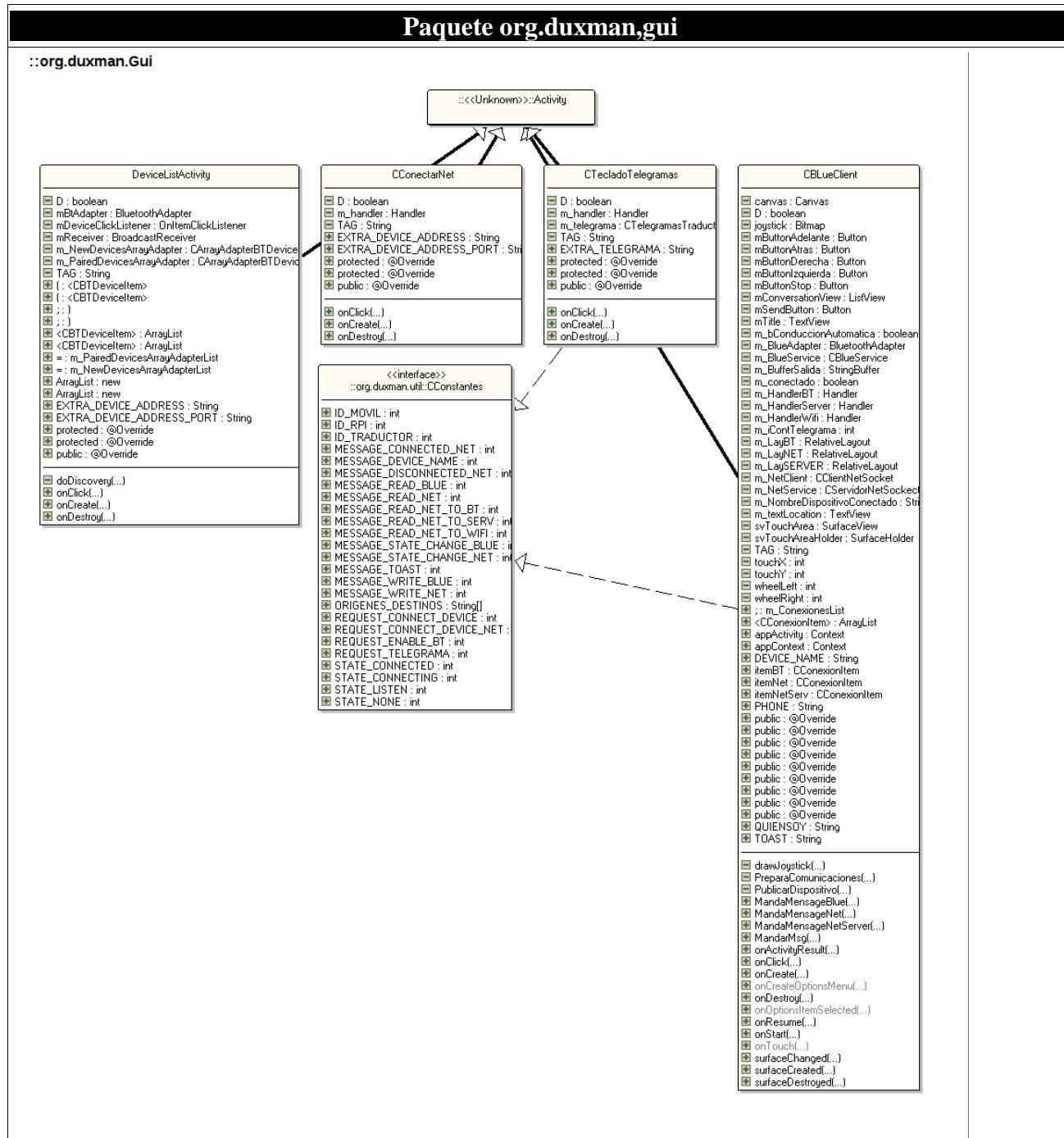


Clase	Descripción
CCamara	<p>Clase principal que nos ofrece una interfaz con la que obtener imágenes , hace la función de sistema de visión artificial controlado por el resto de clases.</p> <p>Actualmente se han desarrollado los siguientes detectores:</p> <ul style="list-style-type: none"> Formas Entrenadas HaarCascade Colores Filtro HSV Obstáculos genéricos con Bloobs Formas fijas Círculos y cuadrados <p>Se trata de una clase singleton para evitar el acceso desde distintos hilos al mismo hardware.</p> <p>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.</p>
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 específicas para la conducción automática
Bloobs	Esta es una clase que hemos tenido que implementar, puesto que la versión de OpenCV que estamos utilizando no implementa detección bloobs, se puede definir blob 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



Programa Control Android

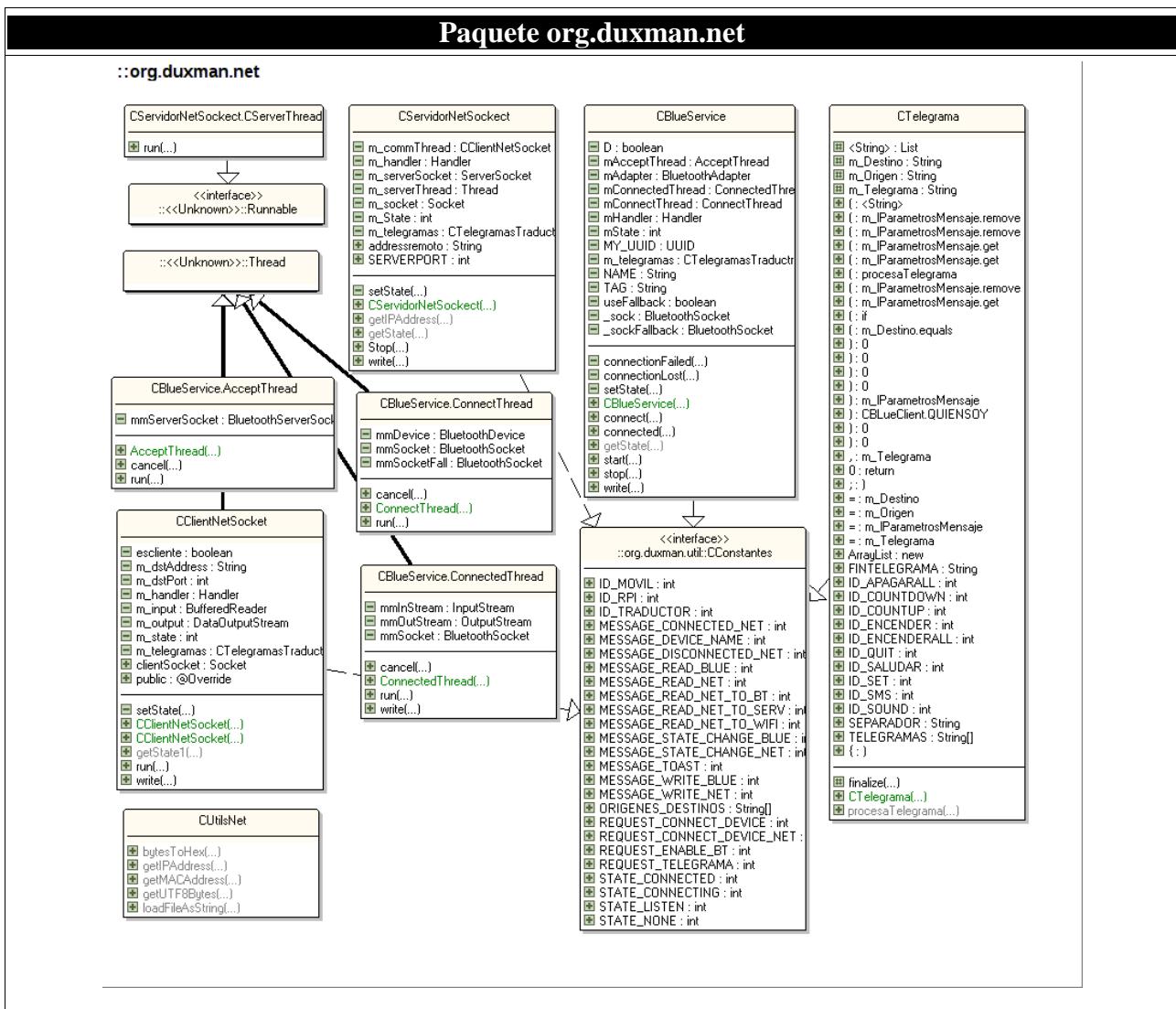
El programa de control android es un evolución de una práctica que realice el año pasado presentada para la asignatura de dispositivos móviles con David Chinarro, hemos aprovechado la interfaz y el core del programa para adaptarlo a nuestras necesidades actuales.



Este es el paquete principal del programa de control android, implementa todas las actividades, pantallas del programa, además implementa las clases clientes de los socket de conexión.

Por otra lado implementa todos los eventos que se ejecutan en el programa.

Clase	Descripción
CblueClient	<p>Clase principal del programa android, es la interfaz principal del programa he implementa todos los mandos de control necesarios para controlar el vehículo.</p> <p>También implementa los controles de conexión con nuestro coche permitiéndonos conectarnos mediante bluetooth o wifi , implementa un pequeño servidor de socket que permitirá realizar la conexión inversa siendo el dispositivo android el servidor y el coche el cliente , aun se está desarrollando esta funcionalidad, aunque la conexión es funcional es necesario realizar modificaciones en el sistema de control de la raspberryPi.</p>
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 nuestros 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 seleccionar mediante un pulsación, lanzando la activity necesaria para completar la conexión deseada.

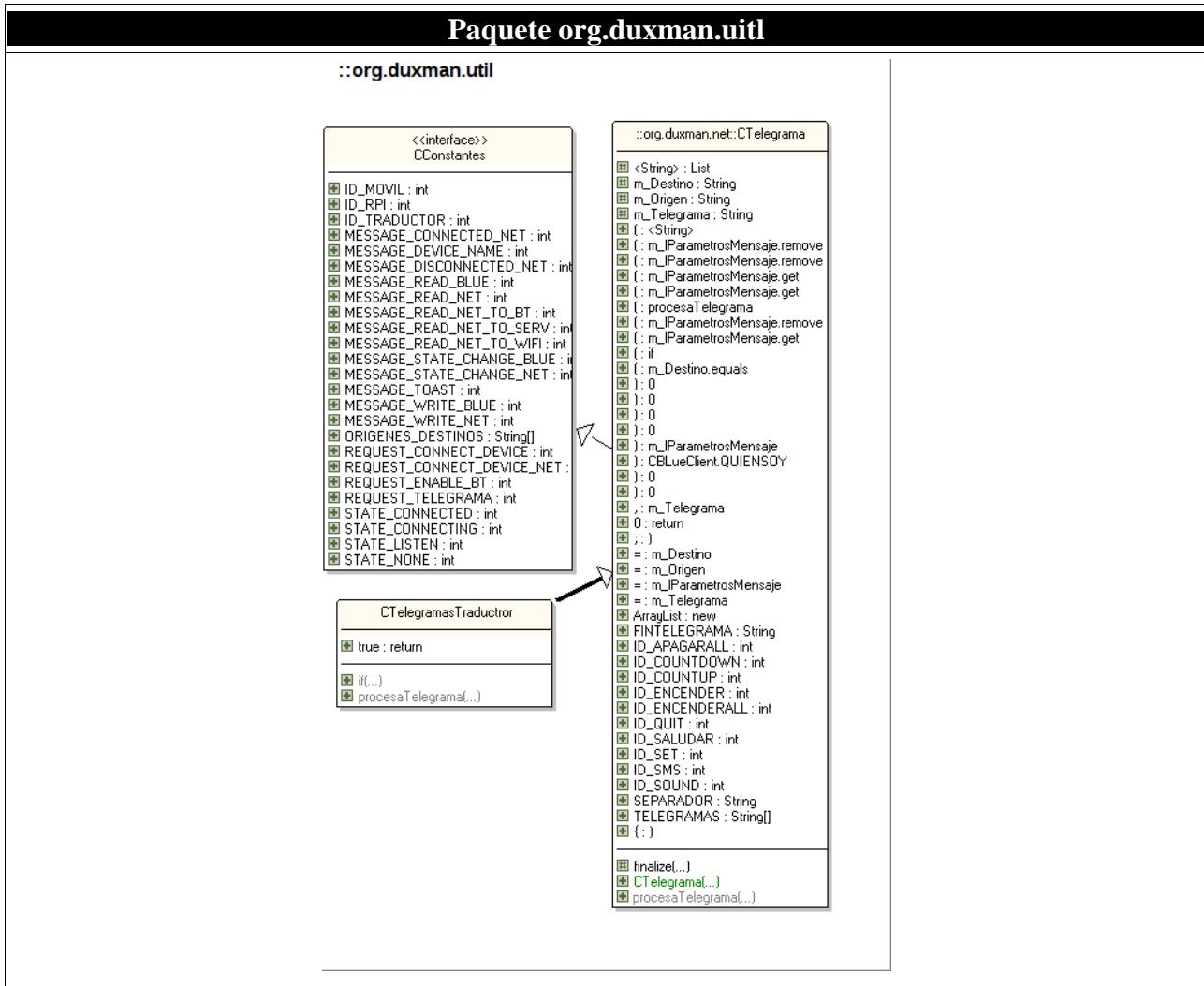


Paquete que implementa todo el sistema de conexión utilizado por el programa android.

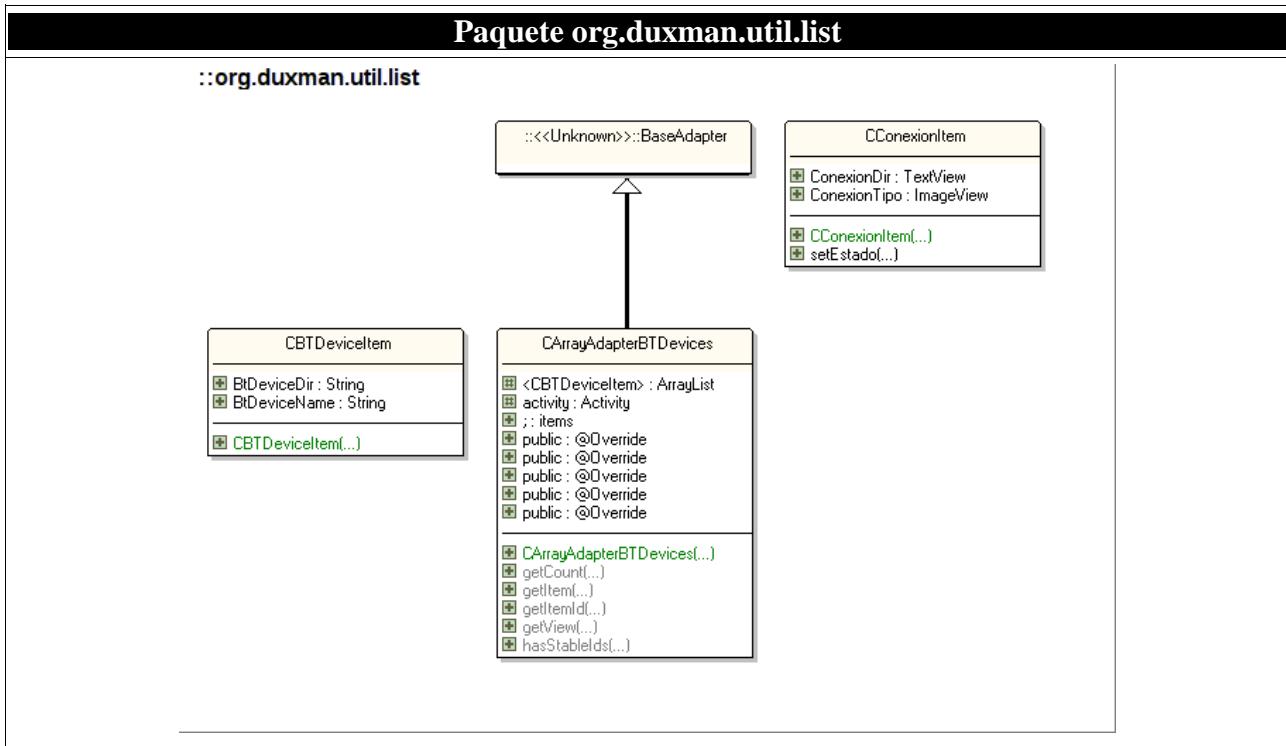
Aunque Android esta basado en java y nuestro programa de control de la raspberryPi también esta desarrollado en java no se ha podido reutilizar el código debido a las peculiaridades existentes en android sobre los métodos de conexión.

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 lógica de conexión a un socket 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 será 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 android a nuestra RaspberryPi.



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



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.
CconexionItem	Items de elementos de conexión , nos permite tener los elementos de conexión como si una lista se tratara e interactuar con ellos en el entorno grafico.

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 división 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 cuánto es el coste real del proyecto y cuanto es el coste imputable a los materiales físicos que necesitamos.

Coste material

Cada elemento refleja donde se ha comprado y su coste unitario.

Material	descripción	coste unitario	unidades	total
Servo Motores Tower 995	Comprado en Dx.com	6,67 €	4	26,68 €
MiniServo 9G	Comprado en Dx.com	2,69 €	4	10,76 €
Servomotor de rotación continua SM-S4303R	Comprado en Bricogeek.com	15,52 €	4	62,08 €
Motor DC TT Encoder	Comprado en Dx.com	2,59 €	6	15,54 €
PuenteH L298N	Comprado en Dx.com	5,61 €	2	11,22 €
Adafruit 16-Channel 12-bit PWM/Servo Driver	Comprado en modmypi.com	14,92 €	1	14,92 €
Sensor de Ultrasonidos HC-SR04	Comprado en modmypi.com	4,06 €	1	4,06 €
Consumibles electrónica	Comprado en modmypi.com	5,42 €	1	5,42 €

Interfaz transformador electrónico 5V	BattBorg,				
		Comprado en modmypi.com	20,34 €	1	20,34 €
WebCamara Logitech C170	USB				
		Comprado en amazon.es	15,99 €	2	31,98 €
Base montaje		Comprado en Dx.com	20,82 €	1	20,82 €
Placa portotipo		Comprado en ebay.com	3,85 €	1	3,85 €
Cableado variado		Comprado en Dx.com	2,89 €	2	5,78 €
RaspberryPi B+		Comprado en modmypi.com	37,99 €	1	37,99 €
TarjetaSD		Comprada en amazon.es	7,23 €	1	7,23 €
Caja RaspberryPi		Comprado en modmypi.com	8,13 €	1	8,13 €
usb wifi		Comprada en amazon.es	9,90 €	1	9,90 €
usb bluettoth		Comprada en amazon.es	11,85 €	1	11,85 €
Tornillería		Comprado en modmypi.com	1,05 €	10	10,50 €
Baterías y portabaterias		Comprado en modmypi.com	5,68 €	1	5,68 €
					324,73

Coste Diseño

Dentro de este punto vamos a calcular el coste desde el punto de vista de diseño

Material	coste unitario	unidades	total
Documentación de análisis funcional y antecedentes	24,00 €	150	3.600,00 €
Análisis del software de control (RaspberryPi)	20,00 €	120	2.400,00 €
Análisis del Software de gestión (Android)	20,00 €	25	500,00 €
Diseño Software	18,00 €	300	5.400,00 €
Pruebas	16,00 €	125	2.000,00 €
Montaje prototipo	14,00 €	24	336,00 €
Estudio de alternativas y posibilidades	19,00 €	75	1.425,00 €
			15.661,00 €

Los rangos salariales han sido extraídos del convenio provincial de Zaragoza del metal, se le ha aplicado un plus de convenio aplicado en varias empresas del sector de I+D.

CONVENIO PROVINCIAL DE COMERCIO DEL METAL

TABLAS SALARIALES 2015

	SALARIO DEF. 2015	PLUS CONVENIO DEF. 2015	ATRASOS 2013 ABONAR 1ER SEMEST 2015	ATRASOS 2013 ABONAR 2º SEMEST. 2015
PERSONAL TECNICO TITULADO				
Titulado grado superior (ingenieros, licenciados)	1.621,87 €	87,68 €	388,89 €	388,89 €
Jefe de equipo de informática	1.621,87 €	87,68 €	388,89 €	388,89 €
Titulados de grado medio (ayudantes técnicos)	1.400,45 €	81,08 €	337,03 €	337,03 €
Analista	1.400,45 €	81,08 €	337,03 €	337,03 €
Ayudante técnico sanitario	1.141,36 €	73,87 €	276,44 €	276,44 €
Maestro Industrial	1.141,36 €	73,87 €	276,44 €	276,44 €
PERSONAL MERCANTIL. TECNICO NO TITULADO				
Director de sección mercantil	1.621,87 €	87,68 €	388,89 €	388,89 €
Jefe de división	1.621,87 €	87,68 €	388,89 €	388,89 €

Ilustración 59 Extracto del convenio del metal aplicado a informática

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

Total coste por origen

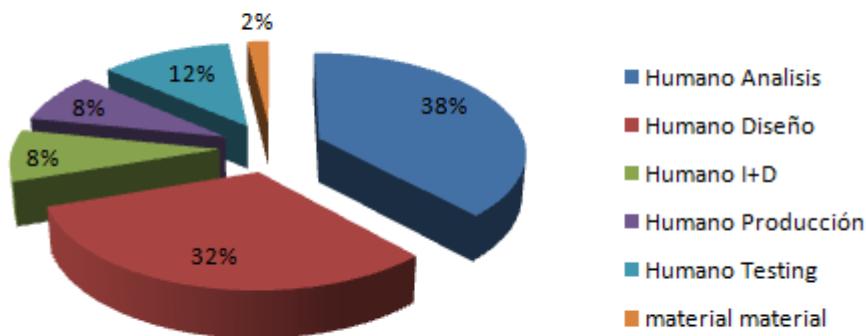


Ilustración 60 Distribución coste por sección

Total coste por tipo

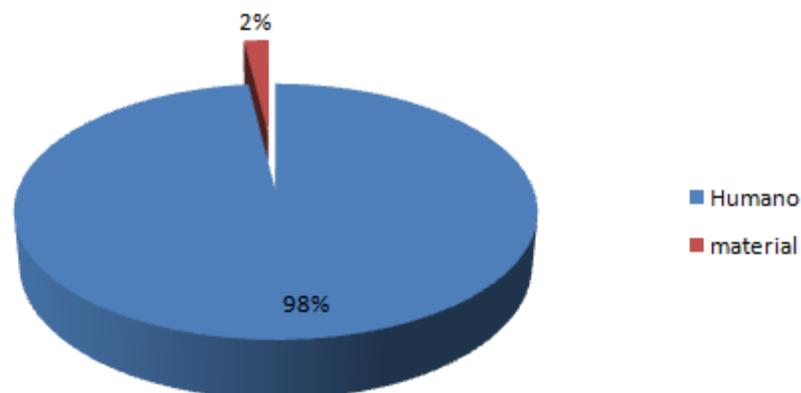


Ilustración 61 Distribución Coste Origen

Explotación de los costes

En esta sección mostramos un análisis completo de todos los coste según su origen

Tipo de coste	Suma de total
Humano	16725
Analisis	6500
Análisis del software de control (RaspberryPi)	2400
Análisis del Software de gestión (Android)	500
Documentación de análisis funcional y antecedentes	3600
Diseño	5400
Diseño Software	5400
I+D	1425
Estudio de alternativas y posibilidades	1425
Producción	1400
Montaje prototipo	1400
Testing	2000
Pruebas	2000
material	324,73
material	324,73
Adafruit 16-Channel 12-bit PWM/Servo Driver	14,92
Base montaje	20,82
Baterías y portabaterias	5,68
Cableado variado	5,78
Caja RaspberryPi	8,13
Consumibles electrónica	5,42
Interfaz BattBorg, transformador electrónico 5V	20,34
MiniServo 9G	10,76
Motor DC TT Encoder	15,54
Placa portotipo	3,85
PuenteH L298N	11,22
RaspberryPi B+	37,99
Sensor de Ultrasonidos HC-SR04	4,06
Servo Motores Tower 995	26,68
Servomotor de rotación continua SM-S4303R	62,08
TarjetaSD	7,23
Tornillería	10,5
usb bluettooth	11,85
usb wifi	9,9
WebCamara USB Logitech C170	31,98
Total general	17049,73

Reflexiones

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.

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.

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

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

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.

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 llegar 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 de aprender y recordar diversos obstáculos que se ha encontrado anteriormente y actuar en consecuencia.

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 seríamos capaces de movernos y calcular trayectorias así como obtener resultados más 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.

Sensores de posición

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

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 en cuenta.

GPS

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

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

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.

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.

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 qué 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.

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 desarrollando un Dron que me permita reconocer de forma aérea el terreno, en el cual aprovechare parte del desarrollo y los conocimientos adquiridos a lo largo del proyecto.

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

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

Referencias y bibliografía

1. Especificación técnica bus I2C
 - a. http://www.semiconductors.philips.com/acrobat_download/literature/9398/39340011.pdf
2. Definición del Bus i2C en
 - a. Wikipedia <http://es.wikipedia.org/wiki/I%C2%B2C>
3. DataSheet del controlador I2C PCA9685
 - a. <https://www.adafruit.com/datasheets/PCA9685.pdf>
4. Controlador de Servo motores de Adafruit, especificación y guia de uso
 - a. <http://www.adafruit.com/products/815>
5. Especificaciones Tecnicas Dispositivo de control de motores DC de PiBorg
 - a. <https://www.piborg.org/picoborgrev/specs>
6. Datasheet del chip mcp23017, expensor de puertos i2C
 - a. <http://ww1.microchip.com/downloads/en/DeviceDoc/21952b.pdf>
7. Referencia de programación del chip MCP23017 con wiringPi
 - a. <http://wiringpi.com/extensions/i2c-mcp23008-mcp23017/>
8. Descripción de RaspberryPi
 - a. http://es.wikipedia.org/wiki/Raspberry_Pi
9. Página oficial de RaspberryPi
 - a. <http://www.raspberrypi.org/>
10. Referencia de la librería de programación Pi4J
 - a. <http://pi4j.com/>
11. Implementación del modulo de adafruit de control de servoMotores por ledouris
 - a. <http://www.ledouris.net/RaspberryPI/servo/readme.html>
12. implementación de un coche RC con raspberryPi
 - a. <http://www.sistemasorp.es/2013/03/23/televigilancia-con-un-coche-rc-arduino-y-la-raspberry-pi/>
13. calibración estero de cámaras con openCV
 - a. <http://www.cnblogs.com/wangyaning/p/4236976.html>
14. Calibración estero de cámaras
 - a. <http://www.jayrambhia.com/blog/stereo-calibration/>
15. como reconocer objetos con visión artificial
 - a. http://es.wikipedia.org/wiki/Reconocimiento_de_Objetos
16. Como realizar un haar training
 - a. <http://www.mememememememe.me/training-haar-cascades/>

17. Haar Training
 - a. http://www.trevorsherrard.com/Haar_training.html
18. Como entrenar a openCV
 - a. <http://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html>
19. Opencv haartraining
 - a. <http://note.sonots.com/SciSoftware/haartraining.html>
20. Como compilar bluecove para raspberry pi
 - a. <https://sites.google.com/site/opengaragedoor1/home/stages/raspberry-pi>
21. como instalar bluecove en raspberry pi
 - a. <http://lukealderton.co.uk/blog/posts/2015/january/raspberry-pi-bluetooth-using-bluecove-on-raspbian.aspx>
22. Android rover controller
 - a. <http://www.pezzino.ch/android-rover-controller/>
23. Calibración de camaras en opencv
 - a. <http://computervisionandjava.blogspot.com.es/2013/10/camera-cailbration.html>
24. Búsqueda de objetos con opencv
 - a. <https://code.google.com/p/find-object/wiki/FindObjectsWithWebcam>
25. Introducción al desarrollo con opencv y java
 - a. http://docs.opencv.org/doc/tutorials/introduction/desktop_java/java_dev_intro.html#java-dev-intro
26. reconocimiento de señales de trafico
 - a. http://es.wikipedia.org/wiki/Reconocimiento_de_se%C3%B1ales_de_tr%C3%A1fico
27. Vehículo rc controlado por raspberrypi
 - a. <http://www.raspberryshop.es/wp/vehiculo-con-raspberry-pi-controlado-distancia-por-bluetooth/>
28. RC car over web
 - a. <http://blog.kaazing.com/2013/04/01/remote-controlling-a-car-over-the-web-ingredients-smartphone-websocket-and-raspberry-pi/>
29. Pibot-B
 - a. <http://www.retas.de/thomas/raspberrypi/pibot-b/index.html>
30. Raspberrypi y los pines implementación de una casa domotica
 - a. <http://www.peatonet.com/raspberry-pi-y-los-pines-gpio-implementando-domotica-asequible-parte-i/>
31. librería de programación de visión artifical en java
 - a. <https://github.com/sarxos/webcam-capture>

32. protocolos y java

a. <https://unpocodejava.wordpress.com/category/protocolos/>

33. Raspberry pi y bluetooth

a. <http://www.diverteka.com/?p=1880>

34. Modelo del color

a. HSV http://es.wikipedia.org/wiki/Modelo_de_color_HSV

35. Definición de Surf

a. <http://es.wikipedia.org/wiki/SURF>

Anexos

Anexo 0 Propuesta proyecto



Propuesta de Proyecto Final

Nombre alumno: Antonio Duce Gimeno

Titulación: Ingeniería Informática

Curso académico: 2014-2015

1. TÍTULO DEL PROYECTO

Vehículo teledirigido con una Raspberry Pi

2. DESCRIPCIÓN Y JUSTIFICACIÓN DEL TEMA A TRATAR

Diseñar un vehículo controlado remotamente que permita grabar y ver vídeo en tiempo Real.

La comunicación del sistema se realizará por Bluetooth o Red inalámbrica, el vehículo se podrá controlar desde una aplicación android o desde el PC.

3. OBJETIVOS DEL PROYECTO

Los objetivos principales de proyecto son los siguientes:

1. Diseñar un vehículo controlado remotamente con elementos electrónicos de bajo coste.
2. Diseñar una aplicación para RaspberryPi que nos permita actuar sobre los puertos GPIO y que permita comunicarse por bluetooth o WiFi.
3. Diseñar una aplicación de RaspberryPi que permita la transmisión de video en tiempo real.
4. Diseñar una aplicación android que permita comunicarse por bluetooth o WiFi con una RaspberryPi y controlar remotamente sus Puertos GPIO, así como recibir video en tiempo real.

4. METODOLOGÍA

Como metodología de desarrollo del proyecto utilizaremos un modelo de metodología ágil, basada en métodos kanban por la cual dividiremos el proyecto en pequeñas metas alcanzables y autónomas entre sí que den pequeños avances funcionales al proyecto.

Esta metodología requiere realizar un estudio de todas las partes del proyecto previo a comenzar a desarrollarlo y nos permitirá avanzar homogéneamente tanto en el desarrollo como en la documentación.

5. PLANIFICACIÓN DE TAREAS

6. OBSERVACIONES ADICIONALES

Anexo 1. Compilar librería BlueCove en RaspberryPi

Seguiremos los siguientes pasos:

1. Descargamos las herramientas necesarias en el sistema

```
sudo apt-get install bluetooth bluez-utils blueman
```

2. 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
```

3. 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=
```

4. Descomprimimos el código fuente generado

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

5. 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
```

6. 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
```

7. 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
```

8. Copiamos los resultados

```
cp /target/libbluecove.so /jars/libbluecove_arm.so
```

Anexo 2. Instalación y uso de librería Pi4J

Ante todo tenemos que tener en cuenta que es una librería para el manejo y control de puertos GPIO sobre Java por lo tanto tenemos que cumplir los siguientes requisitos:

- Java Runtime, instalado por defecto en el sistema operativos Raspbian
- WiringPi Librería Nativa, las versiones actuales de Pi4J llevan incluida esta librería y por lo tanto no es necesario instalarla.

Para instalar la librería simplemente tendremos que descargar el paquete desde los repositorios oficiales y descomprimir la librería en el directorio que nosotros deseemos.

1. Descargamos el paquete

```
Wget http://get.pi4j.com/download/pi4j-1.0.deb
```

2. Instalamos el paquete

```
sudo dpkg -i pi4j-1.0.deb
```

A la hora de usar la librería tenemos que tener en cuenta las siguientes características de nuestro entorno como por ejemplo como se numeran los puertos GPIO de nuestra raspberryPi y como se numeran tanto física como virtualmente dentro de la librería.

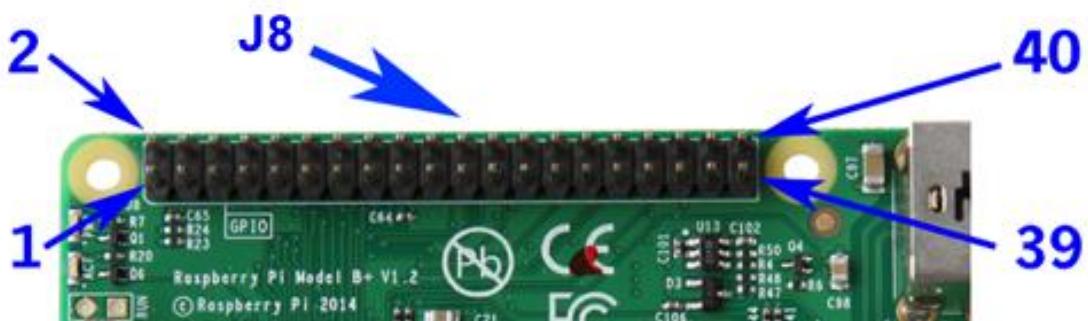
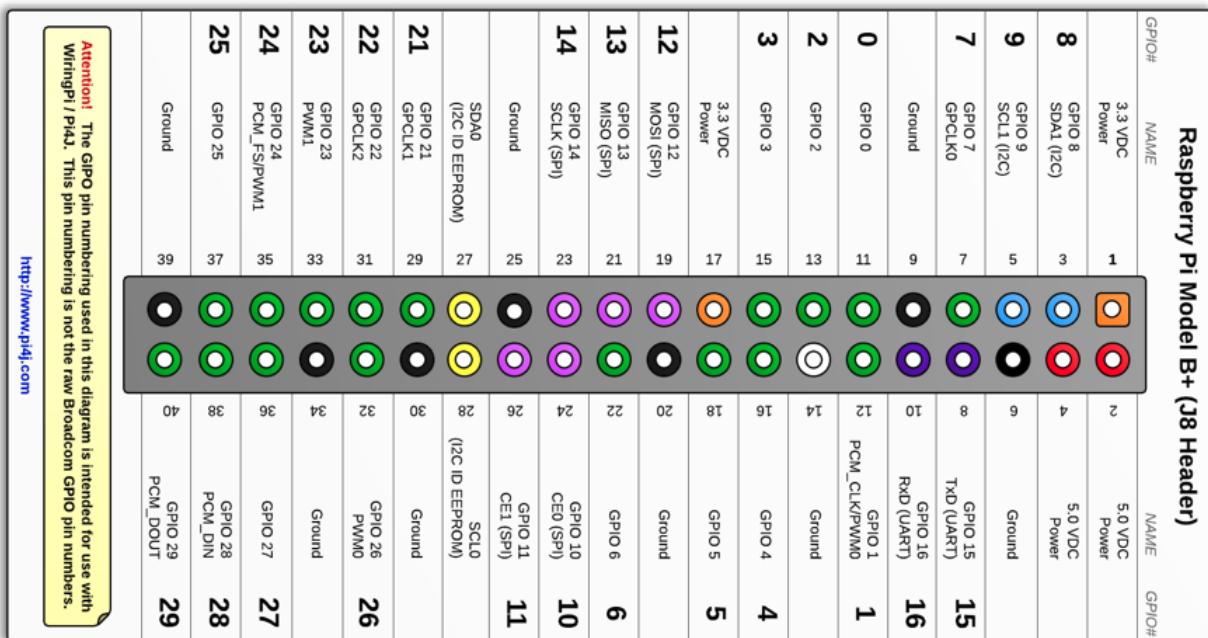


Ilustración 62 Puerto GPIO de raspberryPi B+

www.pi4j.com



Attention! The GPIO pin numbering used in this diagram is intended for use with WiringPi / PIA4. This pin numbering is not the raw Broadcom GPIO pin numbers.

Anexo 3. Compilación OpenJDK DIO

Al tratarse de una librería genérica y de bajo nivel será necesario compilar el código fuente con la finalidad de conseguir una librería que funcione en nuestro entorno para ello seguiremos los siguientes pasos.

1. Actualizamos nuestro entorno de sistema

```
sudo add-apt-repository ppa:webupd8team/java  
sudo apt-get update  
sudo apt-get upgrade  
sudo rpi-update
```

2. Instalamos las dependencias necesarias para compilar

```
sudo apt-get install build-essential mercurial
```

3. 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
```

4. Descargamos el código fuente del repositorio

```
hg clone http://hg.openjdk.java.net/dio/dev  
export DIO_DEV=/java-dio/dev
```

5. Compilamos

```
make
```

6. Copiamos resultados a los directorios de sistema

```
cp -r $DIO_DEV/build/deviceio/lib/* $JAVA_HOME/jre/lib
```

7. 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
```

Anexo 4. Compilación de OpenCV en RaspberryPi

Dado que nuestro sistema de control en RaspberryPi tenemos la necesidad de compilar la librería dentro de este sistema para conseguir un conjunto de binarios compatibles con la tecnologí ARM.

Existe la posibilidad de realizar un compilación denominada CrossCompiling pero enb diversas pruebas realizadas hemos detectar que los resultados no son todo lo óptimos que deseamos, incluso hemos llegado al punto de que el binario no es estable en nuestro entorno destino.

El resultado que nos interesa de esta generación son dos ficheros específicos "libopencv_java2411.so" y "opencv-2411.jar" que son respectivamente la librería de desarrollo de opencv y la interfaz java para nuestro proyecto.

Para compilar la librería OpenCV seguiremos los siguientes pasos:

1. Actualizamos nuestro entorno de sistema

```
sudo add-apt-repository ppa:webupd8team/java  
sudo apt-get update  
sudo apt-get upgrade  
sudo rpi-update
```

2. 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
```

3. 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
```

4. 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-3.0.0  
mkdir build  
cd build
```

5. Generamos los ficheros de compilación

```
export JAVA_HOME=/usr/lib/jvm/jdk-8-oracle-arm-vfp-hflt  
cmake -DBUILD_SHARED_LIBS=OFF ..
```

6. Compilamos e instalamos, aunque en muchas guías nos indican que usemos el parámetro j8 o j4 , como estamos copilando en Raspberry tendremos que usar el comando make sin parámetros.

```
make  
sudo make install
```

Anexo 5. Explicación simplificada del Bus i2C

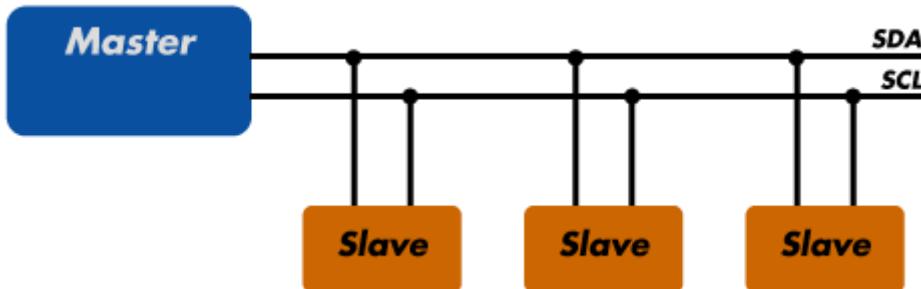


Ilustración 64 Representación Bus i2C

(<http://es.wikipedia.org/wiki/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 líneas para transmitir la información, una de ellas transporta los datos (SDA) y la otra transporta la señal de reloj (SCL) . Es necesaria una tercera línea utilizada única y exclusivamente como masa (GND).

Los dispositivos conectados por I2C tienen 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.

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.

Anexo 5. Calculo de distancia con trigonometría

El cálculo de distancias mediante una cámara es relativamente sencillo puesto que disponemos de dos de los datos más 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 fórmulas de trigonometría.

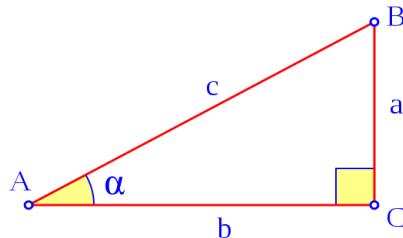


Ilustración 65 Esquema trigonometría

Calculo de la hipotenusa: $c = \frac{b}{\cos\alpha}$

Calculo de la altura: $a = c \cdot \operatorname{sen}\alpha$

Calculo de la distancia: $a = \frac{b \cdot \operatorname{sen}\alpha}{\cos\alpha} \rightarrow b = \frac{a \cdot \cos\alpha}{\operatorname{sen}\alpha}$

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 cálculo 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.

Anexo 6. Calculo de distancia con 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 línea base y es una de las variables que se usaran para el cálculo 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

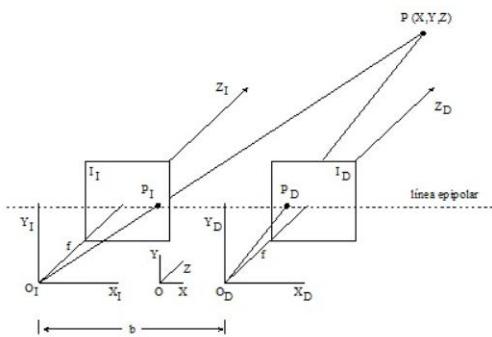


Ilustración 66 Geometria Epipolar

podemos definir que O es el origen de nuestras imágenes, f es la distancia focal y b es la línea 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 punto real en cada una de las imágenes.

Consideramos también las líneas Poi y Pod para la línea trazada desde el centro focal de las cámaras izquierda y derecha respectivamente, estas dos líneas 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.

$$O_i \cdot \frac{\frac{b}{2} + X}{Z} = \frac{X_i}{f} \quad O_d \cdot \frac{\frac{b}{2} - X}{Z} = \frac{X_d}{f}$$

$$\frac{X_d}{f} X_i = \frac{f}{Z} \cdot \left(X + \frac{b}{2} \right) X_d = \frac{f}{Z} \cdot \left(X - \frac{b}{2} \right) d = X_i - X_d = \frac{f \cdot b}{Z} \rightarrow Z = \frac{f \cdot b}{d}$$

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

Anexo 7. Sensor de distancia HC-SR04



Tech Support: services@elecfreaks.com

Ultrasonic Ranging Module HC - SR04

Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The module includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:

- (1) Using IO trigger for at least 10us high level signal,
- (2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
- (3) If the signal back, through high level, time of high output IO duration is the time from sending ultrasonic to returning.

Test distance = (high level time×velocity of sound (340M/S) / 2,

Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

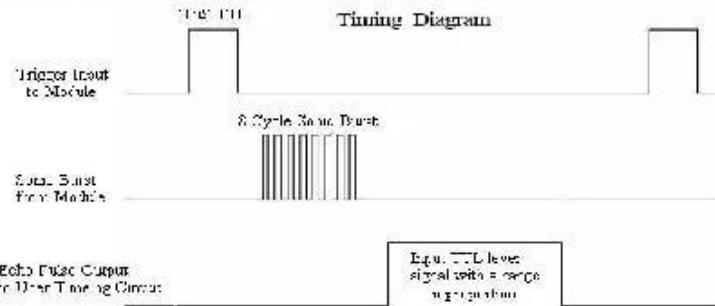
Electric Parameter

Working Voltage	DC 5 V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
MeasuringAngle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15mm



Timing diagram

The Timing diagram is shown below. You only need to supply a short 10 μ s pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion. You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: $uS / 58 = \text{centimeters}$ or $uS / 148 = \text{inch}$; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



Attention:

- The module is not suggested to connect directly to electric, if connected electric, the GND terminal should be connected the module first, otherwise, it will affect the normal work of the module.
- When tested objects, the range of area is not less than 0.5 square meters and the plane requests as smooth as possible, otherwise ,it will affect the results of measuring.

www.ElecFreaks.com



Anexo 8 BattBorg

<https://www.piborg.org/battborg>

The BattBorg is a power converter for your Raspberry Pi which allows you to power the Raspberry Pi off batteries. It will work with most batteries/battery packs that are between 7-36V so it's great for 12V car batteries, 8xAA battery packs, and so on. We're including an AA battery holder in two of the kits as rechargeable AA's are inexpensive, and readily available at most shops, and Ebay etc.

The BattBorg uses a highly efficient (~90%) OKI78SR DCDC converter which has a maximum 1.5A output and due to its efficiency, requires no heatsink.



Ilustración 67 Vista inferior Battborg

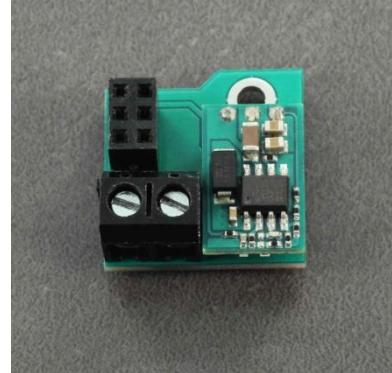


Ilustración 68 Vista superior BattBorg



Akami OKI-78SR Series

Fixed Output 1.5 Amp SIP DC/DC Converters



FEATURES

- Ultra wide 7 to 36 VDC input range
- Fixed Outputs of 3.3 or 5 VDC up to 1.5 Amps
- Vertical or horizontal SIP-mount, small footprint package
- "No heat sink" direct replacement for 3-terminal 78xx-series linear regulators
- High efficiency with no external components
- Short circuit protection
- Outstanding thermal derating performance
- UL/EN/IEC 60950-1, 2nd Edition safety approvals

PRODUCT OVERVIEW

Fabricated on a 0.41 by 0.65 inch (10.4 by 16.5 mm) Single Inline Package (SIP) module, the OKI-78SR series are non-isolated switching regulator (SR) DC/DC power converters for embedded applications. The fixed single output converters offer both tight regulation and high efficiency directly at the power usage site and are a direct plug-in replacement for TO-220 package 78xx series linear regulators. Typically, no extra outside components are required.

Two nominal output voltages are offered (3.3 and 5 VDC), each with 1.5 Amp maximum output. Based on fixed-frequency buck switching topology, the high efficiency means very low heat and little electrical noise, requiring no external components. The ultra wide input range is 7 to 36 Volts DC.

Protection features include short circuit current limit protection. The OKI-78SR is designed to meet all standards approvals. RoHS-6 (no lead) hazardous material compliance is specified as standard.

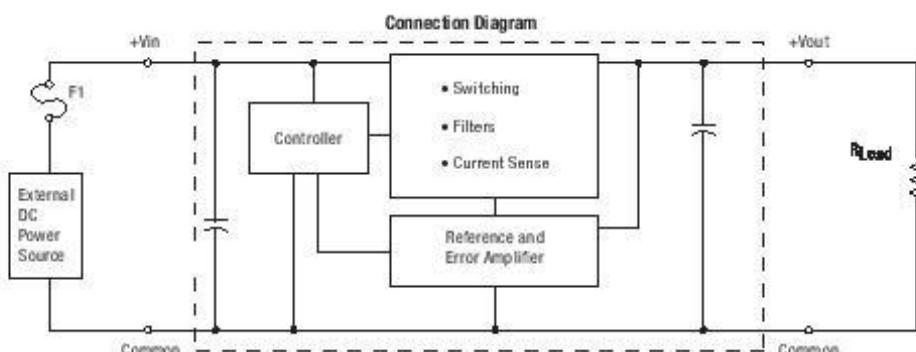


Figure 1. OI0-78SR

Note: Murata Power Solutions strongly recommends an external input fuse, F1.
See specifications.



For full details go to
www.murata-ps.com/rohs



www.murata-ps.com/support

MDC_OKI-78SR-W06.001 Page 1 of 12

Anexo 9. Clasificadores Haar

Puesto que una gran parte del proyecto está basado 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 sería necesario hacer cientos de imágenes en las que mostraremos el objeto que queremos buscar sobre un fondo y necesitaríamos a su vez cientos de fondos, generalmente se suele necesitar una proporción de una imagen positivas por cada dos negativas.

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

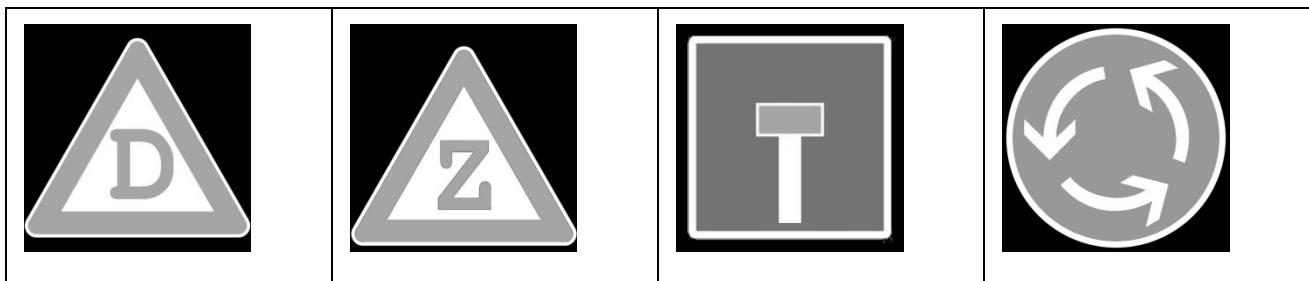
El primer programa que necesitaremos será opencv_createsamples este programa nos permitirá crear las imágenes de muestra desde un repositorio de imágenes de fondo, por ejemplo <http://tutorial-haartraining.googlecode.com/svn/trunk/data/negatives/>, que podemos descargar de internet y nuestro objetos 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 "neg*.jpg" http://tutorial-haartraining.googlecode.com/svn/trunk/data/negatives/
cd negativeImageDirectory
ls -1l *.jpg > negatives.txt
```

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



```
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
```

No hay una regla fija sobre cuánto puede costar el procesamiento de las imágenes pues depende de la complejidad de nuestras imágenes objetivo, se ha detectado que es mucho más eficaz ejecutar pequeños procesamientos de 200 imágenes en pequeñas tandas de etapas 5 o 6 , que intentar procesar todas las imágenes obtenidas en una sola orden.

Anexo 10 DataSheet L298N



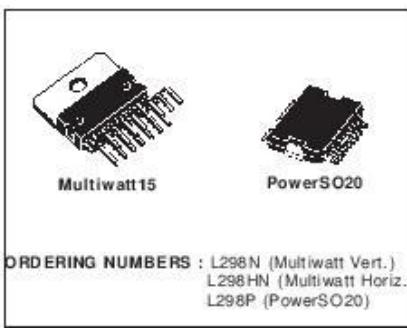
L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERRATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)

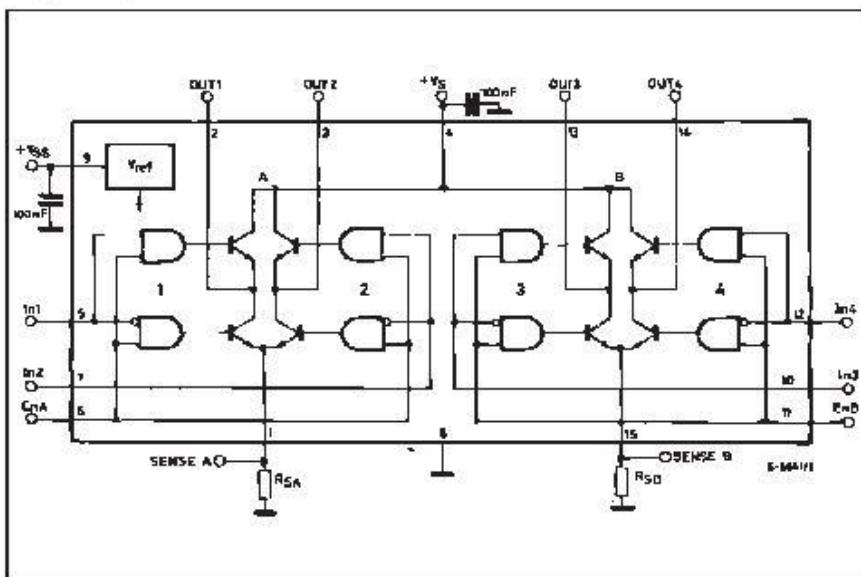
DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.



ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

BLOCK DIAGRAM

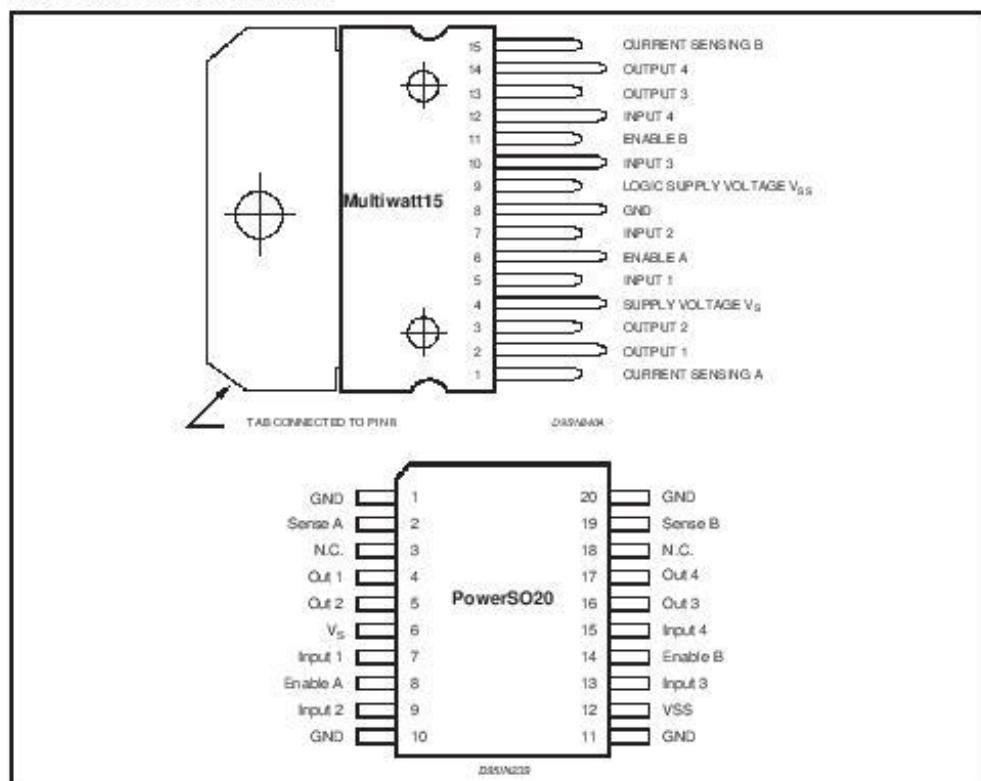


L298

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_i, V_{en}	Input and Enable Voltage	-0.3 to 7	V
I_O	Peak Output Current (each Channel)		
	- Non Repetitive ($t = 100\mu s$)	3	A
	- Repetitive (80% on -20% off; $t_{on} = 10ms$)	2.5	A
	- DC Operation	2	A
V_{sens}	Sensing Voltage	-1 to 2.3	V
P_{tot}	Total Power Dissipation ($T_{case} = 75^\circ C$)	25	W
T_{op}	Junction Operating Temperature	-25 to 130	°C
T_{sg}, T_j	Storage and Junction Temperature	-40 to 150	°C

PIN CONNECTIONS (top view)



THERMAL DATA

Symbol	Parameter	PowerSO20	Multiwatt 15	Unit
$R_{thj-case}$	Thermal Resistance Junction-case	Max.	—	°C/W
$R_{thj-amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	°C/W

(*) Mounted on aluminum substrate

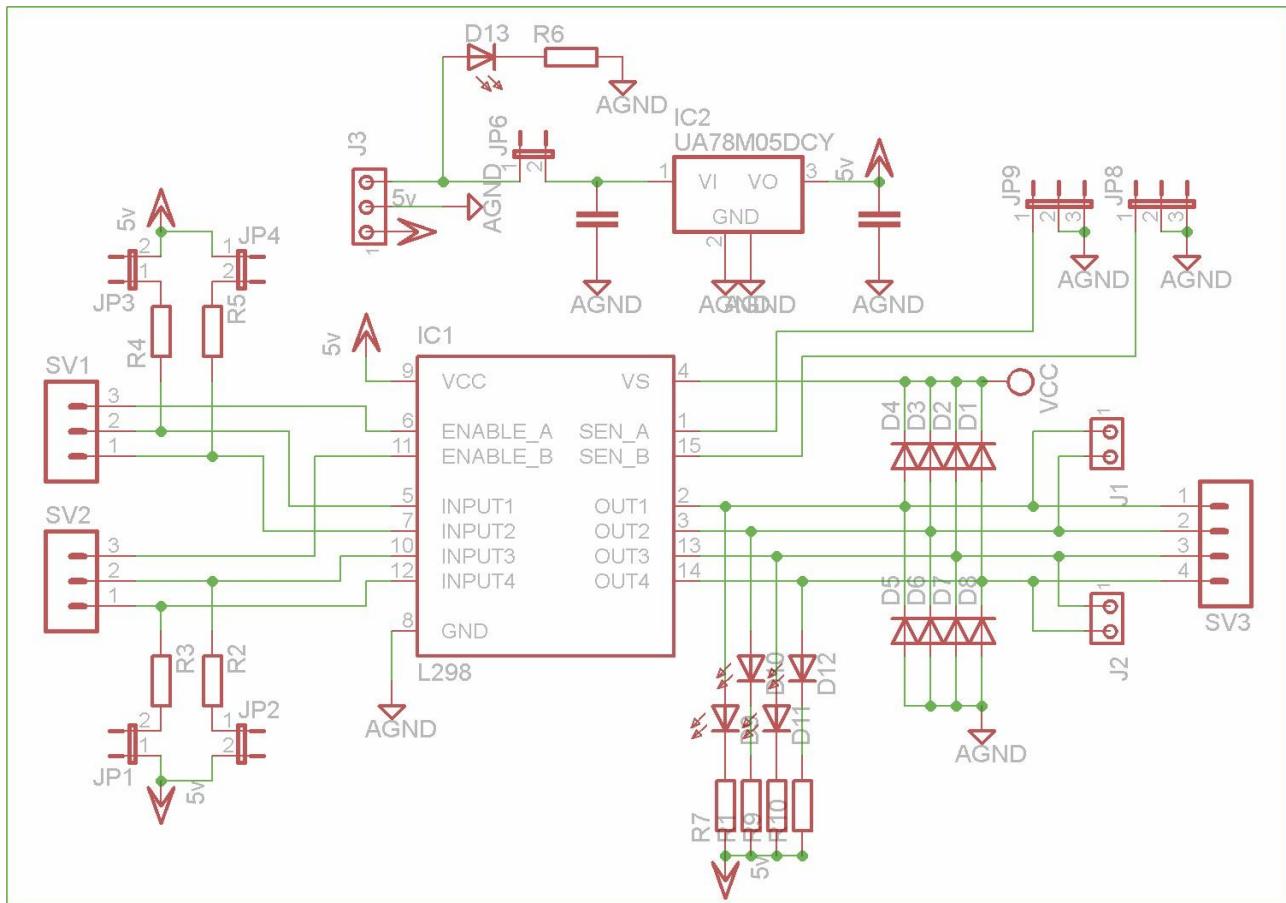
L298

PIN FUNCTIONS (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V _S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	V _{SS}	Supply Voltage for the Logic Blocks. A 100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
—	3;18	N.C.	Not Connected

ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V _S	Supply Voltage (pin 4)	Operative Condition	V _H +2.5		46	V
V _{SS}	Logic Supply Voltage (pin 9)		4.5	5	7	V
I _S	Quiescent Supply Current (pin 4)	V _{in} = H; I _L = 0 V _i = L V _i = H		13 50	22 70	mA mA
		V _{in} = L V _i = X			4	mA
I _{SS}	Quiescent Current from V _{SS} (pin 9)	V _{in} = H; I _L = 0 V _i = L V _i = H		24 7	36 12	mA mA
		V _{in} = L V _i = X			6	mA
V _L	Input Low Voltage (pins 5, 7, 10, 12)		-0.3		1.5	V
V _H	Input High Voltage (pins 5, 7, 10, 12)		2.3		V _{SS}	V
I _L	Low Voltage Input Current (pins 5, 7, 10, 12)	V _i = L			-10	µA
I _H	High Voltage Input Current (pins 5, 7, 10, 12)	V _i = H ≤ V _{SS} -0.6V		30	100	µA
V _{en} = L	Enable Low Voltage (pins 6, 11)		-0.3		1.5	V
V _{en} = H	Enable High Voltage (pins 6, 11)		2.3		V _{SS}	V
I _{en} = L	Low Voltage Enable Current (pins 6, 11)	V _{en} = L			-10	µA
I _{en} = H	High Voltage Enable Current (pins 6, 11)	V _{en} = H ≤ V _{SS} -0.6V		30	100	µA
V _{CEsat(H)}	Source Saturation Voltage	I _L = 1A I _L = 2A	0.95 2	1.35 2.7	1.7 2.7	V
V _{CEsat(L)}	Sink Saturation Voltage	I _L = 1A (5) I _L = 2A (5)	0.85	1.2 1.7	1.6 2.3	V
V _{Csat}	Total Drop	I _L = 1A (5) I _L = 2A (5)	1.80		3.2 4.9	V
V _{sens}	Sensing Voltage (pins 1, 15)		-1 (1)		2	V



Anexo 11. Contenido del CD

1. Carpeta DFD : Diagramas de flujo de datos del proyecto.
2. Carpeta Documentación: Anexos y documentos del proyecto
3. Carpeta imágenes: Imágenes usadas en el desarrollo del proyecto
4. Carpeta Software Proyecto: código fuente desarrollado
 - a. DetectorCirculos : programa para detectar features Haar.
 - b. DuxmanCar: Programa de control RaspberryPi
 - c. TraductorDuxBW: programacontrol Android
5. Carpeta Software adicional utilizado:Software adicional usado en el desarrollo del proyecto