



**UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA**

GRADO EN INGENIERÍA INFORMÁTICA

**TECNOLOGÍA ESPECÍFICA DE
INGENIERÍA DE COMPUTADORES**

TRABAJO FIN DE GRADO

**SISTEMA DE CONTROL VISUAL
EMPOTRADO UTILIZANDO FPGA PARA
DETECCIÓN DE OBSTÁCULOS**

Jose Jiménez Rincón

Julio, 2017

**SISTEMA DE CONTROL VISUAL
EMPOTRADO UTILIZANDO FPGA PARA
DETECCIÓN DE OBSTÁCULOS**



**UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA**

Tecnologías y Sistemas de Información

**TECNOLOGÍA ESPECÍFICA DE
INGENIERÍA DE COMPUTADORES**

TRABAJO FIN DE GRADO

**SISTEMA DE CONTROL VISUAL
EMPOTRADO UTILIZANDO FPGA PARA
DETECCIÓN DE OBSTÁCULOS**

Autor: Jose Jiménez Rincón

Director: Dr. Julio Dondo Gazzano

Julio, 2017

Jose Jiménez Rincón

Ciudad Real – Spain

E-mail: josejimrin@gmail.com

Teléfono: 685 326 422

© 2017 Jose Jiménez Rincón

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Se permite la copia, distribución y/o modificación de este documento bajo los términos de la Licencia de Documentación Libre GNU, versión 1.3 o cualquier versión posterior publicada por la *Free Software Foundation*; sin secciones invariantes. Una copia de esta licencia esta incluida en el apéndice titulado «GNU Free Documentation License».

Muchos de los nombres usados por las compañías para diferenciar sus productos y servicios son reclamados como marcas registradas. Allí donde estos nombres aparezcan en este documento, y cuando el autor haya sido informado de esas marcas registradas, los nombres estarán escritos en mayúsculas o como nombres propios.

TRIBUNAL:

Presidente:

Vocal:

Secretario:

FECHA DE DEFENSA:

CALIFICACIÓN:

PRESIDENTE

VOCAL

SECRETARIO

Fdo.:

Fdo.:

Fdo.:

Resumen

El presente Trabajo Final de Grado (TFG) aborda la problemática de utilizar hardware reconfigurable para mejorar el rendimiento de un sistema encargado de esquivar objetos en tiempo real con un vehículo remoto de forma autónoma.

En este documento se presenta toda la información necesaria acerca de cómo se ha orientado el proyecto; que subobjetivos se han planteado para lograr el objetivo principal del TFG; cómo se han pensado e implementado los algoritmos utilizados en el proyecto; se proporciona una descripción de las herramientas que se han utilizado en el sistema. Además se incluye información acerca del escenario actual en el que se encuentra el mundo de los vehículos autónomos, qué problemas se deben tener en cuenta a la hora de desarrollar un vehículo autónomo y se proporciona una breve introducción al uso de herramientas para trabajar con hardware reconfigurable.

Abstract

This project addresses the problem of using reconfigurable hardware to improve the performance of a system in charge of dodging objects in real time with a remote vehicle autonomously.

This document presents all necessary information about how the project has been oriented; What sub-objectives have been raised to achieve the main objective of TFG; How the algorithms used in the project have been thought and implemented; A description of the tools that have been used in the system is provided. It also includes information about the current scenario in which the world of autonomous vehicles is located, what problems must be taken into account when developing a standalone vehicle and provides a brief introduction to the use of tools to work with reconfigurable hardware .

Agradecimientos

EN primer lugar debo dar las gracias a mi tutor Julio por la confianza que me ha prestado y por hacer posible el hecho de convertir una idea difusa de proyecto en realidad. Nunca pensé que tendría un tutor argentino ni que me gustaría tanto el chimichurri, el mundo es curioso. Gracias a todos mis compañeros de laboratorio de los cuales he aprendido muchas cosas. Sinteticemos experiencias. Un gaditano debería haber invertido en Bitcoin, Codor se acerca. Menos mal que la música está muy alta y no se oye venir. Ahí va eso. Vuelvo para dar las gracias a mis compañeros de piso Juan Carlos, Víctor y Alberto. Sin su ayuda todavía estaría estudiando para sacarme las asignaturas... Otra gente importante como Julia, Patricio y Roberto me ayudaron a pasármelo bien. Son gente respetable también por ayudarme a pasármelo bien... ¡No todo es estudiar!

Muchas gracias a mi hermana Ana de la cual me siento más orgulloso cada día aunque me traumatizara de pequeño, a mi padre Jose el cual es un ejemplo de superación y en especial a mi madre María la cual ha invertido mucho tiempo de su vida en educar y formar la persona que soy hoy en día, te lo debo todo y espero que estés orgullosa de mí. Aunque te lo haya puesto difícil siempre has estado ahí. Me siento afortunado.

Y por último y no menos importante a mi chica María Rosa. La cual ha sido mi suerte y mi apoyo en todos los malos momentos que he vivido durante estos dos últimos años. Te quiero guapa.

Como reflexión final me gustaría introducir un refrán que me ha marcado:

«Entre broma y broma, la verdad se asoma.»

Jose

A mi madre María

Índice general

Resumen	V
Abstract	VII
Agradecimientos	IX
Índice general	XIII
Índice de cuadros	XVII
Índice de figuras	XIX
Índice de listados	XXI
Listado de acrónimos	XXIII
1. Introducción	1
1.1. Estructura del documento	2
2. Objetivos	3
2.1. Objetivo principal	3
2.2. Objetivos específicos	3
2.2.1. Detección de obstáculos	4
2.2.2. Distinción y cálculo de posición del vehículo	4
2.2.3. Cálculo de trayectoria	5
2.2.4. Generación de movimientos mediante datos de la trayectoria	5
2.2.5. Comunicación con el vehículo	5
2.2.6. Movimiento del vehículo	5
2.2.7. Funcionamiento del sistema en tiempo real	6
3. Antecedentes	7
3.1. Coches autónomos	7

3.1.1. Situación actual de los coches autónomos	7
3.1.2. Factores a tener en cuenta para el desarrollo de un coche autónomo	9
3.2. Hardware reconfigurable	13
4. Metodología de la gestión del proyecto	17
4.1. Metodología SCRUM	18
4.1.1. Roles	18
4.1.2. ¿Como funciona SCRUM?	19
4.1.3. Ventajas	20
4.1.4. Desventajas	21
4.1.5. ¿Porqué se elige la metodología Scrum para este TFG?	21
4.1.6. Herramientas utilizadas para la resolución del proyecto	22
5. Desarrollo	25
5.1. Tratamiento de la información de una imagen	26
5.1.1. Cabecera del formato BMP	26
5.2. Algoritmo para detectar cambios en el entorno	27
5.3. Delimitación de los obstáculos	28
5.3.1. ¿Qué punto corresponde a cada objeto?	29
5.3.2. ¿Cómo guardar la información de forma coherente?	30
5.3.3. ¿Distancia entre objetos \leq tamaño del vehículo?	31
5.4. Implementación del algoritmo A* en C++, sin restricciones	32
5.5. Detección del vehículo frente a los objetos	33
5.6. Diseño del vehículo del sistema	34
5.7. Comunicación con el vehículo	35
5.7.1. Protocolo de comunicación	35
5.7.2. Ejemplo de funcionamiento del protocolo de comunicación	36
5.7.3. Maqueta de pruebas del protocolo de comunicación	39
5.8. Desarrollo del firmware para el control del vehículo	41
5.9. Integración de los algoritmos en el sistema	41
5.10. Optimizaciones realizadas en el sistema	44
5.10.1. Dilatación de obstáculos mediante <i>OpenCV</i>	44
5.10.2. Tratamiento de la información de una imagen en varios formatos	45
5.10.3. Segmentación de la imagen en macro-bloques	46
5.10.4. Implementación algoritmo A* según restricciones de <i>Vivado HLS</i>	47
5.11. Integración del sistema en la placa <i>Zedboard</i>	53

6. Resultados	55
6.1. Detección de objetos	55
6.2. Detección del vehículo frente a los objetos	55
6.3. Modificación algoritmo A* [FB310]	55
6.4. Comunicación	56
6.5. Coordinador de los movimientos del vehículo	56
6.6. Integración del sistema	57
6.7. Optimizaciones realizadas	57
6.7.1. Tratamiento de imágenes en varios formatos	58
6.7.2. Detección de obstáculos con <i>OpenCV</i>	58
6.7.3. Segmentación de la imagen en macrobloques	58
6.7.4. A* optimizado para <i>Vivado HLS</i>	58
6.8. Integración del sistema en la placa <i>Zedboard</i>	58
7. Conclusiones	61
7.1. Objetivos logrados	61
7.2. Trabajo futuro	61
A. Componentes que forman el vehículo	65
A.1. <i>Arduino</i> UNO R3 ATMEGA328	65
A.2. Controlador <i>Sparkfun</i> Ardumoto y chasis	65
A.3. Módulos <i>Xbee</i> de la marca <i>Digi</i>	65
A.4. Batería recargable <i>Odec</i> 9V	66
A.5. Osciloscopio y multímetro	66
A.6. Placa de pruebas para <i>Arduino</i>	67
A.7. <i>Arduino shield</i>	68
A.8. Sensor efecto Hall A3144	68
A.9. Estación de soldadura <i>JBM</i> AR5800	69
A.10. Impresión de marca identificativa para el coche	69
B. Código de los algoritmos del sistema	71
B.1. Cabecera del formato Bits Maps Protocole (BMP)	71
B.2. Detección de objetos	73
B.3. Implementación A* para la herramienta <i>Vivado HLS</i>	78
B.4. Detección del vehículo frente a los objetos	84
B.5. Comunicación puerto serie	86

B.6. Coordinador de los movimientos del vehículo	89
B.7. Dilatación de obstáculos mediante <i>OpenCV</i>	94
C. GNU Free Documentation License	97
C.0. PREAMBLE	97
C.1. APPLICABILITY AND DEFINITIONS	97
C.2. VERBATIM COPYING	98
C.3. COPYING IN QUANTITY	98
C.4. MODIFICATIONS	99
C.5. COLLECTIONS OF DOCUMENTS	100
C.6. AGGREGATION WITH INDEPENDENT WORKS	100
C.7. TRANSLATION	100
C.8. TERMINATION	100
C.9. FUTURE REVISIONS OF THIS LICENSE	101
C.10. RELICENSING	101
Referencias	103

Índice de cuadros

5.1. Cabecera formato BMP	27
5.2. Tiempos de ejecución en software	43
5.3. Estructura de datos de cada nodo del mapa.	48
6.1. Tiempos de ejecución del sistema sin optimizar	57
6.2. Tiempos de ejecución del sistema optimizado	58

Índice de figuras

2.1. Estados del entorno del proyecto	4
2.2. Ejemplo de detección de la posición del vehículo en el entorno.	4
2.3. Ejemplo del Cálculo de trayectoria	5
3.1. Ejemplo de funcionamiento de las técnicas enumeradas	10
3.2. Ejemplos de algoritmos para la detección del vehículo	11
4.1. Roles de Scrum para este TFG	22
5.1. Ejemplo resta de imágenes	28
5.2. Ejemplo de la delimitación de los obstáculos	31
5.3. Casos de prueba con 4 Direcciones	32
5.4. Casos de prueba con 8 Direcciones	32
5.5. Ejemplo de uso con las funciones <i>cvtColor</i> y <i>inRange</i>	33
5.6. Uso de las funciones <i>cvtColor</i> , <i>inRange</i> y <i>HoughCircles</i>	34
5.7. Diagrama de secuencia del protocolo de comunicación	37
5.8. Maqueta de pruebas del protocolo de comunicación	40
5.9. Ejemplo de dilatación de objetos	44
5.10. Ejemplo de segmentación de imagen en macro-bloques	46
6.1. Ejemplo de aplicación del algoritmo de detección de objetos	55
6.2. Ejemplo de detección de la posición del vehículo en el entorno.	56
6.3. Ejemplo de cálculo de trayectoria	56
6.4. Comunicación por el puerto serie	56
6.5. Ejemplo de envío de movimientos al vehículo	57
6.6. Escenario con el que se ha realizado la medición de los tiempos	57
6.7. Periodo de reloj estimado para el algoritmo A*	59
6.8. Latencias para cada bucle del algoritmo A*	59
6.9. Recursos necesarios para la ejecución del algoritmo A*	59
A.1. Ardumoto	66

A.2. Módulo de comunicación del sistema	66
A.3. Placa base	67
A.4. Ardumoto instalado sobre <i>Arduino UNO</i>	68
A.5. Sensor Efecto Hall A3144	69

Índice de listados

5.1.	Parte del algoritmo que calcula los nodos adyacentes	49
5.2.	Función con la que se calcula la heurística	52
5.3.	Función con la que se calcula el mejor nodo	52
B.1.	Cabecera del formato BMP	71
B.2.	Detección de objetos	73
B.3.	Implementación A* para la herramienta <i>Vivado HLS</i>	78
B.4.	Detección del vehículo frente a los objetos	84
B.5.	Comunicación puerto serie	86
B.6.	Coordinador de los movimientos del vehículo	89
B.7.	Dilatación de obstáculos mediante OpenCV	94

Listado de acrónimos

ACK	Acknowledgement
ARM	Acorn RISC Machine
BFS	Breadth First Search
BMP	Bits Maps Protocole
CCPM	Critical Chain Project Management
CMMI	Capability Maturity Model Integration
FPGA	Field Programmable Gate Array
HLS	High-Level Synthesis
HSV	Hue Saturation Value
IDE	Integrated Development Environment
IP	Intellectual Property
JPEG	Joint Photographic Experts Group
JPG	Joint Photographic Graphics
KB	Kilobyte
MB	Megabyte
MHz	Megahercio
PMBOK	Project Management Body of Knowledge
PNG	Portable Network Graphics
PMI	Project Management Institute
RGB	Red-Green-Blue
ROI	Return of Investmen
RTL	Register Transfer Levelç
SAE	Sociedad de Ingenieros Automotrices
SDK	Software Development Kit
SEI	Software Engineering Institute
SMART	Specific, Measurable, Achievable, Relevant and Time-bound

TFG	Trabajo Final de Grado
TIFF	Tagged Image File Format
UML	Unified Modeling Language
VHDL	VHSIC Hardware Description Language
WIP	Work In Progress

Capítulo 1

Introducción

En este TFG se pretende aportar una solución a la problemática existente en la detección de obstáculos en tiempo real con vehículos autónomos para entornos cerrados. Para acotar un poco el problema se desea conseguir que un vehículo autónomo consiga llegar de un lugar a otro esquivando los obstáculos que aparezcan en su entorno en tiempo real. Esquivar objetos en tiempo real implica que si aparece un nuevo obstáculo en la trayectoria que se ha calculado anteriormente, el sistema será capaz de recalcular la ruta para que el automóvil no colisione y consiga llegar al destino.

Para la detección de obstáculos se utilizará una cámara cenital, de tal manera que, procesando la imagen obtenida se puedan detectar los obstáculos y determinar la trayectoria a realizar por el vehículo.

Dado que uno de los requisitos es que el vehículo debe ser autónomo, el sistema a desarrollar será el encargado de indicar al coche qué dirección debe tomar para poder llegar desde un punto de origen a un punto de destino. En nuestro sistema, el procesamiento de la imagen y el cálculo de la trayectoria serán realizados por un sistema empotrado híbrido, formado por dos procesadores ARM (Acorn RISC Machine) y una FPGA (Field Programmable Gate Array), que será el encargado de calcular y enviar la información al automóvil utilizando para ello transmisión inalámbrica.

Con la información que tenemos hasta ahora podemos observar los siguientes elementos principales en nuestro sistema:

- Una **cámara cenital** se encuentra conectada por cable al computador y se utilizará como herramienta para obtener los datos que procesaremos posteriormente.
- Un **computador**, con hardware reconfigurable, procesará la información obtenida por la cámara, generará una trayectoria y la traducirá en movimientos que se transmitirán de forma inalámbrica al vehículo.
- Un **coche** a la espera de recibir información con el fin de realizar los movimientos necesarios para llegar desde su posición actual a un punto de destino.

1.1 Estructura del documento

A continuación se va a realizar una breve descripción de los capítulos que se van a abordar en este TFG:

Capítulo 2: Objetivos

Se concretarán y expondrán los objetivos principales y específicos del TFG.

Capítulo 3: Antecedentes

Paradigma de los vehículos autónomos y hardware reconfigurable.

Capítulo 4: Metodología de la gestión del proyecto

Metodología y herramientas utilizadas para el desarrollo del proyecto.

Capítulo 5: Desarrollo

Resolución de los objetivos planteados en este TFG.

Capítulo 6: Resultados

Muestra y comparación de los resultados obtenidos en el proyecto.

Capítulo 7: Conclusiones

Análisis de los resultados y posibles mejoras.

Capítulo 2

Objetivos

En primer lugar se va a exponer el objetivo principal del TFG. Y en segundo lugar qué subobjetivos se han planteado de tal forma que se pueda seguir una buena metodología de trabajo debido a la correcta modularización del proyecto.

2.1 Objetivo principal

El objetivo principal del proyecto es conseguir que un vehículo llegue desde un punto de origen a un punto de destino esquivando los obstáculos que aparezcan en el escenario en tiempo real.

Para llegar a cumplir este objetivo es necesario plantear distintos subobjetivos de tal forma que se cubran todas las necesidades para llevar a cabo nuestro proyecto.

2.2 Objetivos específicos

Como objetivos específicos del sistema podemos encontrar los siguientes:

Subobjetivo 2.2.1: Detección de obstáculos

Distinguir los obstáculos del entorno.

Subobjetivo 2.2.2: Distinción y cálculo de posición del vehículo

Distinguir el vehículo frente a los demás objetos y calcular su posición en el entorno.

Subobjetivo 2.2.3: Cálculo de trayectoria

Calcular la trayectoria por la que debe avanzar el vehículo.

Subobjetivo 2.2.4: Generación de movimientos mediante datos de la trayectoria

Traducción de los datos de la trayectoria en movimientos.

Subobjetivo 2.2.5: Comunicación con el vehículo

Comunicar el coche con el sistema.

Subobjetivo 2.2.6: Movimiento del vehículo

Realizar movimientos con el vehículo de manera inalámbrica y automática.

Subobjetivo 2.2.7: Funcionamiento del sistema en tiempo real

Conseguir que el sistema funcione en tiempo real.

2.2.1 Detección de obstáculos

Se deben detectar los obstáculos que aparezcan en el escenario en función del tiempo. Teniendo en cuenta que el funcionamiento del sistema está pensado para que el coche esquive objetos en tiempo real, se necesita implementar un algoritmo que detecte los obstáculos que aparecen y desaparecen en el entorno. Es decir, el escenario sobre el que se desarrolla la ejecución del proyecto es fijo. Por lo que este algoritmo debe comprobar si se ha producido algún cambio en el entorno para delimitar la posición de los objetos.



Figura 2.1: Estados del entorno del proyecto

2.2.2 Distinción y cálculo de posición del vehículo

Se requiere un algoritmo que detecte la posición del vehículo en el escenario para poder introducir las coordenadas desde las que comenzar el cálculo de la trayectoria que debe seguir el vehículo. Para ello, se debe detectar un rasgo identificativo que diferencie al coche de los demás objetos.

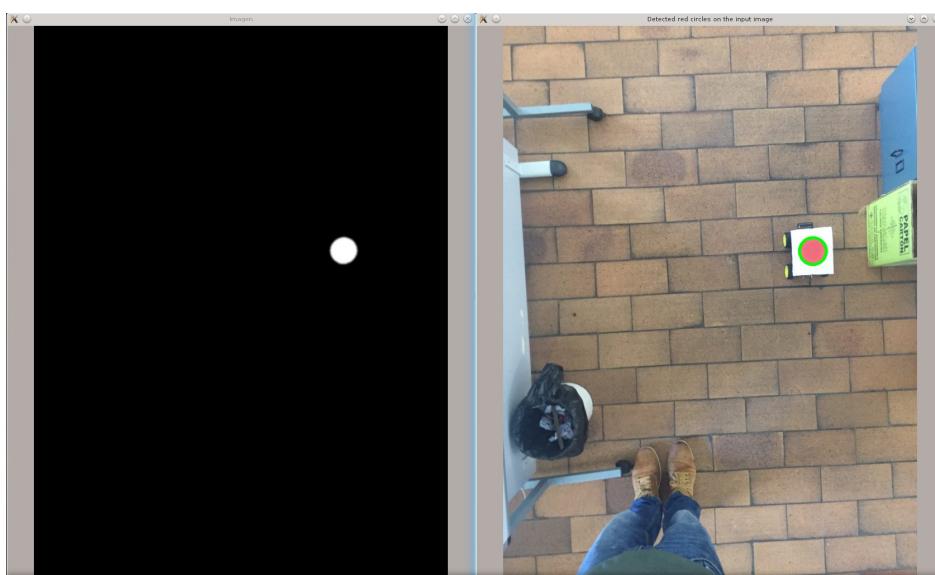


Figura 2.2: Ejemplo de detección de la posición del vehículo en el entorno.

2.2.3 Cálculo de trayectoria

Es necesaria la inclusión de un algoritmo en el sistema que calcule la trayectoria que debe seguir el vehículo para evitar colisionar con los posibles obstáculos del entorno. Como este proyecto está orientado al funcionamiento en tiempo real, el sistema deberá calcular una nueva ruta cada vez que el entorno cambie. Por ello, es necesario que el algoritmo no demore mucho tiempo.

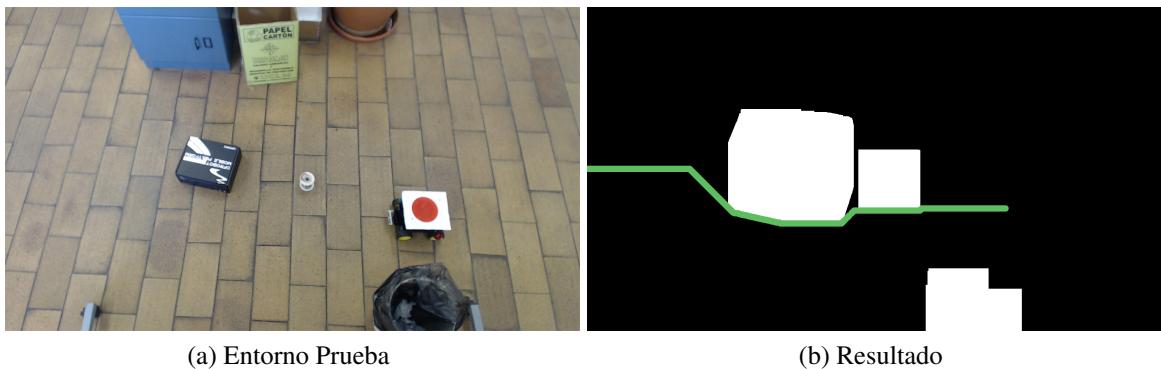


Figura 2.3: Ejemplo del Cálculo de trayectoria

2.2.4 Generación de movimientos mediante datos de la trayectoria

Se requiere un algoritmo que interprete los resultados generados por el cálculo de trayectoria y coordine los movimientos que debe ejercer el vehículo para llegar desde el origen al destino. Los movimientos se comunicarán de forma segmentada para controlar si los movimientos se realizan correctamente. Si se produce un cambio en el entorno, el coche deberá permanecer parado hasta que se calcule la nueva trayectoria.

2.2.5 Comunicación con el vehículo

Se necesita establecer una comunicación entre el coche y el computador del sistema para que el vehículo reciba la información correspondiente a los movimientos. Para asegurar que se sigue el camino correcto, el vehículo transmitirá mensajes de confirmación de los movimientos que vaya realizando.

2.2.6 Movimiento del vehículo

Dado que el vehículo recibirá la información correspondiente al movimiento que debe realizar, este debe extraer la información y ejecutar los movimientos de los motores en función de la orientación y la distancia que deba recorrer. Es importante que la dirección la tome con precisión, ya que de no ser así el coche podría seguir una trayectoria errónea aunque el sistema la hubiera calculado correctamente.

2.2.7 Funcionamiento del sistema en tiempo real

Una vez se han conseguido todos los subobjetivos comentados anteriormente, se deben integrar las funcionalidades en el sistema de tal forma que se permita esquivar objetos en tiempo real. La velocidad del vehículo no es importante en este proyecto, por lo que se debe priorizar en detectar los cambios en el escenario de forma rápida y calcular la trayectoria sin que se demore mucho tiempo en el cálculo.

Capítulo 3

Antecedentes

Es muy importante conocer un mínimo de información acerca de las áreas de trabajo que se pretenden abordar para la realización de un proyecto. Ya que el conocimiento es poder, y el hecho de leer sobre el trabajo que ha dedicado nuestra civilización previamente al nuestro, nos puede ayudar a evitar errores que en su día fueron quebraderos de cabeza para otros.

Sección 3.1: Coches autónomos

Complejidad y factores a tener en cuenta para el desarrollo de un vehículo autónomo.

Sección 3.2: Hardware reconfigurable

Ventajas y desventajas de la utilización de hardware reconfigurable.

3.1 Coches autónomos

El hecho de que un sistema sea capaz de hacer llegar un vehículo de un lugar a otro sin poner en riesgo la integridad del entorno, y de sí mismo, es complejo. Ya que hay infinidad de sucesos que pueden ocurrir dando lugar a una colisión. Sin tener en cuenta otros sucesos como por ejemplo que el vehículo no llegue al destino correcto o tome una ruta que sea más larga de lo que debería. Es por ello, que al ser tan extenso el número de posibilidades que afronta este sistema es necesario acotar el problema. Y aún así, el número de factores externos e internos que hay que tener en cuenta son muchos.

En esta sección se va a tratar de poner en situación al lector sobre la situación actual de los coches autónomos para comprender el porqué se ha decidido realizar este proyecto y de los factores que se deben tener en cuenta para el desarrollo de un coche autónomo que cumpla los objetivos de este TFG.

3.1.1 Situación actual de los coches autónomos

Hoy en día las compañías automovilísticas están introduciendo funciones autónomas en sus coches. Un claro ejemplo son los coches que aparcan solos o que circulan de manera prácticamente autónoma por autopista.

Hay que tener en cuenta que se pueden diferenciar distintos niveles de autonomía en un coche, ya que no tiene la misma complejidad conseguir que un vehículo aparque en ba-

teria frente a lograr una conducción totalmente autónoma por ciudad. Es por esto que la Sociedad de Ingenieros Automotrices (SAE) [SAEb] decidió estandarizar la *división en 6 niveles* [SAEa] de la capacidad de conducción autónoma de un vehículo:

1. **Nivel 0.** El coche no tiene ningún sistema automatizado que le permita tomar el control, sólo puede tener sistemas que emitan alguna advertencia.
2. **Nivel 1.** En este nivel los coches incluyen sistemas como el *control de crucero*¹ o la tecnología para mantener el coche en el carril.
3. **Nivel 2.** Aquí el vehículo puede denominarse *semiautónomo*. El conductor debe permanecer en alerta por si en algún momento tiene que tomar el control del coche ya que éste puede no responder adecuadamente y es obligatorio que el sistema se desactive cuando el conductor tome el control. Un ejemplo de coche semiautónomo es el *Mercedes-Benz Clase E* [Mer]. Está a la venta desde marzo de 2016 y destaca por el denominado *Drive Pilot* que es capaz de evitar la salida de la calzada sin la necesidad de que existan líneas de carril. Nunca un coche fabricado en serie fue capaz de guiarse por una carretera sin la ayuda de las líneas de carril hasta que salió este *Mercedes-Benz*.
4. **Nivel 3.** Los vehículos pueden circular de forma autónoma en entornos controlados como autopistas. En este nivel podría encontrarse el sistema *Autopilot* de *Tesla* en el *Model S* [Tes]. Es un sistema que está desactivado por defecto en el coche y que debe ser conectado voluntariamente por el conductor. Realiza constantes comprobaciones para asegurarse de que el conductor permanece atento y con las manos en el volante avisando mediante alertas sonoras y luminosas si no detecta las manos en él.
5. **Nivel 4.** Los coches autónomos pueden circular sin supervisión del conductor en áreas acotadas donde el coche tenga suficiente información para no depender del conductor. Algunas empresas muy potentes ya han comenzado a hacer sus primeras pruebas serias para ver cuál es el funcionamiento real de los coches autónomos. *Google* es un ejemplo de conducción autónoma; desde mayo de 2012 tiene licencia para probar coches autónomos en algunos estados de Estados Unidos. *Volvo* y *Uber* también se han aliado para crear la primera flota de taxis autónomos del mundo. Y otras marcas como *BMW*, *Citroën* o *Audi* también se han lanzado a las pruebas con los coches autónomos.
6. **Nivel 5.** La conducción autónoma en este nivel es completa. Puede circular por cualquier carretera o ciudad siempre y cuando sea legal la conducción autónoma. Gracias a la tecnología, el coche podrá reaccionar ante cualquier imprevisto. Y no sólo los fabricantes de automoción están interesados en crear coches autónomos, otras empresas

¹Sistema electrónico que permite fijar una velocidad de marcha que se mantiene sin necesidad de que el conductor mantenga pisado el acelerador. El sistema se desactiva cuando se pisa el freno. Con sólo pulsar el correspondiente botón se recupera automáticamente la velocidad previamente seleccionada. Los más modernos incorporan un radar en la parte delantera del coche, de forma que pueden controlar también de forma automática la distancia con el vehículo que circula delante.

tecnológicas también indagan en el tema. Por ejemplo, *Microsoft* está diseñando aplicaciones que puedan formar parte de estos vehículos [Mic]. La marca china *Baidu* ya prueba su coche [Bai], mientras *Apple* podría estar detrás de un proyecto de vehículo autónomo que, de momento, ofrece más rumores que realidades [App].

Después de conocer acerca de los seis niveles de autonomía, este sistema que se pretende desarrollar en este TFG emulará una autonomía de nivel 4. Esto es debido a que el sistema será capaz de analizar la información tomada y comunicará los movimientos pertinentes al vehículo de tal forma que no necesite la supervisión del ser humano. A continuación vamos a ver qué factores se deben tener en cuenta para el desarrollo de este TFG.

3.1.2 Factores a tener en cuenta para el desarrollo de un coche autónomo

En primer lugar hay que entender que el sistema no va a resolver ninguna situación que no se haya implementado. Dicho de esta forma suena muy evidente pero hay situaciones que al comienzo del planteamiento del problema no se tienen en cuenta. Esto es debido a que el mero hecho de hacer funcionar el sistema en escenarios distintos puede provocar que en alguno de ellos no funcione como se esperaba. Debido a cuestiones como la luz, la composición del suelo, las características meteorológicas, el tamaño del escenario, etc... Por ello, es importante definir el tipo de escenario en el cual se va a desarrollar el proyecto para estudiar las diferentes características del entorno a las cuales se va a enfrentar el vehículo. Una vez definido el escenario sobre el cual se va a desarrollar el cálculo de la trayectoria del vehículo, hay que realizar pruebas con el dispositivo que deseamos utilizar para tomar la información del escenario. Existen diferentes formas de tomar la información pero en este TFG se decidió utilizar una cámara desde el principio con el objetivo de que un futuro se utilizara hardware reconfigurable para optimizar el rendimiento. Por ello, se debe tener en cuenta que la información que obtenemos del entorno viene codificada en imágenes. Debido a que toda la información la tomamos de imágenes, debemos observar las diferentes situaciones que se pueden dar lugar en el entorno para tenerlas en cuenta en un futuro. Como por ejemplo, si la luz que existe en el escenario es constante o varía de una forma drástica dependiendo de la hora en la que se realicen las pruebas del sistema.

Finalizadas las pruebas pertinentes con la cámara, se debe pensar si van a aparecer objetos en el escenario, ya que no es lo mismo calcular una trayectoria en un escenario sin objetos o con objetos que debe tener en cuenta a la hora de calcular la trayectoria para evitar colisionar con estos. Si existen objetos en el escenario en el cual se va a desarrollar el proyecto, se debe tener en cuenta que el dispositivo que toma la información del escenario debe detectar la posición de todos los objetos del entorno para evitar que la trayectoria pase por las coordenadas en las que se encuentren posicionados los objetos. Debido a esto nos surge la problemática acerca de cómo diferenciamos los objetos del entorno.

Detección de objetos

Para detectar objetos en una imagen, primero debemos plantearnos cómo diferenciar los objetos del entorno en el que se encuentran. Existen diferentes técnicas para distinguir los bordes² de la imagen:

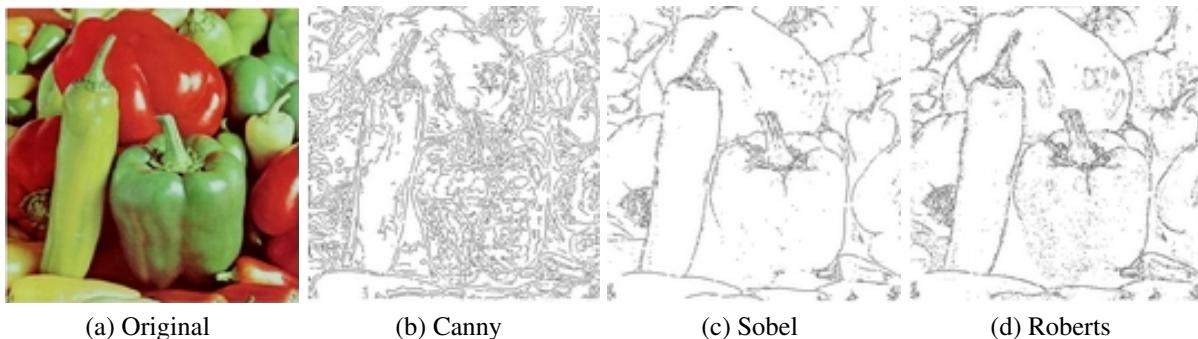


Figura 3.1: Ejemplo de funcionamiento de las técnicas enumeradas

1. Operador Roberts. Obtiene buena respuesta ante bordes diagonales. Ofrece buenas prestaciones en cuanto a localización. El gran inconveniente de este operador es su extremada sensibilidad al ruido³ y por tanto tiene pobres cualidades de detección. [Ope05]
2. Operador Prewitt. En el operador Prewitt se involucran a los vecinos de filas o columnas adyacentes para proporcionar mayor inmunidad al ruido. [Ope05]
3. Operador Sobel. El operador Sobel, se supone que es más sensible a los bordes diagonales que el de Prewitt aunque en la práctica hay poca diferencia entre ellos. [Ope05]
4. Algoritmo Canny. Es un algoritmo de múltiples fases para detectar un amplio rango de bordes. Es sin duda el operador más utilizado en la detección de bordes.[Can09]

Mediante la utilización de cualquiera de estas técnicas, figura 3.1, se pueden detectar los bordes de los objetos, solo falta elegir en función de las características de nuestro escenario. Por ejemplo, si nuestra cámara percibe mucho ruido y elegimos la técnica del operador Roberts obtendremos muy malos resultados ya que el operador Roberts tiene una extrema sensibilidad al ruido y por lo tanto tiene pobres cualidades de detección.

Delimitación de objetos

Obtenidos los resultados de la detección de objetos, se puede apreciar el inconveniente de que el algoritmo de detección de bordes detecta el vehículo como un objeto. Esto es un problema, ya que para calcular la trayectoria que tomará el vehículo es necesario saber

²Se hace referencia a que los bordes son el perímetro de los objetos que hay en la imagen.

³Se llama ruido a toda la información no deseada que se mezcla con la información útil que queremos procesar.



(a) Detección de formas geométricas

(b) Detección de objetos en función del color

Figura 3.2: Ejemplos de algoritmos para la detección del vehículo

el lugar en el que se encuentra, y dado que el vehículo debe ser autónomo se tiene que obtener la coordenada mediante el procesamiento de las imágenes tomadas por la cámara. Por ello, se debe pensar en alguna forma de diferenciar el vehículo del resto de objetos. Existen diferentes técnicas para diferenciar un objeto:

1. Reconocimiento del vehículo debido a su forma geométrica. Existen algoritmos que sirven para detectar objetos según su forma geométrica. Figura 3.2a. Si al vehículo se le añade una forma geométrica característica, como por ejemplo un triángulo, el algoritmo será capaz de detectar en qué posición se encuentra esa forma geométrica, y por consiguiente el vehículo. [For16]
2. Establecer un patrón⁴ al vehículo. Si el vehículo posee un tamaño característico que se diferencie de los demás objetos se puede diferenciar el vehículo detectando el tamaño exacto. Esta técnica tiene sus inconvenientes, ya que según dónde se encuentre la cámara el tamaño del objeto cambia. Y también se puede dar la casualidad de que aparezca un objeto en el entorno que tenga el mismo tamaño que el vehículo. Por ello se debe establecer un patrón al vehículo para poder reconocerlo. [Pat10]
3. Existen algoritmos para la detección de objetos según su color. Por ejemplo, la librería *OpenCV* [dt17] contiene un algoritmo que consigue detectar un objeto en función de su color. [Alg15] Como se puede observar en la figura 3.2b. Pero existe el inconveniente de que aparezca en la imagen cualquier obstáculo que tenga el mismo color que el vehículo.

Una vez se ha conseguido distinguir la posición en la que se encuentra el vehículo y la posición de los obstáculos que se encuentran en el entorno, se debe procesar esta información para realizar el cálculo de trayectoria que debe seguir el vehículo para evitar colisionar.

⁴Objeto, proceso o procedimiento que sirve para definir la unidad de una magnitud física.

Cálculo de trayectoria

Existen diferentes tipos de algoritmos para calcular la trayectoria, es el diseñador el que debe elegir cuál de ellos se ajusta mejor al problema. Se van a describir una serie de algoritmos:

1. Algoritmo Djikstra [Alg12b]. El algoritmo de dijkstra determina la ruta más corta desde un nodo⁵ origen hacia los demás nodos, para ello es requerido como entrada un grafo⁶ cuyas aristas posean pesos⁷. Si los pesos de las aristas son negativos no se puede usar el algoritmo de dijkstra, para pesos negativos tenemos otro algoritmo llamado Algoritmo de Bellmand-Ford [THCS01]. Si los pesos de las aristas son de valor 1, entonces bastará con usar el algoritmo de Breadth First Search (BFS).
2. Algoritmo BFS [Alg12a]. Este algoritmo de grafos es muy útil en diversos problemas de programación. Por ejemplo, halla la ruta más corta cuando el peso entre todos los nodos es 1. Formalmente, BFS es un algoritmo de búsqueda sin información, que expande y examina todos los nodos de un árbol sistemáticamente para buscar una solución. El algoritmo no usa ninguna estrategia heurística⁸.
3. Algoritmo A* [Les03]. El algoritmo A* es un algoritmo de búsqueda que puede ser empleado para el cálculo del camino más corto en una red. Se va a tratar de un algoritmo heurístico, ya que una de sus principales características es que hará uso de una función de evaluación heurística, mediante la cual etiquetará los diferentes nodos de la red y que servirá para determinar la probabilidad de dichos nodos de pertenecer al camino óptimo.

Estos son algunos ejemplos de los algoritmos de cálculo de trayectoria existentes, pero si desea probar este tipo de algoritmos para poder obtener una visión más realista acerca de las ventajas y desventajas de cada algoritmo a simple vista, se recomienda acceder a la página web [Xu16], donde podrá definir usted mismo el entorno sobre el cual trabajará el algoritmo, así como decidir qué algoritmo utilizar, que heurística elegir y será posible ver la ejecución del algoritmo en directo.

Tras la elección del algoritmo de cálculo de trayectoria, es importante observar que el cálculo de la trayectoria en imágenes de gran resolución puede ser muy costoso en cuanto al tiempo de ejecución. Es decir, dado que el objetivo principal del proyecto es conseguir que un vehículo esquive en tiempo real los objetos que se encuentren en el entorno para llegar desde un punto de origen a un punto de destino. Se debe tener en cuenta el tiempo de ejecución de

⁵Es un punto de intersección, conexión o unión de varios elementos que confluyen en el mismo lugar

⁶en el ámbito de las ciencias de la computación es un tipo abstracto de datos (TAD), que consiste en un conjunto de nodos (también llamados vértices) y un conjunto de arcos (aristas) que establecen relaciones entre los nodos

⁷El peso de un camino en un grafo es la suma de los pesos de todas las aristas atravesadas

⁸Se conoce como heurística al conjunto de técnicas o métodos para resolver un problema. Para la informática, la heurística consiste en encontrar o construir algoritmos con buena velocidad para ser ejecutados.

CAPÍTULO 3. ANTECEDENTES

los algoritmos utilizados en el sistema. El algoritmo de cálculo de trayectoria forma el cuello de botella⁹ del sistema debido a la cantidad de recursos que maneja. La resolución que utiliza la cámara asignada al proyecto es *full HD* (2.076.000 pixel por imagen). Es por ello que es muy importante la elección del algoritmo de cálculo de trayectoria ya que hay algunos que tardan más que otros.

Comunicación

Teniendo en cuenta que se ha conseguido un algoritmo que permite calcular la trayectoria en tiempo real, el siguiente paso que se debe plantear es cómo comunicar la solución al vehículo. Hay que tener en cuenta si los datos se van a transmitir por cable o de forma inalámbrica, ya que depende de la decisión que se tome el resultado puede verse afectado. Si por ejemplo se decide realizar la comunicación por cable, este puede llegar a limitar el movimiento del vehículo debido a que se enrede con algún obstáculo del entorno. Mientras que si se realiza de forma inalámbrica se debe tener en cuenta la distancia en la cual la transmisión de datos es segura. Es decir, si el vehículo se aleja demasiado la transmisión de datos puede verse afectada debido a la lejanía.

Movimientos del vehículo

Tomada la decisión acerca de qué tipo de comunicación se va a realizar, se debe tener en cuenta la forma en la que el vehículo debe traducir la información recibida en movimientos. Para ello se deben realizar una serie de pruebas acerca de cómo reacciona el vehículo a los estímulos recibidos. Estas pruebas son imprescindibles debido a que se desconoce cómo acelera el vehículo, qué respuesta de frenada tiene, así como la sensibilidad de giro que posee.

3.2 Hardware reconfigurable

Una de las razones por las que se decidió tomar la información del sistema con la cámara es debido a que se tenía intención de embeber todo el sistema en hardware reconfigurable para optimizar el tiempo de ejecución y así poder lograr un funcionamiento del sistema en tiempo real. El hardware reconfigurable es aquél descrito mediante un lenguaje de descripción de hardware. Su naturaleza es completamente diferente a la del hardware estático¹⁰. Se desarrolla de una manera muy similar a como se hace con el software, mediante archivos de texto, que contienen el código fuente. Antes de profundizar acerca de cómo funciona el hardware reconfigurable se va a proceder a mostrar una serie de información acerca de FPGA.

Un FPGA es un circuito integrado que, dicho en términos llanos, puede configurarse para

⁹En un proceso productivo, una fase de la cadena de producción más lenta que otras, que ralentiza el proceso de producción global.

¹⁰Es el conjunto de elementos materiales o tangibles de los sistemas electrónicos.

llevar a cabo cualquier función lógica y hacer lo que a su dueño le plazca. Claro que para conseguir eso el diseñador debe programar/configurar el circuito, normalmente siguiendo la especificación de un lenguaje de descripción de hardware. Esto es algo así como preocuparse de implementar el código en el lenguaje *C* o *C++* sobre la funcionalidad del sistema, en vez de preocuparse en diseñar la electrónica digital del hardware. Una equiparación en forma de ejemplo sería si a la hora de conducir un vehículo sólo nos preocupamos de aprender las normas de circulación y cómo debemos manejar el vehículo para conducirlo, en vez de tener que diseñar y fabricar el vehículo previamente antes de poder aprender a conducirlo.

Por supuesto, en la práctica la creación está limitada por la capacidades de cada placa en la que se integra la FPGA, así como de las limitaciones existentes acerca del uso de las herramientas de diseño. Como por ejemplo las funciones y librerías de programación a las que da soporte la herramienta.

A continuación se va a proceder a explicar qué tipo de herramientas son necesarias para la consecución del objetivo de este TFG:

1. *Vivado Design Suite* [Viva]. Con el uso de esta herramienta se genera el bitstream¹¹ necesario para programar la placa FPGA. Es decir, se incluyen e integran módulos con diferentes funciones para poder generar el sistema operativo sobre el que ejecutar el sistema que desarrollemos. Llamamente hablando, se podría decir que con esta herramienta configuramos el hardware de tal forma que indicamos a la placa qué partes vamos a utilizar.
2. *Vivado Software Development Kit (SDK)* [Vivc]. Esta herramienta nos permite programar y depurar el software que queremos incluir mediante un módulo en la placa. Como lenguajes de programación solo se pueden utilizar los lenguajes *C* y *C++* por lo que el entorno de desarrollo está enfocado a la programación en estos lenguajes. Este entorno tiene una funcionalidad muy parecida al IDE (Integrated Development Environment) *Eclipse*. Un claro ejemplo de ello es la forma en la que se incluyen las librerías necesarias para el desarrollo del programa.
3. *Vivado High-Level Synthesis (HLS)* [Vivb]. Mediante esta herramienta se puede realizar la simulación del funcionamiento del algoritmo en el lenguaje *C* o *C++*, la síntesis del algoritmo que se desea embeber en la placa *Zedboard* partiendo de la descripción en *C* o *C++*, en vez de utilizar una descripción en VHDL (VHSIC Hardware Description Language). La co-simulación que realiza una comparación con el modelo RTL (Register Transfer Level) y finalmente el empaquetado del algoritmo para su programación en la placa.
4. *Petalinux* [Pet]. Esta herramienta se ha utilizado para realizar la compilación de la aplicación del sistema para el procesador ARM de la placa desde un ordenador con

¹¹Tecnología de transferencia de datos muy utilizada en gráfica y computación.

CAPÍTULO 3. ANTECEDENTES

una distribución de x64 bits. Es decir, se ha realizado una compilación cruzada desde un ordenador con un procesador distinto emulando la configuración del procesador ARM. Realizar la compilación del algoritmo en el procesador ARM directamente es muy complejo ya que no tiene el mismo soporte que un procesador con distribución de x64 bits. Por ello, mediante el uso de esta herramienta se ha generado un directorio sobre el cual se emula la compilación en el procesador ARM y así solo es necesario mover el fichero binario, generado en este directorio, a la placa para su ejecución.

Capítulo 4

Metodología de la gestión del proyecto

A Causa de la multitud de objetivos que se deben realizar para llevar a cabo este TFG, es necesario seguir una metodología de trabajo para poder ir desarrollando el proyecto de una forma clara, organizada y acotada en función del tiempo.

Si nos basamos en la definición del Project Management Institute (PMI)¹, la gestión de proyectos sería la aplicación de herramientas, conocimientos, habilidades, y técnicas para conseguir los objetivos del proyecto. A continuación se van a enumerar las metodologías de gestión más utilizadas, según la referencia [Met]:

1. PMBOK (Project Management Body of Knowledge) [PMB]. El PMI elaboró este libro, donde se establece todo un conjunto de herramientas y buenas prácticas que todo jefe de proyecto debe conocer y aplicar. El PMBOK está orientado a una gestión predictiva de los proyectos. Atiende a las diversas fases de una planificación lineal, una vez superada una etapa, no se volverá a ella. La planificación se realiza en la fase inicial, en donde se estable la necesidad o solución, el alcance de las actividades y se predice su coste.
2. CCPM (Critical Chain Project Management) [CCP] o método de la cadena crítica, basada en la teoría de las limitaciones. Este método define el plazo mínimo en que un proyecto puede terminarse e impone las restricciones que consiguen forzar a no perder la alineación con la secuencia de actividades de menor duración. Se trata de una metodología perfecta para proyectos complejos en los que los recursos son muy escasos, reduciendo al mismo tiempo el tiempo de finalización del proyecto.
3. SCRUM [Scr]. La metodología ágil² por excelencia. SCRUM es un proceso de metodología ágil que se usa para minimizar los riesgos durante la realización de un proyecto, pero de manera colaborativa. Entre las ventajas se encuentran la productividad, calidad y frecuente seguimiento de los avances del proyecto, logrando que los integrantes estén unidos, comunicados y que el cliente vaya viendo los avances.

¹PMI es la asociación profesional sin fines de lucro más importante y de mayor crecimiento a nivel mundial que tiene como misión convertir la gerencia de proyectos como la actividad indispensable para obtener resultados en cualquier actividad de negocios.

²Las metodologías ágiles son una serie de técnicas para la gestión de proyectos que han surgido como contraposición a los métodos clásicos de gestión como CMMI.

4. CMMI (Capability Maturity Model Integration) [CMM]. Se tiende a pensar que es la contraposición con metodologías ágiles como Scrum. Sin embargo, debemos entender que CMMI es un modelo no una metodología. Se centra en el qué se espera encontrar en una organización, mientras que metodologías y métodos ágiles se centran en cómo elaborar productos. Y es que ambas pueden ser utilizadas al mismo tiempo, sin necesidad de decantarnos exclusivamente por una, tal y como explica el informe *CMMI or Agile: Why Not Embrace Both!* [Why] del SEI (Software Engineering Institute)³.
5. LEAN [LEA] Se trata de una metodología orientada para procesos de producción industrial. Su objetivo principal es dar velocidad de respuesta por medio de la reducción de desperdicios, costes y tiempos. La gestión se centra en especificar el valor del consumidor, permitiendo producir sólo lo que el cliente pide, para evitar así la generación de un stock innecesario.
6. SMART (Specific, Measurable, Achievable, Relevant and Time-bound) Planning [SMA]. Metodología de planificación orientada a la especificación de objetivos que sean realmente aplicables en un plan de acción. Ése será su principal cometido: cumplir con lo planificado. Para ello, los objetivos deberán ser específicos, medibles, alcanzables, realistas y acotados en el tiempo. Herramientas de gestión como *Sinnaps* trabajan para que las tareas planificadas cumplan con estas condiciones, permitiendo un control y evaluación continua a lo largo del proyecto.

La metodología de trabajo que se ha seguido en este proyecto es SCRUM. Antes de proceder a explicar el porqué se ha elegido esta metodología se va a proceder a ver de qué trata.

4.1 Metodología SCRUM

SCRUM es un proceso de la Metodología Ágil que se usa para minimizar los riesgos durante la realización de un proyecto, pero de manera colaborativa. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos. Por ello, SCRUM está especialmente indicado para proyectos en entornos complejos. Donde se necesita obtener resultados pronto; donde los requisitos son cambiantes o poco definidos; donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

4.1.1 Roles

El equipo SCRUM está formado por los siguientes roles:

- SCRUM master: Persona que lidera al equipo guiándolo para que cumpla las reglas

³Es el instituto federal estadounidense de investigación y desarrollo, fundado por el Congreso de los Estados Unidos en 1984 para desarrollar modelos de evaluación y mejora en el desarrollo de software. Financiado por el Departamento de Defensa de los Estados Unidos y administrado por la Universidad Carnegie Mellon.

y procesos de la metodología. Gestiona la reducción de impedimentos del proyecto y trabaja con el Product Owner para maximizar el ROI (Return of Investment)⁴.

- Product owner: Representante de los accionistas y clientes, puede ser interno o externo a la organización, que usan el software. Se focaliza en la parte de negocio y es responsable del ROI del proyecto. Traslada la visión del proyecto al equipo, formaliza las prestaciones en historias a incorporar en el Product Backlog⁵ y las reprioriza de forma regular.
- Team: Grupo de profesionales con los conocimientos técnicos necesarios y que desarrollan el proyecto de manera conjunta llevando a cabo las historias a las que se comprometen al inicio de cada sprint, también llamado iteración.
- Cliente: Persona u organización interesado/a en obtener el producto.

4.1.2 ¿Cómo funciona SCRUM?

En SCRUM un proyecto se ejecuta en bloques temporales cortos y fijos. Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

El proceso parte de la lista de objetivos/requisitos priorizada del producto, que actúa como plan del proyecto. En esta lista el cliente ordena los objetivos balanceando el valor que le aportan respecto a su coste quedando repartidos en iteraciones y entregas.

Las actividades que se llevan a cabo en SCRUM son las siguientes:

1. Planificación de las tareas. La planificación de las tareas a realizar en la iteración se divide en dos partes:
 - a) El cliente presenta al equipo la lista de requisitos priorizada del producto o proyecto, pone nombre a la meta de la iteración (de manera que ayude a tomar decisiones durante su ejecución) y propone los requisitos más prioritarios a desarrollar en ella. El equipo examina la lista, pregunta al cliente las dudas que le surgen, añade más condiciones de satisfacción y selecciona los objetivos/requisitos más prioritarios que se compromete a completar en la iteración, de manera que puedan ser entregados si el cliente lo solicita.
 - b) El equipo planifica la iteración, elabora la táctica que le permitirá conseguir el mejor resultado posible con el mínimo esfuerzo. Esta actividad la realiza el equipo dado que es el responsable de organizar su trabajo y es quien mejor conoce cómo realizarlo.
2. Ejecución de la iteración. Para poder completar el máximo de requisitos en la iteración,

⁴Entregar un valor superior al dinero invertido

⁵La lista de objetivos/requisitos priorizada representa la visión y expectativas del cliente respecto a los objetivos y entregas del producto o proyecto.

se debe minimizar el número de objetivos/requisitos en que el equipo trabaja simultáneamente, WIP (Work In Progress), completando primero los que den más valor al cliente. Esta forma de trabajar, que se ve facilitada por la propia estructura de la lista de tareas de la iteración, permite tener más capacidad de reacción frente a cambios o situaciones inesperadas.

Existe la restricción de que no se pueden cambiar los objetivos/requisitos de la iteración en curso. Sólo en situaciones muy excepcionales el cliente o el equipo pueden solicitar una terminación anormal de la iteración. Esto puede suceder si, por ejemplo, el contexto del proyecto ha cambiado enormemente y no es posible esperar al final de la iteración para aplicar cambios, o si el equipo encuentra que es imposible cumplir con el compromiso adquirido. En ese caso, se dará por finalizada la iteración y se dará inicio a otra mediante una reunión de planificación de la iteración.

3. Inspección y adaptación. El último día de la iteración se realiza la reunión de revisión de la iteración. Tiene dos partes:

- a) Demostración. El equipo presenta al cliente los requisitos completados en la iteración. En función de los resultados mostrados y de los cambios que haya habido en el contexto del proyecto, el cliente realiza las adaptaciones necesarias de manera objetiva, ya desde la primera iteración, replanificando el proyecto.
- b) Retrospectiva. El equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad. El líder del equipo se encargará de ir eliminando los obstáculos identificados.

4.1.3 Ventajas

SCRUM es una propuesta de gestión basada en la división del trabajo en iteraciones, es decir, fases con objetivos y tareas específicas. Esto hace que necesariamente aporte beneficios en aspectos como los siguientes:

- Gestión de las expectativas de los clientes. Los clientes pueden participar en cada una de las iteraciones y proponer soluciones. De hecho, el proceso está pensado para un tipo de evaluación conjunta.
- Resultados anticipados. Cada iteración logra una serie de resultados. No es necesario que el cliente espere hasta el final para ver el producto.
- Flexibilidad y adaptación a los contextos. Se adapta a cualquier contexto, área o sector de la gestión. No es una técnica exclusiva de ninguna disciplina.
- Gestión sistemática de riesgos. Del mismo modo, los riesgos que pueden afectar a un proyecto son gestionados en el mismo momento de su aparición. La intervención de los equipos de trabajo es inmediata.

También se utiliza para resolver situaciones en las que no se está entregando al cliente lo que necesita; cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable; cuando se necesita capacidad de reacción ante la competencia; cuando la moral de los equipos es baja y la rotación alta; cuando es necesario identificar y solucionar inefficiencias sistemáticamente; o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo del producto.

4.1.4 Desventajas

Pero ojo, no todo es oro lo que reluce. Se deben tener en cuenta estos aspectos que pueden dar lugar a un mal uso de la metodología:

- Funciona sobre todo con equipos reducidos. Las empresas grandes, por ejemplo, deben estar sectorizadas o divididas en grupos con objetivos concretos. De lo contrario, el efecto de la técnica se perderá.
- Requiere una exhaustiva definición de las tareas y sus plazos. Cuando estos dos aspectos no se definen adecuadamente, SCRUM se desvanece. La división del trabajo en iteraciones es la esencia de esta metodología.
- Exige una alta cualificación o formación. No es una modalidad de gestión propia de grupos junior o que apenas estén en proceso de formación. Gran parte del éxito de SCRUM radica en la experiencia que aportan los profesionales de los equipos, quienes por lo general acumulan años de experiencia.

Se debe tener en cuenta el tipo de proyecto con el que se trabaja ya que ninguna metodología de gestión de proyectos es la mejor en todos los formatos de empresa existentes, hay algunos tipos de empresas o proyectos que se adaptan mejor a otras metodologías.

4.1.5 ¿Porqué se elige la metodología Scrum para este TFG?

Aunque la metodología de gestión de proyectos SCRUM esté enfocada para grupos de trabajo, se ajusta de forma extraordinaria a los recursos del TFG. Esto es debido a que la función de líder del equipo de desarrollo la puede ejercer el tutor de este trabajo, debido a que posee una gran experiencia a la hora de saber cómo gestionar el desarrollo de un proyecto, mientras que el rol de cliente lo ejerce el alumno, ya que el alumno es quien posee la idea principal del TFG. El tutor valora esta idea y propone mejoras en su diseño. Por último, el rol de equipo desarrollador lo puede ejercer solo el alumno ya que es quien implementa el proyecto.

Además, el proceso de funcionamiento de esta metodología se ajusta perfectamente al TFG debido a que entre el alumno y el tutor se pueden planificar las tareas que debe desarrollar el alumno, eligiendo cuáles son las más importantes para priorizar en su implementación, de tal forma que se puede planificar la iteración para conseguir el mejor resultado posible con el menor esfuerzo gracias al trabajo en equipo.

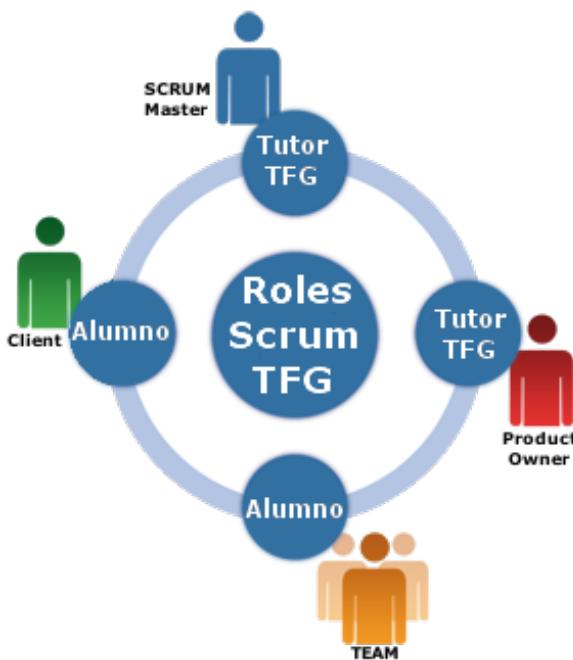


Figura 4.1: Roles de Scrum para este TFG

En lo referente a la ejecución de la iteración también se ajusta perfectamente a este proyecto, dado que como solo debe hacer el trabajo el alumno se reduce al mínimo el número de objetivos/requisitos en los que el equipo debe trabajar simultáneamente. Adicionalmente, también se van a completar primero los objetivos importantes para el cliente, es decir los que han elegido el alumno y el tutor previamente, de tal forma que si surge algún problema se tiene mucha capacidad de reacción frente a situaciones inesperadas, como por ejemplo si el contexto del proyecto cambia debido a que se prefiere orientar de otra forma.

Acerca de la inspección y adaptación de esta metodología al proyecto, se considera afín. Ya que el alumno y el tutor deciden que día va a ser la próxima reunión para mostrar los resultados. De esta forma se pueden realizar las consideraciones pertinentes acerca de si el trabajo de la iteración necesita alguna adaptación, o bien si la manera de trabajar se puede optimizar mediante las preguntas que realiza el alumno al profesor en las reuniones programadas.

4.1.6 Herramientas utilizadas para la resolución del proyecto

A continuación se van a mencionar todas las herramientas utilizadas para el desarrollo del proyecto.

Software

- IDE Arduino. Entorno de desarrollo para la programación de las placas Arduino.
- Bitbucket. Repositorio donde se ha almacenado todo el proyecto.
- Eclipse. Entorno de desarrollo sobre el que se realiza la programación de los algoritmos que forman el sistema.

- Fritzing. Programa para la realización de esquemas eléctricos en proyectos con Arduino.
- Gimp. Programa de manipulación de imágenes sobre el cual se han diseñado y modificado figuras que aparecen en el proyecto.
- StarUML. Programa para diseñar gráficos UML (Unified Modeling Language) con licencia Open Source.
- Texmaker. Entorno de desarrollo sobre el que se ha realizado la memoria del proyecto.
- Vim. Editor de texto utilizado en la programación de algoritmos.
- Visual Paradigm. Este es un software de modelado UML que nos permite analizar, diseñar, codificar, probar y desplegar. Dibuja todo tipo de diagramas UML, genera código fuente a partir de dichos diagramas y también posibilita la elaboración de documentos.
- Vivado Design Suite. Con el uso de esta herramienta se genera el bitstream necesario para programar la placa FPGA
- Vivado SDK. Software necesario para embeber los módulos IP (Intellectual Property) desarrollados en Vivado HLS en la placa *Zedboard*.
- Vivado HLS. Para poder programar y verificar el funcionamiento de los algoritmos que se desean embeber en la placa.
- Petalinux. Herramienta necesaria para realizar la compilación cruzada de la aplicación que se desea ejecutar en el procesador ARM de la placa.
- XCTU. Programa necesario para realizar la conexión entre los dos módulos de comunicación *XBee*.

Hardware

- Arduino Shield. Imprescindible para poder utilizar los pines de Arduino que no utiliza el controlador Ardumoto.
- Arduino UNO R3 ATMEGA328. Plataforma computacional con la que se controlan los elementos de prototipado que, en su conjunto, forman el vehículo.
- Batería recargable *Odec* 9V. Fuente de alimentación del vehículo.
- Controlador *Sparkfun* Ardumoto. Módulo con el cual se activa y configura el movimiento de los motores.
- Estación de soldadura JBM AR5800.
- Imán de neodimio. Utilizado para interactuar con el sensor de efecto Hall.
- Placa de pruebas para Arduino. Sobre esta placa se establece la instalación de los sensores, leds y resistencias del coche.

- Placa *Zedboard* de la marca *Xilinx*. Computador sobre el cual se realizarán los cálculos del sistema.
- Módulo *Xbee* de la marca *Digi*. Módulos con los que se realizará la comunicación entre la placa *Zedboard* y *Arduino*.
- Osciloscopio y multímetro.
- Sensor efecto Hall A3144. Utilizado para contar las vueltas que realizan las ruedas del coche.

Capítulo 5

Desarrollo

Comencemos con el desarrollo del proyecto. Se van a plantear las etapas del proyecto por iteraciones. El número de iteraciones representarán los objetivos que se han ido planteando hasta la resolución de los mismos.

Iteración 5.1: Tratamiento de la información de una imagen

Obtención de los datos que contiene una imagen con formato BMP.

Iteración 5.2: Algoritmo para detectar cambios en el entorno

Detección de objetos mediante la resta de la información de dos imágenes.

Iteración 5.3: Delimitación de los obstáculos

Delimitación de los obstáculos mediante la representación de los cuatro puntos representativos en cuadrados o rectángulos.

Iteración 5.4: Implementación del algoritmo A* en C++, sin restricciones

Implementación del algoritmo A* en C++, sin restricciones de programación.

Iteración 5.5: Detección del vehículo frente a los objetos

Detección de la coordenada en la que se encuentra el vehículo.

Iteración 5.6: Diseño del vehículo del sistema

Diseño del vehículo con el que trabajará el sistema. *OpenCV*.

Iteración 5.7: Comunicación con el vehículo

Establecimiento de la comunicación entre la placa *Zedboard* y el vehículo mediante módulos *Xbee*.

Iteración 5.8: Desarrollo del firmware para el control del vehículo

Programación del vehículo para su control.

Iteración 5.9: Integración de los algoritmos en el sistema

Integración de todos los algoritmos desarrollados en el sistema.

Iteración 5.10: Optimizaciones realizadas en el sistema

Optimización de algunos de los algoritmos del sistema.

Iteración 5.11: Integración del sistema en la placa Zedboard

Integración del sistema en la placa *Zedboard* de Xilinx mediante el uso de las herra-

mientas *Vivado* y *Petalinux*.

Reunión inicial del proyecto

Para definir una planificación en función de los objetivos del proyecto se decidió establecer una serie de premisas que se debían tener en cuenta en la realización del proyecto. Estas premisas sirven para incorporar una serie de buenos hábitos y reglas al desarrollo del trabajo:

1. Las reuniones con el tutor del proyecto deben ser una vez por semana.
2. Solo se establecerá un objetivo por reunión.
3. Cada objetivo que se plantea en las reuniones se considerará una iteración.
4. El trabajo de cada iteración se debe definir exclusivamente al objetivo específico que se plantea en las reuniones con el tutor.
5. La duración de las iteraciones puede llevar más tiempo de una semana.
6. Es imprescindible el uso de repositorios para salvar la integridad del proyecto.
7. Cada vez que se termine una iteración se debe documentar el trabajo realizado.

5.1 Tratamiento de la información de una imagen

Se decide utilizar el formato BMP (Bits Maps Protocole) debido a que no sufre pérdidas de calidad y por tanto resulta adecuado para guardar imágenes que se desean manipular posteriormente. El único inconveniente es que el tamaño de las imágenes es grande. Para imágenes con resolución *FullHD* (1920x1080) correspondería a 3.7 Megabyte (MB) por imagen en color y 2.1 MB por imagen en escala de grises.

El uso de imágenes como argumentos del sistema tiene la ventaja de que cada fotografía posee la información completa del escenario. Es decir, la información de la cabecera también nos sirve para comprobar si el formato de la imagen que se pasa como argumento es correcto. Ya que las imágenes con formato BMP poseen la firma *BM*, 424D en hexadecimal, que indica que se trata de un mapa de bits de Windows. Gracias a que la cabecera posee información acerca de la anchura de la imagen, la altura, el número de bits por píxel, el tamaño total, la resolución horizontal y la resolución vertical se puede implementar el sistema para que funcione con cualquier resolución. Ya que los algoritmos que necesiten este tipo de datos, tomarán la información que se ha obtenido de la cabecera. De esta forma, aunque se use una cámara de mejor o peor calidad, el sistema podrá realizar los cálculos pertinentes sin problema ya que reservará memoria en función de los parámetros de la cabecera.

5.1.1 Cabecera del formato BMP

El algoritmo que se tiene pensado implementar posteriormente está pensado para que funcione con imágenes en blanco y negro. Por lo que para esta iteración el objetivo es realizar

un programa con el que se extraiga la información de una imagen en blanco y negro. Es decir, hay que transformar el valor de cada píxel de 24 bits de información, por defecto correspondiente al modelo Red-Green-Blue (RGB)¹, a 8 bits. De esta forma, los posibles 256 valores por píxel servirán para representar la imagen desde el valor 255 que representa el color blanco hasta el valor 0 que representa el color negro. Para realizar esta conversión de color a blanco y negro se debe tener en cuenta que los 24 bits de información representan los 3 bytes correspondientes al color rojo, verde y azul. Véase el cuadro 5.1. Por consiguiente, se deben sumar los valores correspondientes a los 3 bytes y posteriormente realizar una división entre estos. Esta operación se debe realizar por cada píxel de la imagen. De esta forma, pasaremos a tener un byte por cada píxel que corresponderá al valor, en escala de grises, de la imagen a color.

Desplazamiento	Bytes	Descripción
0x00	2	Tipo de formato de la imagen
0x02	4	Tamaño del fichero en bytes.
0x06	4	Reservado
0x0A	4	Desplazamiento matriz píxeles
0x0E	4	Tamaño estructura en bytes
0x12	4	Ancho de la imagen
0x16	4	Largo de la imagen
0x1A	2	Número de planos
0x1C	2	Bits por píxel
0x1E	4	Tipo de compresión
0x22	4	Tamaño de la imagen
0x26	4	Resolución vertical
0x2A	4	Resolución horizontal
0x2E	4	Número de colores
0x32	4	Número de colores requeridos
0x33	1	Componente rojo
0x34	1	Componente verde
0x35	1	Componente azul
0x36	1	Reservado
0x37	Tamaño	La matriz de datos

Cuadro 5.1: Cabecera formato BMP

5.2 Algoritmo para detectar cambios en el entorno

Para detectar cambios en el entorno se tomarán imágenes con la cámara cenital cada cierto tiempo (Imagen *B*) y se comparará con una imagen original sin obstáculos (Imagen *A*). Para realizar la comparación entre las dos imágenes se ha decidido utilizar una operación de resta

¹RGB es un modelo de color basado en la síntesis aditiva, con el que es posible representar un color mediante la mezcla por adición de los tres colores de luz primarios. Rojo, verde y azul.

entre la imagen *A* y *B*. Esta resta va a permitir saber que:

Regla Si la diferencia entre el valor de la misma coordenada, de las dos imágenes, es mayor o menor que un intervalo (definido en función de las características del escenario) significa que en esa coordenada ha aparecido un nuevo objeto.

Esta regla se cumple ya que si en la coordenada de la imagen *A* no existe ningún objeto y en la misma coordenada de la imagen *B* existe un objeto, significa que el resultado de la resta entre el valor de la coordenada *A* menos el valor de la coordenada *B* debe ser mayor o menor que 0 dado que los valores de los píxel son distintos. Por ejemplo, si en la imagen *A* el valor que representa el suelo del escenario es el 200 (Recordar que los valores de los píxel van desde 0 hasta 255) y en la imagen *B* el valor que representa el objeto es el 50, la resta entre estos dos valores daría lugar al resultado 150, por lo que al ser distinto que cero significa que ha aparecido un nuevo objeto.

En la descripción de la regla se hace mención a un intervalo, esto es debido a que la cámara no tiene una buena precisión a la hora de capturar la información que hace referencia a cada píxel. Ya que la luz del entorno, el polvo, la proyección de una sombra, etc... pueden dar lugar a que, aún teniendo el mismo escenario, los valores de los píxeles no coincidan. Por ello, se ha decidido declarar un intervalo en función del entorno en el que se sitúe el sistema para poder evitar el problema de la falta de precisión de la cámara.

En la figura 5.1 se observan ejemplos de la imagen inicial *A*, imagen con obstáculos *B* y la imagen obtenida con la operación de resta.

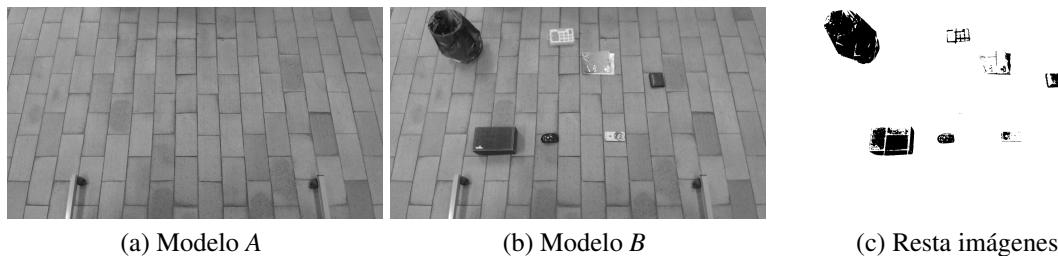


Figura 5.1: Ejemplo resta de imágenes

5.3 Delimitación de los obstáculos

Con tal de evitar que la forma irregular de los objetos pueda dar lugar a que el vehículo colisione con alguno de ellos, se ha decidido implementar un algoritmo que pinte un cuadrado o rectángulo alrededor de los objetos. Para ello, es necesario tomar los 4 puntos más representativos del objeto. el punto superior en la imagen, el punto inferior, el punto más a la izquierda y el punto más a la derecha.

Aunque parezca trivial, es un algoritmo complejo. Estos son los problemas que se han de resolver:

1. ¿Qué punto representativo corresponde a cada objeto?
2. ¿Cómo guardar la información de forma coherente?
3. ¿La distancia entre dos objetos es menor o igual que el tamaño del vehículo?

5.3.1 ¿Qué punto corresponde a cada objeto?

Para la resolución de este algoritmo se ha decidido recorrer la matriz de datos en dos fases:

- Fase de análisis

Hay que tener en cuenta que por cada iteración se recorre una columna entera de las 1920 posibles. En esta fase se tomará la información acerca de cuántos objetos hay en dicha columna. Si el número de objetos es distinto en comparación con la columna anterior significa que ha comenzado o terminado uno o varios objetos. Como se puede observar hay tres casos posibles referentes al número de objetos:

1. Que sea menor. Si el número de objetos actual es menor frente al de la columna anterior quiere decir que un objeto ha finalizado. Si se produce esta situación se debe guardar el punto representativo más a la derecha y se debe realizar una comparación tal que si el punto es más alto que en la columna anterior se debe reemplazar por el existente. De manera inversa ocurre lo mismo con el punto inferior ya que si el punto es más pequeño que el guardado anteriormente, se sustituye.
2. Que sea mayor. Si el número de objetos actual es mayor frente al de la columna anterior quiere decir que un objeto nuevo se ha identificado en la matriz de datos. Dado que ha aparecido, se tienen que guardar los valores de los puntos representativos hasta el momento. Es decir, el punto superior e inferior (que pueden ser los mismos en este caso) y el punto más a la izquierda, porque el barrido va de izquierda a derecha.
3. Que sea igual. Si el número de objetos es el mismo pero es igual a cero, no se realiza ninguna operación. Debido a esto se ha implementado el algoritmo de tal forma que tras la etapa de análisis se comprueba si el número de objetos es mayor que uno, con un contador, y si es así se procede con la fase de resultados. De esta forma evitamos realizar comparaciones con todos los píxeles de la imagen y solo procesamos las columnas que contienen objetos. Si el número es igual pero es mayor que uno, se realiza una comparación entre los valores de los objetos para ver si alguno es mayor o menor que otro punto representativo y merece la pena guardar la información.

- Fase de resultados

En esta fase se desplaza el puntero a la derecha por cada iteración y se lleva un orden acerca de dónde se guardan los puntos más representativos de cada objeto.

5.3.2 ¿Cómo guardar la información de forma coherente?

Cabe mencionar que se ha utilizado un buffer para las coordenadas X y otro buffer para las coordenadas Y, donde se guardará la información que se capture en la fase de análisis para su posterior comparación con los resultados que se guardan en las matrices de resultados para las coordenadas X e Y. Cada objeto posee dos coordenadas X y dos coordenadas Y. Por lo que el primer valor para la coordenada X corresponderá al punto más representativo de la izquierda y el primer valor de la coordenada Y al punto más representativo superior. De tal forma que una vez se hayan examinado las 1920 columnas, se tengan las coordenadas guardadas en las matrices de resultados de las coordenadas X e Y.

Se va a seguir la misma estructura en cuanto a fase de análisis y resultados para explicar cómo se guardan los datos:

- Fase de análisis

De igual forma se va a mantener la enumeración acerca de los números de objetos;

1. Que sea menor. En este caso, si el número de objetos es menor se debe guardar el valor correspondiente a la coordenada X de la columna anterior. Es decir, $x - 1$. Así como realizar una comparación con los valores de los puntos representativos superior e inferior por si los valores de la columna anterior son superiores o inferiores respectivamente.
2. Que sea mayor. Si el resultado del número de objetos de la columna actual es mayor que la columna anterior, en primer lugar se debe comprobar si el número de objetos es mayor que uno o igual que uno, debido a que si, por ejemplo, el número de objetos de la columna anterior es uno y el actual dos significa que ha aparecido un nuevo objeto que todavía no se ha terminado de obtener información del objeto que se encuentra en la misma columna a diferente altura. Es por esto, que se debe tener en cuenta el desplazamiento en función del número de objetos ya que sino eliminaríamos valores de objetos que todavía no han terminado de comprobar todos sus puntos. En este caso en concreto se guardaría la coordenada X de la columna actual y las coordenadas Y, aunque tengan el mismo valor, se deben guardar en el punto superior y el punto inferior, ya que posteriormente se irá actualizando.
3. Que sea igual. Si el número de objetos es igual, pero es mayor que cero se debe comprobar el valor de las coordenadas Y, del punto superior e inferior de la coordenada actual por si los valores son más representativos. La información se encuentra en un buffer por lo que el acceso a las coordenadas Y se hace mediante las coordenadas que se han capturado en la fase de análisis y guardado en el buf-

fer. El contador del número de objetos es el que nos servirá para saber a que parte de la matriz debo acceder para consultar la coordenada Y que se ha capturado en la fase de análisis.

- Fase de resultados

En la fase de resultados se actualizan los valores de las coordenadas que pasarán a ser las coordenadas de la columna anterior en la siguiente fase de análisis, y se actualizará el buffer con las variables de la coordenada X según el número de objetos que haya habido.

5.3.3 ¿Distancia entre objetos \leq tamaño del vehículo?

Se va a proceder a describir la parte del algoritmo que controla si la distancia entre dos objetos es menor o igual que el tamaño de vehículo para que se tomen los puntos representativos de los dos objetos en conjunto. Es decir que el cuadrado o rectángulo abarque los dos objetos en uno.



Figura 5.2: Ejemplo de la delimitación de los obstáculos

Para explicar como se ha ido teniendo en cuenta el proceso en concreto de delimitar los objetos, solo se van a utilizar las fases de análisis y resultados a causa de que el número de objetos, salvo que debe ser mayor que uno, no es relevante.

- Fase de análisis

Se va a producir una resta entre el punto inferior más representativo de la coordenada Y del segundo objeto y el punto superior más representativo de la coordenada Y del primer objeto. Si esta resta da un resultado menor que el tamaño del vehículo se guardará el punto inferior más representativo del segundo objeto como el punto superior más representativo del primer objeto. Y el número de objetos se decrementará en uno. El procedimiento es el mismo para un número de objetos mayor.

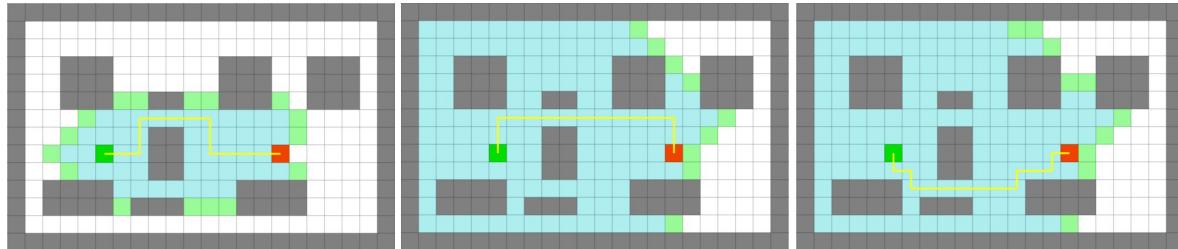
- Fase de resultados

En esta fase se va a realizar el mismo tipo de cálculo que en la fase de análisis, pero esta vez con las coordenadas X. Si el número de objetos es mayor que uno se

realizará una resta entre el punto más a la izquierda del segundo objeto y el punto más a la derecha del primer objeto. Si el resultado de esta operación es menor que el tamaño del vehículo, se modificará el punto más a la derecha del primer objeto por el punto representativo más a la izquierda del segundo objeto. Y también se decrementará en uno el número de objetos.

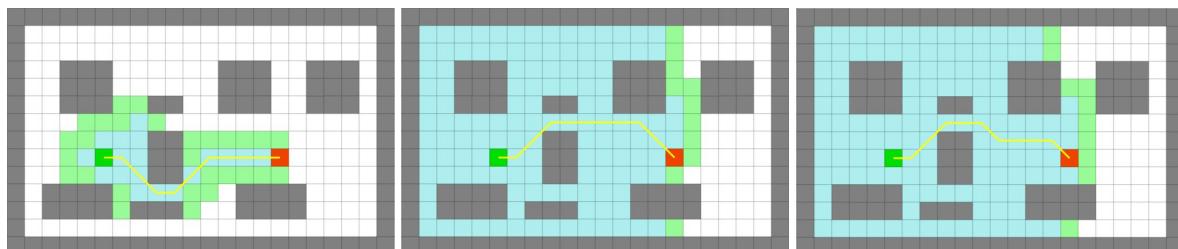
5.4 Implementación del algoritmo A* en C++, sin restricciones

Se ha decidido utilizar el algoritmo de A* debido a que es un algoritmo que calcula la ruta óptima realizando un número de operaciones menor que otro tipo de algoritmos clásicos como Djikstra o BFS. Se han realizado un número de pruebas acerca de los resultados obtenidos de estos tres algoritmos y estos han sido los resultados para el mismo caso de estudio.



(a) Algoritmo A* Longitud = 14 (b) BFS: Longitud = 14 ,Tiempo =(c) Djikstra: Longitud = 14 ,Tiempo = 1.2 ms, Operaciones: 114 1.4 ms, Operaciones: 277
= 5.0 ms, Operaciones: 274

Figura 5.3: Casos de prueba con 4 Direcciones



(a) Algoritmo A* Longitud = 11.7 (b) BFS: Longitud = 11.7 ,Tiempo =(c) Djikstra: Longitud = 11.7 ,Tiempo = 3.6 ms, Operaciones: 68 1.7 ms, Operaciones: 273
= 3.5 ms, Operaciones: 270

Figura 5.4: Casos de prueba con 8 Direcciones

Para esta iteración, el objetivo principal es conseguir que funcione el algoritmo de cálculo de trayectoria A*. Se ha utilizado de base una implementación realizada en el lenguaje C++ [FB310]. Con la modificación de esta implementación para que los datos que se utilizan como argumento sean los que se han calculado previamente en la detección de los obstáculos se ha podido calcular la trayectoria teórica por la que debe avanzar el vehículo.

5.5 Detección del vehículo frente a los objetos

Para distinguir el vehículo del resto de objetos del escenario se ha decidido utilizar una combinación de algoritmos de la librería *OpenCV*. En primer lugar se utiliza la función *cvtColor* que nos permite transformar una imagen en modelo de color RGB (Red-Green-Blue) a HSV (Hue Saturation Value)². Se ha decidido utilizar el modelo de color HSV ya que nos permite identificar un color por un solo valor, el matiz, en lugar de tres valores como en el modelo de color RGB. Con el uso de la función *cvtColor* y la función *inRange* se puede seleccionar el umbral de color que se quiera obtener de la imagen de referencia. Por ejemplo, el color rojo en *OpenCV* tiene el valor del matiz entorno al rango de 0 a 10 y de 160 a 180. Con el uso de estas dos funciones, configuradas para detectar objetos de color rojo, se obtienen los resultados de la figura 5.5.

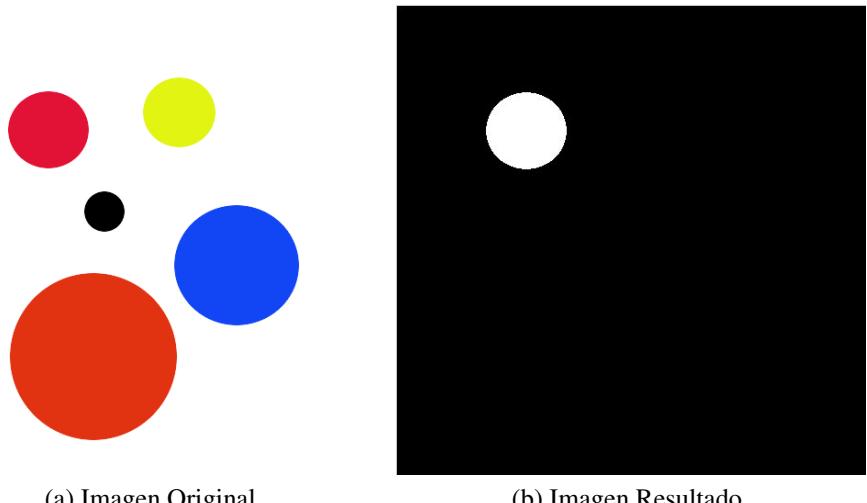


Figura 5.5: Ejemplo de uso con las funciones *cvtColor* y *inRange*

Debido a que se puede dar la situación en la que haya dos objetos de color rojo, se ha decidido utilizar la función *HoughCircles* que detecta los círculos en una imagen, de tal manera que en el caso de que haya figuras diferentes con el mismo color solo detecte las que tengan forma circular. Como aparece en la imagen 5.6. Esta funcionalidad es muy útil, ya que si se define un umbral de color reducido para el matiz y se pone una marca circular identificativa al vehículo con el color perteneciente al umbral, se puede diferenciar el vehículo del resto de objetos con facilidad.

Además, con la función *Pointcenter* se pueden obtener las coordenadas del punto central del círculo. Esta funcionalidad es muy útil ya que para poder inicializar el algoritmo de cálculo de trayectoria es necesario conocer las coordenadas en las que se encuentra el vehículo. Por ello, mediante el uso combinado de las cuatro funciones mencionadas anteriormente, se puede diferenciar el coche del resto de objetos del escenario mediante la detección de

²El modelo de color HSV es una transformación no lineal del modelo RGB en coordenadas cilíndricas de manera que cada color viene definido por el tinte o matiz, saturación y brillo.

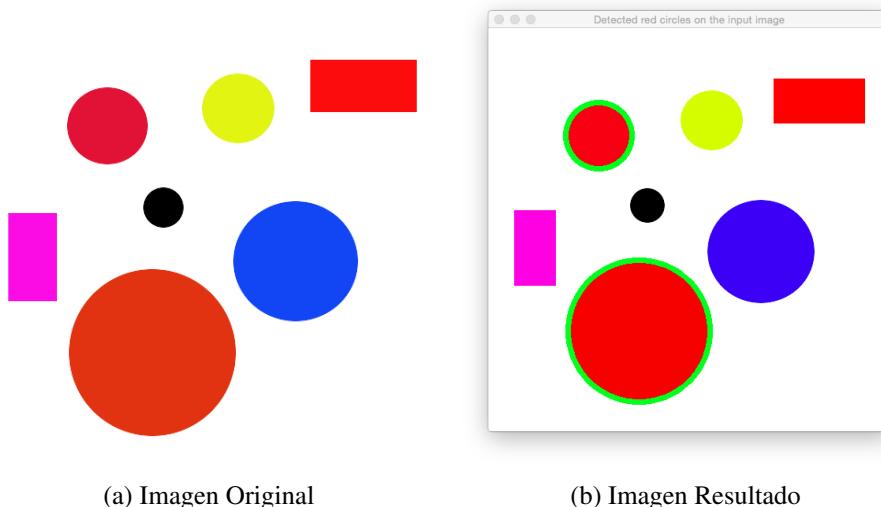


Figura 5.6: Uso de las funciones *cvtColor*, *inRange* y *HoughCircles*

la marca circular identificativa de color rojo, así como obtener las coordenadas en las que se encuentra el vehículo para poder llevar a cabo la ejecución del algoritmo de cálculo de trayectoria.

5.6 Diseño del vehículo del sistema

Como brazo ejecutor del sistema se ha decidido integrar una serie de componentes con los cuales se pueda llevar a cabo la conducción, comunicación, y traducción de la información en movimientos. A continuación se va a proceder a nombrar los elementos que forman el vehículo:

- *Arduino UNO R3 ATMEGA328*
- Controlador *Sparkfun Ardumoto*
- Chasis con motores de corriente continua de 9V.
- Módulos *Xbee* de la marca *Digi*.
- Batería recargable *Odec 9V*.
- Placa de pruebas *Arduino*.
- *Arduino Shield*.
- Sensor efecto Hall A3144.

Si desea conocer más acerca del porqué del uso de estos componentes puede echar un vistazo en el anexo B.

5.7 Comunicación con el vehículo

Para establecer la comunicación entre la placa *Zedboard* y la placa del coche *Arduino UNO* se ha decidido utilizar una pareja de módulos XBee³ de la marca *Digi* [Dig]. Mediante estos chips de conexión inalámbrica podemos establecer una comunicación serial sin necesidad de conectarlos por cable. Esta opción es muy interesante debido a que mediante el software *XCTU*⁴ que proporciona la marca *Digi* se puede establecer una comunicación punto a punto entre las dos placas del sistema de una forma rápida y sencilla.

Ahora bien, debido a la importancia que tiene que se realicen todas las órdenes de forma ordenada y secuencial, es necesario establecer un protocolo de comunicación mediante el cual se controle que no se pierde ni se corrompe ningún paquete de información.

Dado que el sistema está pensado para funcionar en tiempo real, se ha decidido implementar un protocolo específicamente pensado para este TFG de tal forma que se optimice el tamaño de datos que se envían. Se controle mediante ACK (Acknowledgement) que los paquetes de información han llegado y se comunique si ha habido algún error en la ejecución del movimiento.

5.7.1 Protocolo de comunicación

Debido a que se ha diseñado un vehículo que ejecuta sus movimientos accionando dos motores independientes entre si de corriente continua, sin encoder lineal⁵, la distancia a recorrer se traducirá en números de vueltas a ejecutar por las ruedas del vehículo. Para ello, los paquetes de información a transmitir poseeran los siguientes campos:

- Identificador de la orden que se debe ejecutar.

Se ha decidido utilizar un byte para identificar la orden a ejecutar. El sistema genera cada orden con un número identificador de manera secuencial, no genera un nuevo movimiento hasta que sepa el estado final de la comunicación que ha realizado previamente. Por ello, no se puede dar lugar la situación en la que se comuniquen dos movimientos distintos que tengan el mismo identificador.

- La dirección que debe tomar el vehículo.

Según se ha diseñado, el vehículo puede tomar cualquiera de las 8 direcciones de los puntos cardinales. Por ello, se ha decidido asignar 3 bits con los cuales se pueden representar las 8 posibles direcciones. No es necesario especificar que el vehículo frene ya que si no se realiza ninguna orden, el vehículo no acciona ningún motor.

³Los chips XBee son capaces de comunicarse de forma inalámbrica unos con otros.

⁴XCTU [XCT] es una aplicación multi-plataforma, gratuita, que incluye herramientas que facilitan la instalación, configuración y prueba de funcionamiento entre módulos XBee a través de una interfaz gráfica fácil de usar.

⁵Un encoder lineal es un dispositivo o sensor que cuenta con una escala graduada para determinar su posición. Los sensores en el encoder leen la escala para después convertir su posición codificada en una señal digital que puede ser interpretada por un controlador de movimiento electrónico.

- El número de vueltas que deben realizar las ruedas del vehículo.

La cámara cenital se puede encontrar fija a distintas alturas, por lo que la distancia real que corresponde a cada píxel puede variar. Ya que cuanto más lejos esté la cámara del suelo, mayor será la distancia real que exista entre dos píxeles de una imagen. Por ello, se ha decidido asignar un nuevo byte más el resto de bits del segundo byte en el que se encuentran los tres bits que representan la dirección que debe tomar el vehículo. Con lo cual, el número de bits que se utilizan para representar el número de vueltas que deben realizar las ruedas del vehículo son 13.

De esta forma, cada paquete tendrá un tamaño fijo de tres bytes. A continuación se va a proceder a exponer un ejemplo de cómo funciona el protocolo de comunicación.

5.7.2 Ejemplo de funcionamiento del protocolo de comunicación

Se ha decidido diseñar un escenario de prueba con el cual se pueda explicar cómo funciona en detalle el protocolo de comunicación que se ha implementado. Para ello, se ha generado un diagrama de secuencia, figura 5.7, en el cual se representa la comunicación que se establece entre la placa *Zedboard* y *Arduino*. Para evitar desarrollar un escenario muy extenso, se va a representar un ejemplo aleatorio de 4 peticiones de movimiento. Se contemplará la opción de que se pierda información en la comunicación para representar los posibles inconvenientes que se pueden dar lugar en la ejecución del sistema en tiempo real.

Posteriormente, se va a proceder a explicar cómo está estructurada la información de los paquetes que se desean comunicar. Ya que como se ha optimizado el tamaño, existen tres conjuntos de bits que representan los parámetros mencionados en la subsección 5.7.1.

Se pueden distinguir tres bloques fijos por paquete que representan los bytes, aunque la información está estructurada tal como representan los colores:

- Morado: Representa los 8 primeros bits en los cuales se desea transmitir el identificador numérico de la trama⁶ con el que se desea comprobar si la comunicación se está realizando de forma secuencial y ordenada. Como utilizamos 8 bits para representar el identificador estos valores estarán entre 0 y 255.
- Rojo: Representa los 3 primeros bits del segundo byte de la trama. Esta información se utilizará para activar los motores del vehículo de tal forma que se mueva en la dirección que interese.
- Negro: Representa los 5 últimos bits del segundo byte y los 8 bits del tercer byte de la trama. En total se utilizan 13 bits para representar el número de vueltas que deben realizar las ruedas del vehículo para completar la orden.

⁶En redes, una trama es una unidad de envío de datos. Es una serie sucesiva de bits, organizados en forma cíclica, que transportan información y que permiten en la recepción extraer esta información.

CAPÍTULO 5. DESARROLLO

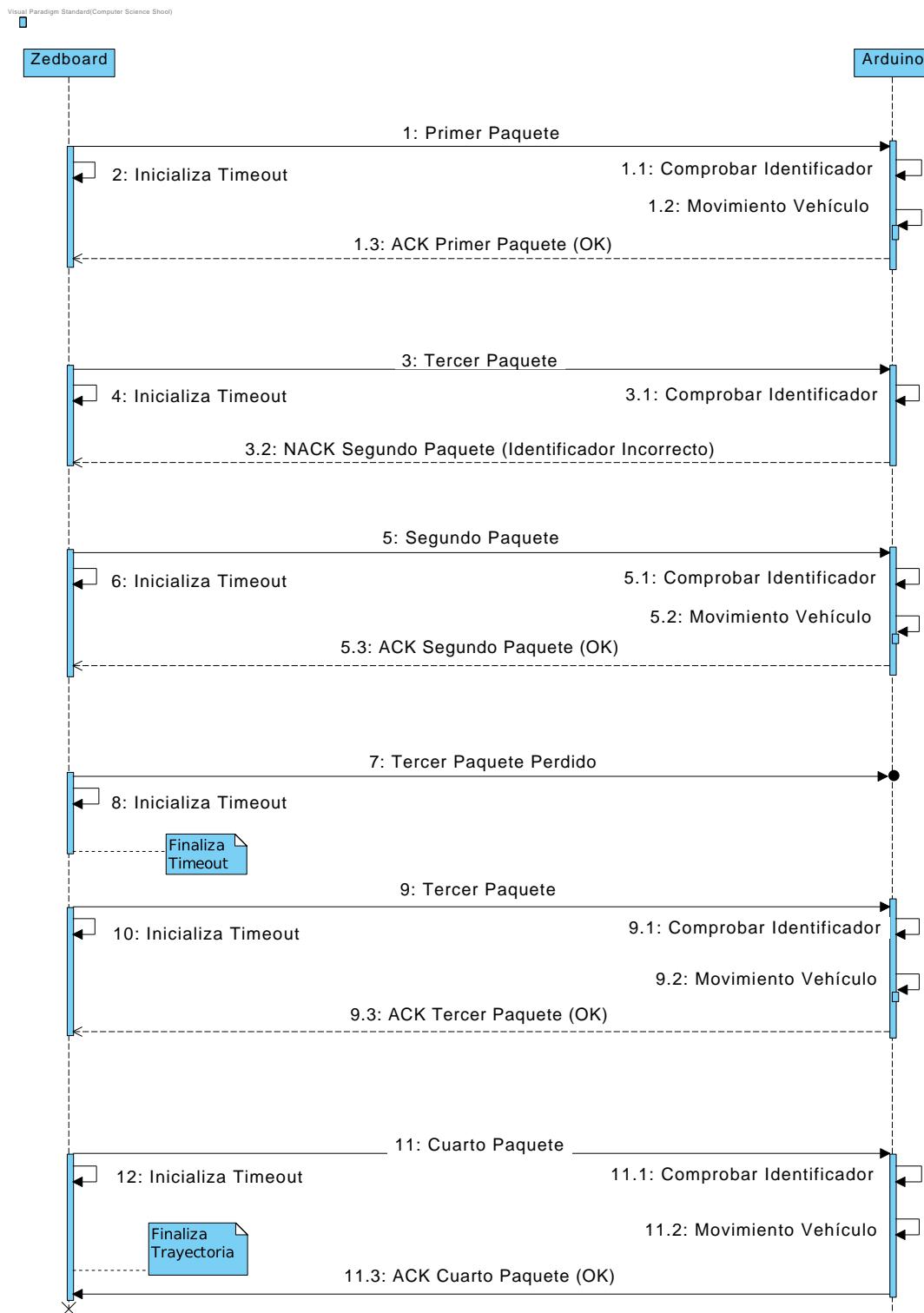


Figura 5.7: Diagrama de secuencia del protocolo de comunicación

A continuación se va a proceder a explicar el diagrama de secuencia, de la figura 5.7, con la información detallada de cada paquete.

1. Primer paquete



00000000|00100000|0000101

Se produce el envío del primer paquete desde la placa *Zedboard* para que se realice un movimiento de 5 vueltas hacia la diagonal superior derecha, según la tabla ???. Una vez se ha realizado el envío se inicializa el timeout del puerto serie de la placa *Zedboard*.

Desde la placa *Arduino* se comprueba si el identificador coincide con el número de paquete que se desea recibir. Como se espera recibir el paquete con el identificador 0, se procede a llamar al método que activa los motores para avanzar hacia la diagonal superior izquierda. Una vez se han realizado las 5 vueltas, *Arduino* devuelve un ACK a la placa *Zedboard* confirmando que se ha realizado el movimiento correctamente y finaliza la comunicación del primer paquete.

2. Segundo paquete



00000010|01100000|0000111

Debido a un error en el sistema, se procede a enviar el paquete que corresponde al identificador del tercer movimiento. Como la placa *Arduino* tiene un contador propio, para llevar el control acerca de que paquete espera, detecta que el paquete que está recibiendo desde la placa *Zedboard* no contiene la información correspondiente al segundo movimiento. Por ello, tras realizar la comprobación del identificador genera un NACK que envía a la placa *Zedboard* para comunicar que está esperando la información que corresponde al segundo movimiento. De esta forma se termina la comunicación del paquete incorrecto y se genera un nuevo paquete con la información correspondiente al identificador que espera.

Una vez se ha generado el paquete correspondiente al segundo movimiento, se envía la información y el software de *Arduino* realiza el movimiento de una vuelta hacia la derecha, enviando un ACK a la placa *Zedboard* cuando finaliza el movimiento.

3. Tercer paquete

La placa *Zedboard* envía la información correspondiente al tercer movimiento por el puerto serie. Pero debido a las interferencias que se encuentran en el escenario, este

00000010|01100000|0000111

paquete nunca llega a la placa *Arduino*. Como consecuencia, el timeout que se ha definido en el puerto serie, de la placa *Zedboard*, llega a su fin y genera una interrupción que finaliza la comunicación correspondiente al paquete del tercer movimiento.

Dado que el tercer movimiento no se ha realizado, se vuelve a enviar el paquete por el puerto serie. Esta vez la información si que llega a la placa *Arduino* realizando un movimiento de 8 vueltas de rueda hacia la dirección diagonal inferior derecha.

4. Cuarto paquete

00000011|11100000|0000011

Por último, se realiza el envío por el puerto serie correspondiente al cuarto movimiento. Que tras llegar sin problemas realiza un movimiento de tres vueltas de rueda hacia la dirección diagonal superior izquierda. Como en este ejemplo solo se realizan cuatro movimientos, el programa encargado de coordinar la comunicación, desde la placa *Zedboard*, finaliza el proceso.

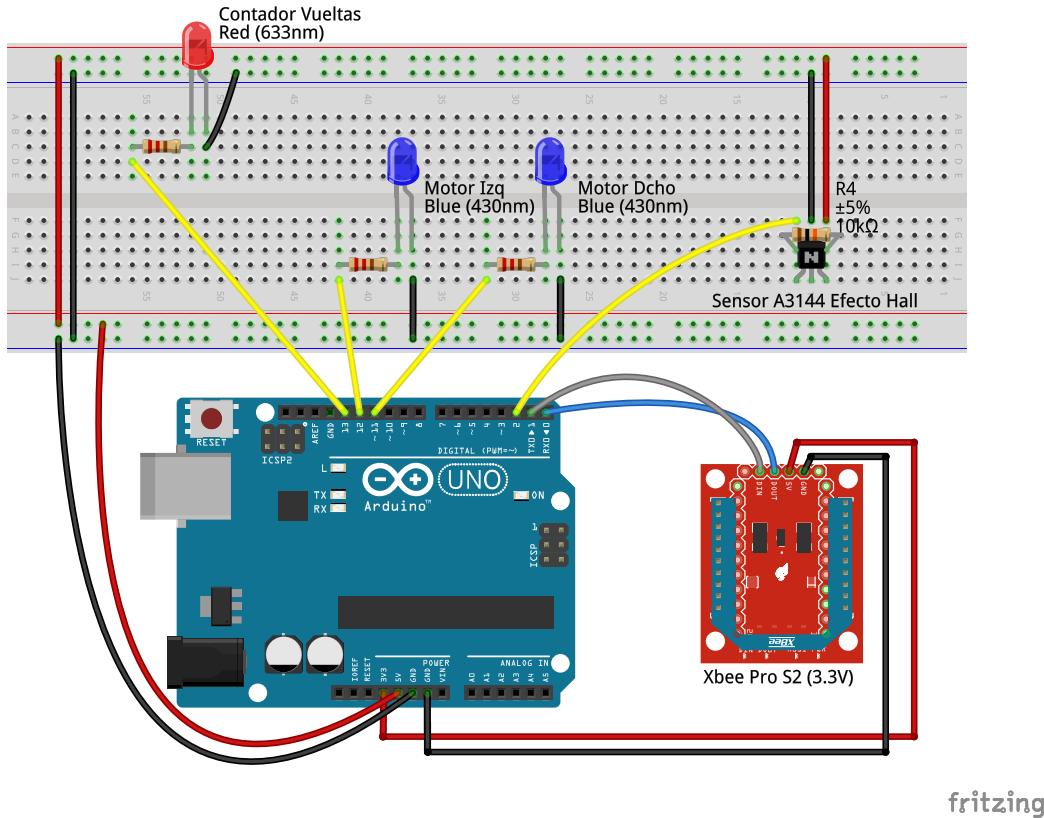
5.7.3 Maqueta de pruebas del protocolo de comunicación

Se ha diseñado una maqueta para poder comprobar el funcionamiento del protocolo de comunicación previamente a su instauración en el sistema. De esta forma se puede probar el funcionamiento del protocolo de una manera más eficiente en la etapa de desarrollo. Utilizar una maqueta de pruebas proporciona las siguientes ventajas al programador:

- Evita problemas ajenos al funcionamiento del protocolo de comunicación. Debido a que esta maqueta está desarrollada exclusivamente para emular la comunicación entre la placa *Zedboard* y *Arduino*. Se evita que errores que se puedan deber al funcionamiento de otros algoritmos enturbien el desarrollo del protocolo.
- Elude utilizar más recursos de los necesarios. De esta forma se evita perder tiempo en la configuración de la placa *Zedboard*, ya que mediante la utilización del puerto serie del ordenador, en el cual implementamos el código, es suficiente para poder comprobar el funcionamiento de la comunicación.
- Reduce el tiempo de diseño.

A continuación se va a proceder a explicar el funcionamiento de la maqueta aplicado al protocolo de comunicación.

Funcionamiento de la maqueta



fritzing

Figura 5.8: Maqueta de pruebas del protocolo de comunicación

Los dos leds de color azul representan los motores del vehículo. El led rojo representa el contador de las vueltas completas que realizan las ruedas de los motores. Y el sensor A3144 representa el sensor que está instalado en la rueda para contar cada vuelta que realizan. El vehículo posee dos sensores de efecto Hall, uno para cada rueda, y cada rueda llevará un imán de neodimio para activar el sensor y, de esta manera, contar las vueltas realizadas.

Para monitorizar el conteo de vueltas se utiliza el led de color rojo. Este se apaga cada vez que se aproxima el imán al sensor de tal forma que nos permite observar a simple vista si el contador se incrementa.

Los dos leds de color azul emulan la activación de los dos motores del vehículo. Uno representa el motor izquierdo y otro el derecho. Estos leds nos sirven para saber en qué dirección se va a mover el vehículo. Ya que por ejemplo si se encienden los dos leds, significa que se está realizando un movimiento hacia delante. Solo se encienden los leds cuando se activan los motores por lo que nos permite saber si ha recibido la información cuando se enciende alguno de los dos leds. Y también permite saber cuando se ha terminado una orden si están los dos leds apagados.

5.8 Desarrollo del firmware para el control del vehículo

La programación del vehículo realizará en el lenguaje *Arduino*. Dado que se utiliza un protocolo específico para este proyecto es necesario implementar una estructura para extraer la información de los paquetes de información en variables que el sistema pueda manejar. Como se explicó en la sección anterior la información viene estructurada en tres bloques. En primer lugar se comprueba si la información correspondiente al primer byte (ID) es igual al paquete de información que se espera. Es decir, si el movimiento que el vehículo espera, cronológicamente, es el primero, el identificador debe ser igual a 0. De no ser así el sistema no tratará la información de ese paquete ya que la información no corresponde al movimiento esperado. Una vez el identificador corresponde al paquete de información esperado se extrae la información del segundo byte correspondiente a la acción que se quiere realizar. Recordar que el segundo byte de información posee datos de la acción que se quiere realizar y del número de movimientos. Por ello se debe aplicar una máscara para extraer los tres bits más significativos correspondientes a la acción de movimiento.

Si extraemos la información de esta forma se puede proceder a realizar los movimientos correspondientes al paquete de información recibido.

Luego se procederá a activar los motores y con ellos el contador correspondiente al número de vueltas que realizan las ruedas. Cuando el contador alcance el número de vueltas correspondientes al paquete de información, el programa devolverá un ACK correspondiente a la realización de la orden. Si por algún problema el vehículo no logra realizar los movimientos esperados, el timeout se cumplirá y se comunicará a la placa *Zedboard* que el movimiento no ha sido realizado correctamente. El sistema lanzará de nuevo el proceso principal para que calcule una nueva trayectoria correspondiente al estado actual del escenario. De no ser así el vehículo podría haber realizado un número de vueltas inferior al esperado y estar colocado en un lugar distinto del que el sistema tiene calculado en su trayectoria, por lo que la información se quedaría desfasada y no correspondería a la situación del momento.

Como la función *loop* se ejecuta en bucle, el vehículo irá realizando movimientos a medida que las órdenes vayan siendo recibidas. Con la comprobación del identificador se controla que la orden es la que se espera y el coordinador de la placa *Zedboard* será el encargado de ir enviando las órdenes en función de que se vayan cumpliendo. De esta forma, cuando el vehículo complete la trayectoria no recibirá más órdenes. A no ser que se proponga otro objetivo.

5.9 Integración de los algoritmos en el sistema

Este proyecto está formado por una serie de algoritmos que en su conjunto forman el sistema. Antes de la integración de todos los algoritmos se ha realizado la implementación de cada uno de ellos de forma independiente. De acuerdo a la metodología ágil que se ha seguido se ha intentado modularizar el proyecto por objetivos. Cada objetivo ha sido desarrollado

hasta que se han conseguido los resultados esperados. De esta forma nos permite asegurar que, antes de la integración en el sistema, cada uno de estos algoritmos funciona de forma correcta. Este hecho evita muchos dolores de cabeza a la hora de afrontar los errores que se dan lugar en la fase de desarrollo.

Para poder integrar todos los algoritmos se ha decidido realizar una planificación acerca de cómo deben estar estructurados los datos de entrada y los resultados de cada algoritmo. A continuación se va a proceder a explicar cómo están organizados en función de los objetivos del sistema.

1. Distinción del vehículo sobre los demás objetos. Como argumento de entrada de este algoritmo se utilizará una imagen en formato JPG (Joint Photographic Graphics)⁷ en la cual aparezca el coche en el entorno, listo para calcular la trayectoria desde la ubicación en la que se encuentra. Los resultados de este algoritmo generarán un fichero de texto en el cual se guardará la información correspondiente a la coordenada en la que se encuentra el coche.
2. Detección de obstáculos. Para poder detectar los obstáculos es necesario que se utilicen dos imágenes como argumentos del algoritmo. Estas imágenes están en formato JPG, pero el algoritmo está implementado de tal forma para que se puedan utilizar distintos tipos. Como por ejemplo, BMP, TIFF (Tagged Image File Format)⁸, PNG (Portable Network Graphics)⁹... Los resultados obtenidos serán reflejados en un fichero de texto, cuya extensión es .txt, y mediante el cual se reflejará el mapa binario en el que los obstáculos se representan con 1 y el camino libre con 0.
3. Cálculo de trayectoria. Este algoritmo utilizará los datos contenidos en dos ficheros de texto. Uno de ellos corresponderá al mapa binario que se dio resultado en la ejecución del algoritmo de detección de obstáculos, mientras que el otro fichero guardará la información correspondiente a la coordenada de origen en la que se encuentra el vehículo. Como resultado de la aplicación del algoritmo se generará un fichero de texto en el cual se integre en el mapa la representación de la trayectoria que debe realizar el vehículo desde el punto de origen al destino. Esta trayectoria se representará mediante el número 3 para reflejar el comienzo, el número 2 para reflejar la ruta y el número 4 para el destino. De esta forma el mapa representará la ruta y los obstáculos contenidos en el entorno. Por otra parte se generará un fichero de texto en el cual se almacenarán todas las coordenadas correspondientes a todos los puntos que ha recorrido la trayectoria desde el inicio hasta su fin. Además, como forma de monitorización de los resultados también se generará una imagen en formato JPG en la cual se distinga el

⁷JPG es un formato de compresión de imágenes, tanto en color como en escala de grises, con alta calidad

⁸Formato de archivo de imágenes etiquetada. Un formato de imagen de alta resolución basado en etiquetas.

TIFF se utiliza para el intercambio universal de imágenes digitales.

⁹PNG (Portable Network Graphics) es un formato gráfico basado en un algoritmo de compresión sin pérdida para bitmaps no sujetos a patentes.

CAPÍTULO 5. DESARROLLO

mapa y la trayectoria mencionadas anteriormente.

4. Generación de movimientos mediante datos de la trayectoria. Para la ejecución de este algoritmo es necesario un fichero de texto en el cual se representen todos los movimientos desde el inicio hasta el final de la trayectoria. Como resultado, el algoritmo no generará ningún fichero como tal pero establecerá la comunicación con el coche para que vaya ejecutando los movimientos correspondientes a la trayectoria.
5. Movimiento del vehículo. Para la ejecución de este algoritmo es necesario que se establezca una comunicación entre la placa *Zedboard* y *Arduino*. De tal forma que la información que reciba el coche sea interpretada para realizar los movimientos correspondientes a la orden.

Teniendo en cuenta los argumentos y resultados recientemente enumerados, se debe realizar una implementación general del sistema en la que se incluyan todos los algoritmos de forma modular. Es decir, que exista una única función *main* desde la cual se llame a las funciones que realizan los objetivos del sistema. De esta forma si en la ejecución ocurre algún error es fácil comprobar en qué llamada ha ocurrido ya que aparecerá el nombre de la función en cuestión y será sencillo comprobar qué ocurre. Además, de esta forma el proyecto se encuentra organizado por funcionalidades lo cual permite ejecutar los algoritmos del sistema que se deseen sin necesidad de comentar o eliminar el código que no se quiera utilizar.

Una vez conseguido el objetivo de la integración, se realizó una medición de los tiempos de ejecución del sistema y se comprobó que todos los algoritmos tienen un tiempo de ejecución menor a un segundo excepto el algoritmo de cálculo de trayectoria. Se podría considerar el cuello de botella del sistema. En el cuadro 5.2 se pueden observar algunos de los tiempos de ejecución calculados:

	Coordenadas		Tiempos Ejecución	
	Inicio	Final	4 direcciones	8 direcciones
Diagonal	(0,0)	(1919,1079)	4.27 min	0.38 seg
Diagonal	(0,1079)	(1919,0)	31.15 min	0.40 seg
Centro	(959,539)	(961,541)	0.28 seg	0.37 seg
Centro	(959,541)	(961,539)	0.29 seg	0.34 seg
Abajo Arriba	(959,0)	(961,1079)	9.62 seg	3.20 seg
Arriba Abajo	(961,1079)	(959,0)	2.102 seg	2.57 min
Izq Dcha	(0,539)	(1919,541)	0.37 seg	1.60 seg
Dcha Izq	(1919,541)	(0,539)	0.34 seg	0.37 seg

Cuadro 5.2: Tiempos de ejecución en software

Como se puede observar en algunos casos el tiempo de ejecución es bastante mayor a un segundo. Se aumenta el orden de magnitud a minutos en un par de casos. Aunque en el sistema final solo se calcula la trayectoria para 8 direcciones, hay un caso en el que tarda

alrededor de 3 minutos para calcular la trayectoria. Es por ello que se decidió realizar una optimización desarrollando un nuevo código que mediante el uso de hardware reconfigurable se obtienen tiempos de ejecución mejores.

Dado que el único algoritmo que suponía una limitación en cuanto al tiempo de ejecución fue el algoritmo A*, el resto se decidió que puede ejecutarse en el procesador ARM de la placa *Zedboard* utilizándose las librerías de *OpenCV* para un par de casos en los cuales se podría realizar una optimización. Cabe mencionar que el objetivo de este TFG se pudo realizar con la integración de los algoritmos que hemos visto hasta ahora sin optimizar. A continuación se va a proceder a desarrollar las optimizaciones que se han llevado a cabo en el sistema.

5.10 Optimizaciones realizadas en el sistema

5.10.1 Dilatación de obstáculos mediante *OpenCV*

Tras asistir a una charla acerca de procesamiento de imágenes que se propuso en el laboratorio ARCO, se planteo modificar el algoritmo de detección de objetos ya que la implementación de la delimitación de obstáculos es compleja. Uno de los temas que se abordó en la charla fueron los algoritmos de erosión y dilatación para el tratamiento de información de una imagen. En el momento, la charla no iba orientada a este TFG, sino que se mostraron ejemplos de cómo procesar la información de fotografías y ver cómo afectaban distintos tipos de operaciones en el resultado. Pero tras mencionar cómo funcionaban los algoritmos de erosión y dilatación se vislumbró la idea de adaptarlo al sistema.

Una de las razones por las que se delimitan los obstáculos del entorno en cuadrados o rectángulos es para evitar que el vehículo colisione debido a la irregularidad de los obstáculos. Es decir, que debido a su forma irregular el vehículo colisione con los objetos. Otra de las razones, si la distancia entre un objeto u otro es menor que las medidas del vehículo, el algoritmo de cálculo de trayectoria puede encontrar un ruta que en principio es óptima pero que ocasionaría una colisión debido a que el vehículo no cabe entre ambos objetos. ¿Qué ocurriría si se decide dilatar el grosor de los obstáculos en función de las medidas del vehículo?



Figura 5.9: Ejemplo de dilatación de objetos

La operación de dilatación consiste en aplicar una máscara que modifique los valores que

se encuentran alrededor del píxel objetivo. De tal manera que expanda el valor del píxel en función del tamaño que se le introduzca como argumento. Un ejemplo de dilatación sería el de la figura 5.9. La librería *OpenCV* posee un método que aplica la dilatación en función de un tamaño como parámetro. Si en dicho argumento se introduce el tamaño del vehículo, este algoritmo dilatará el valor de los objetos, es decir el valor 1, alrededor del píxel origen hasta que se corresponda con el tamaño del vehículo en todos sus píxeles vecinos. De esta forma mediante la utilización de la librería se puede evitar utilizar el algoritmo implementado anteriormente ya que la complejidad del proceso origina un peor rendimiento en comparación con la función de dilatación optimizada de *OpenCV*.

5.10.2 Tratamiento de la información de una imagen en varios formatos

Debido a que en un primer lugar la idea era embeber¹⁰ todo el sistema en la placa *Zed-board*, se evitó utilizar la librería de *OpenCV* para el tratamiento de los datos de una imagen. Ya que, como se introdujo en el capítulo de objetivos 2, el uso de las herramientas de *Vivado* poseen una serie de restricciones en los mecanismos de programación y el uso de librerías está limitado al soporte de la propia herramienta. Por ejemplo la librería de *OpenCV* ofrece muchas más funciones de las que la herramienta *Vivado HLS* da soporte [Res]. Como consecuencia de evitar utilizar *OpenCV*, se decidió implementar un programa para extraer la información de imágenes en formato BMP¹¹. Posteriormente se realizó otra implementación, pero esta vez utilizando la librería *OpenCV*.

La librería *OpenCV* nos proporciona el objeto tipo *Mat* que es de gran utilidad a la hora de trabajar con imágenes. Ya que nos permite interactuar con una gran cantidad de funciones con las que podemos realizar infinidad de operaciones. [Mat] En este proyecto, para el tratamiento de la información de una imagen, se ha utilizado la función *imread*. [ImR]

Con el uso de esta función solo tenemos que preocuparnos de introducir la ruta en la que se encuentra la imagen como primer argumento. Y como segundo argumento cualquiera de estas tres opciones:

1. CV_LOAD_IMAGE_UNCHANGED. Si se quiere cargar el valor alpha, normalmente hace referencia a la intensidad, en el objeto *Mat*. Es decir, cuatro bytes de información por píxel. Uno para el canal alpha y tres para el modelo de color RGB.
2. CV_LOAD_IMAGE_GRAYSCALE. Si se quiere guardar la información de la imagen en formato de escala de grises. Es decir, un byte de información por píxel.
3. CV_LOAD_IMAGE_COLOR. Si se quiere guardar la imagen en modelo de color RGB. Es decir, tres bytes por píxel.

¹⁰Un sistema embebido o empotrado (integrado, incrustado) es un sistema de computación diseñado para realizar una o algunas pocas funciones dedicadas, frecuentemente en un sistema de computación en tiempo real.

¹¹BMP es un formato de imagen de mapa de bits, propio del sistema operativo Microsoft Windows.

Con esta función, con una simple línea de código se interpreta la información de la cabecera, los datos en crudo de la imagen y hasta se puede cambiar el número de canales que posee la imagen en la misma llamada. Para la detección de objetos el algoritmo utiliza como argumentos las imágenes en escala de grises con formato JPG (Joint Photographic Graphics), que es el formato de salida por defecto de la cámara cenital que se utiliza en el proyecto. Por lo que no es necesario realizar ningún tipo de conversión. Además, la función *imread* permite utilizar los formatos BMP, JPEG (Joint Photographic Experts Group), TIFF, PNG y algunos más que no son necesarios mencionar. [ImR]

5.10.3 Segmentación de la imagen en macro-bloques

Como consecuencia de la utilización de imágenes *FullHD*, el mapa utilizado en el algoritmo de cálculo de trayectoria es grande, ya que la cantidad de nodos por los que se puede recorrer la trayectoria del vehículo es superior a 2 millones. Esto implica que los tiempos de ejecución del algoritmo A* se disparen en función del escenario en el que se encuentren. Bien tardando segundos o minutos en calcular la trayectoria. Es por ello que se ha decidido segmentar la información del mapa en macro-bloques que guarden la información relevante de la imagen, en este caso los obstáculos.

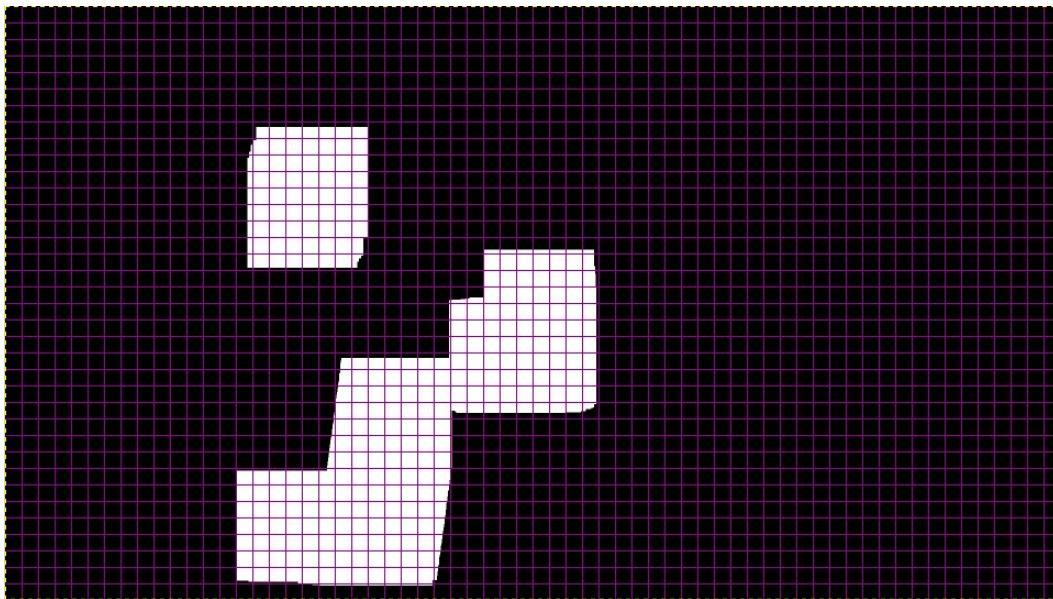


Figura 5.10: Ejemplo de segmentación de imagen en macro-bloques

La división del escenario en bloques permite que se pueda re-escalar la imagen a una resolución menor. Ahora bien, cada píxel de la nueva imagen contiene la información de interés de la sección a la que corresponde en el escenario. Esta información hace referencia a la cantidad de obstáculos que se encuentran en dicha sección. Como tras la aplicación del algoritmo de dilatación se genera un mapa binario en el que un 1 corresponde a un obstáculo y un 0 corresponde a una posición por la que el vehículo puede pasar, se decidió realizar una sumatoria de todos los valores que se encuentran en el bloque para saber si una sección

está llena de obstáculos o por el contrario vacía. Al utilizar una resolución *FullHD* se ha buscado una resolución menor con la que se pueda guardar la proporción de la imagen. Esta resolución es 64x36, ya que no genera ningún valor flotante con el que podríamos perder información. Con dicha resolución, pasamos de tener más de 2 millones de píxeles a 2048 píxeles por imagen, cada bloque tiene un tamaño de 30x30 píxeles. Si la sección correspondiente al mapa tiene un valor equivalente a 900 significa que este bloque está completamente ocupado por un obstáculo. Mientras que si por el contrario tiene un valor 0 está completamente libre de obstáculos. Se puede dar el caso intermedio en el que el valor de dicho bloque se encuentre entre 900 y 0. Cuanto más se acerque a 900 más completo estará de obstáculos mientras que cuanto más bajo sea menos obstáculos se encontrarán en el bloque. Esto genera un inconveniente, ya que aunque el valor del bloque sea 30 puede ser que el obstáculo que aparece en la sección bloquee el camino de ese nodo. Es por ello, que para evitar posibles problemas se ha decidido que si el valor de un bloque es mayor que 0 se asigne un 1. De esta forma la ruta calculada no es óptima en cuanto a distancia, pero se reduce mucho el tiempo de ejecución del algoritmo de cálculo de trayectoria ya que hemos pasado de manejar más de 2 millones de nodos a 2048 nodos para el peor caso de pruebas.

De esta forma se ha reducido el tamaño del mapa que se pasa al algoritmo de cálculo de trayectoria. Esto implica que los tiempos de ejecución mejoren mucho ya que el número de recursos que utiliza es mucho menor. La ruta que genera no es óptima en cuanto a la distancia pero es muy similar a la que se calcula en la resolución *FullHD*.

5.10.4 Implementación algoritmo A* según restricciones de *Vivado HLS*

Una vez llegados a este punto se decidió embeber en hardware reconfigurable el algoritmo de cálculo de trayectoria con la intención de optimizar el tiempo de ejecución. Para ello se hará uso de la herramienta *Vivado HLS*, mediante la cual generaremos un componente IP con la funcionalidad del algoritmo para ser implementado en la FPGA.

Para poder crear este componente es necesario tener en cuenta la restricción principal de la herramienta *Vivado HLS*.

Restricción No se puede utilizar asignación dinámica de memoria.

Como consecuencia de esta limitación ha sido necesario implementar desde cero el algoritmo de cálculo de trayectoria ya que en un principio, como se dijo en la sección 5.4, se utilizó una modificación de una implementación del algoritmo A* que se encontró en internet. [FB310]

Los algoritmos de cálculo de trayectoria utilizan por defecto asignación dinámica de memoria para crear y destruir los nodos en función de la búsqueda. Es decir, cada vez que se comprueba algún nodo adyacente se reserva memoria para la información del nodo. Mientras que si por el contrario el nodo que se ha visitado no es necesario mantenerlo en memoria se

Estructura de datos tipo Nodos	
VARIABLES	BITS
Coordenada X	6
Coordenada Y	6
Posición del padre	12
Coste	14
Heurística	7
F (Coste + Heurística)	21
Visitado	1

Cuadro 5.3: Estructura de datos de cada nodo del mapa.

elimina. Este tipo de operaciones no se pueden realizar ya que al embeber el algoritmo en hardware la placa no puede gestionar dinámicamente la memoria. Para ejecutar el algoritmo es obligatorio definir previamente que memoria debe reservar el sistema. Por ello, como primer paso para desarrollar la nueva implementación del algoritmo se decidió definir una estructura de datos con la cual se pueda acotar la cantidad máxima de memoria necesaria para la resolución del peor caso posible.

Según el cuadro 5.3 cada nodo va a poseer una estructura de datos tipo Nodos. Este tipo de estructura nos permite definir una plantilla sobre la cual reservar la estructura de memoria necesaria para cada nodo del mapa. Gracias a esta definición se puede generar un vector unidimensional de nodos. Cada celda de este vector está representada por la estructura de datos tipo Nodos, sobre la cual se irá añadiendo la información correspondiente a los elementos del mapa. De esta forma podemos reservar la memoria de forma estática previamente a la ejecución del algoritmo y posteriormente, en tiempo de ejecución, ir actualizando las variables en función de la posición en la que se encuentre el nodo. Ahora bien, para poder ir actualizando los valores de la estructura tipo Nodos es necesaria la implementación de una función con la que calcular la posición de los nodos adyacentes a la posición inicial, de tal manera que cada vez que el algoritmo A* seleccione un nodo por el cual avanzar se calculen los nodos adyacentes al nodo actual.

Para calcular la posición unidimensional de los nodos adyacentes al nodo actual es necesario tener en cuenta una serie de premisas. Por ejemplo, hay que tener en cuenta los bordes del mapa. Es decir, según la posición en la que se encuentre el nodo actual no se pueden expandir todos los nodos adyacentes ya que puede ser que se encuentren fuera del mapa. Si nos encontramos en la posición unidimensional 0, la cual hace referencia a la esquina superior izquierda, hay una serie de nodos que no podemos expandir. Estos nodos serían los que hicieran referencia a los movimientos arriba, izquierda, diagonal superior izquierda, diagonal superior derecha y diagonal inferior izquierda. Existen más restricciones como esta, por ejemplo las esquinas restantes y si el nodo actual se encuentra en cualquier límite del mapa. Cada una de estas situaciones afecta a una posición adyacente distinta por lo que se ha deci-

CAPÍTULO 5. DESARROLLO

dido separar los nodos adyacentes por coordenada x e y . De esta forma primero se calcula la posición unidimensional correspondiente a la coordenada x e y que se pasa como argumento al método para posteriormente invalidar el valor de la posición si esta no puede expandirse debido a los límites del mapa. A continuación veamos un ejemplo de cómo se calculan los nodos adyacentes a la posición $x = 0$, véase el listado de código 5.1

```
void adjacentNodes(int x, int y){
    int initPosition = x+(y*HORIZONTAL);
    //Left
    adjacentPosition[0][2] = initPosition-1; //Position
    //Right
    adjacentPosition[1][2] = initPosition+1; //Position
    //Top
    adjacentPosition[2][2] = initPosition-HORIZONTAL; //Position
    //Bot
    adjacentPosition[3][2] = initPosition+HORIZONTAL; //Position
    //Upper Left Diagonal
    adjacentPosition[4][2] = initPosition-HORIZONTAL-1; //Position
    //Upper Right Diagonal
    adjacentPosition[5][2] = initPosition-HORIZONTAL+1; //Position
    //Bottom Left Diagonal
    adjacentPosition[6][2] = initPosition+HORIZONTAL-1; //Position
    //Bottom Right Diagonal
    adjacentPosition[7][2] = initPosition+HORIZONTAL+1; //Position

    //if nodes are not adjacent set -1 on position.
    if(x==0){
        //Left
        adjacentPosition[0][2] = -1; //NotAdjacent
        //Right
        adjacentPosition[1][0] = x+1; //X
        //Top
        adjacentPosition[2][0] = x; //X
        //Bot
        adjacentPosition[3][0] = x; //X
        //Upper Left Diagonal
        adjacentPosition[4][2] = -1; //Not adjacent
        //Upper Right Diagonal
        adjacentPosition[5][0] = x+1; //X
        //Bottom Left Diagonal
        adjacentPosition[6][2] = -1; //Not adjacent
        //Bottom Right Diagonal
        adjacentPosition[7][0] = x+1; //X
    }else if...
```

Listado 5.1: Parte del algoritmo que calcula los nodos adyacentes

Como se puede observar, para almacenar los datos correspondientes a los ocho nodos adyacentes al nodo actual se ha generado una matriz bidimensional en la que la posición de la primera dimensión corresponde al identificador de los ocho posibles nodos adyacentes. La posición de la siguiente dimensión hace referencia a la coordenada X, coordenada Y y posición unidimensional en el vector de estructuras tipo Nodos, cuadro 5.3. Una vez que sabemos

como está estructurada la información de los nodos adyacentes se puede observar que al comienzo se calcula la posición equivalente a la coordenada X e Y en el vector unidimensional. Como hay ocho posibles nodos adyacentes se calcula la posición correspondiente a las ocho direcciones posibles. Izquierda, derecha, arriba, abajo, diagonal superior izquierda y derecha y diagonal inferior derecha e izquierda. Posteriormente se comprueba si el nodo actual se encuentra en alguno de los límites del mapa. En este caso solo aparece $x = 0$ debido a la longitud del código, pero habría un caso para $x = HORIZONTAL - 1$, para los valores intermedios entre $x = 0$ y $x = HORIZONTAL - 1$ y lo mismo para la coordenada y. Es decir, $y = 0$, $y = VERTICAL - 1$ y los valores intermedios. Para $x = 0$ los nodos adyacentes que no se pueden expandir son los de la izquierda, diagonal superior izquierda y diagonal inferior izquierda. Ya que son las posición en los que se resta uno al valor de la x. Y como vale 0, no puede ser negativo. Para los valores que no se pueden expandir se actualiza el valor a -1 , el cual nos va a servir para identificar si es un nodo que hay que evitar en el futuro.

Una vez conocemos los nodos adyacentes se puede proceder a aplicar el algoritmo A*. Los pasos que sigue el algoritmo son los siguientes:

1. Inicializamos el vector unidimensional de estructuras tipo Nodos.
2. Inicializamos el nodo inicial de búsqueda. Correspondiente a la coordenada en la que se encuentra el vehículo.
3. Calculamos los nodos adyacentes al nodo inicial.
4. Calculamos cuál es la posición del padre, que coste tiene avanzar a la posición adyacente y la heurística que evalúa la distancia entre la posición actual y el destino.
5. Calculamos la mejor opción de los nodos adyacentes.
6. Establecemos la mejor opción como visitada.
7. Inicializamos el bucle que realiza la búsqueda hasta que llega a la posición final.
 - a) Calculamos los nodos adyacentes al nodo actual.
 - b) Calculamos el parent, el coste y la heurística del movimiento.
 - c) Elegimos la mejor opción entre la lista de nodos abiertos. Incluye los nodos adyacentes y los nodos calculados previamente que no han sido visitados. Esta implementación de una lista de nodos abiertos es la que nos permite asegurar que el resultado obtenido es óptimo. Ya que va desarrollando la ruta en función del mejor valor, no del mejor adyacente.
 - d) Establecemos la mejor opción como visitada para evitar volver a pasar.
8. Una vez se ha calculado la ruta nos situamos en la posición final y vamos preguntando quién es el parent del nodo actual hasta que llegamos a la posición inicial. De esta forma se guarda la trayectoria que se ha seguido.

CAPÍTULO 5. DESARROLLO

Para la implementación de este algoritmo se han ido desarrollando las funciones en función de la lista de pasos comentada previamente. Como base teórica del algoritmo se ha seguido la explicación del algoritmo, sin implementación, de la página web [APr]. A continuación se va a proceder a explicar cómo se ha resuelto cada uno de estos pasos.

Inicialización del vector unidimensional de estructuras tipo Nodos

Dado que el archivo sobre el cual obtenemos los datos del mapa viene representado de forma bidimensional se decidió implementar un algoritmo con el cual se pudiera inicializar la matriz unidimensional, de estructuras tipo Nodos, a través del archivo bidimensional. Para ello, se pensó en recorrer la matriz en función del tamaño, *horizontal * vertical*, de esta forma se recorre la misma distancia para el formato bidimensional y unidimensional. Se debe tener en cuenta que cada vez que se recorra el tamaño correspondiente al tamaño horizontal se debe incrementar el valor de la coordenada Y. De esta forma se recorrerá la matriz unidimensional y se podrán guardar los valores correspondientes a la coordenada X e Y del mapa. Los valores de la coordenada X e Y se utilizan para consultar en el mapa si en la coordenada correspondiente hay un obstáculo o es camino transitable. Si el camino es transitable se inicializa el valor de visitado a 0, mientras que si es un obstáculo se inicializa a 1. Así se evita calcular la información correspondiente al nodo. Como resultado obtenemos el vector unidimensional de estructuras tipo Nodos con las coordenadas X e Y correspondientes a su posición y la información acerca de si los nodos se pueden visitar o no.

Inicialización del nodo inicial de búsqueda

El nodo inicial de búsqueda es el correspondiente a las coordenadas iniciales en las que se encuentra el vehículo. Por lo que es necesario extraer la información del fichero que genera el algoritmo de detección del vehículo de la sección 5.5. Una vez se obtienen las coordenadas X e Y se calcula la posición unidimensional correspondiente, como aparece al comienzo del listado 5.1, y se inicializa a 1 el campo correspondiente a visitado.

Cálculo de los nodos adyacentes al nodo inicial

Se llama a la función *adjacentNodes*, mencionada anteriormente en el listado 5.1, con las coordenadas X e Y en las que se encuentra el vehículo como argumentos.

Actualizamos la información de los nodos adyacentes

Para calcular y actualizar la información de los nodos se deben recorrer los 8 nodos adyacentes posibles. Una vez se comprueba si el nodo se puede visitar, bien porque no sea un obstáculo o porque no se encuentre en el límite del mapa, se procede a incluir el nodo en la lista de nodo abiertos. Posteriormente se actualiza la información del padre, que en este caso es el nodo en el que se encuentra el vehículo. Se calcula la heurística con la función de *Manhattan*, la cual actualiza el valor según el código del listado 5.2. A continuación se

calcula el coste al que equivaldría la ejecución del movimiento al nodo adyacente. Este coste es acumulativo, pero como estamos al principio se suman 10 si el movimiento es ortogonal y 14 si el movimiento es diagonal. Por último se suma el coste y la heurística y se guarda en la variable *costPlusHeuristic*.

```
void manhattanHeuristic(int x, int y, int position, int xFinish, int yFinish){
    int xDistance, yDistance;
    xDistance = xFinish - x;
    yDistance= yFinish - y;

    node[position].heuristic = abs(xDistance)+abs(yDistance);
}
```

Listado 5.2: Función con la que se calcula la heurística

Calculamos la mejor opción de los nodos adyacentes

Una vez se ha actualizado la información de los 8 nodos adyacentes al nodo actual se realiza una comparación entre los valores de la variable *costPlusHeuristic* de cada nodo que aparezca en la lista de nodos abiertos. Esta lista puede contener los nodos adyacentes y los nodos que se han calculado previamente. Como en este caso solo hemos calculado los nodos adyacentes del punto inicial, la lista de nodos abiertos estará formada solo por los adyacentes. Al comprobar si el nodo se encuentra en la lista abierta se evita contar con nodos que no son accesibles o ya están visitados. Si el valor de la variable *costPlusHeuristic* es menor que el mejor valor guardado previamente se sustituye y se almacena la posición correspondiente al mejor nodo. De esta forma tras recorrer todos los nodos de la lista abierta se devuelve la posición correspondiente al mejor nodo por el que continuar la trayectoria. Véase el listado de código 5.3.

```
int calculateBestOption(){
    int bestOption = 2147483647;
    int positionBestOption;

    for (int i = 0; i < (HORIZONTAL*VERTICAL); i++) {
        if(openNodes[i]==0){ //If it is an open node
            if( node[i].costPlusHeuristic < bestOption ){ //If is a better option...
                bestOption = node[i].costPlusHeuristic;
                positionBestOption = i;
            }
        }
    }

    return positionBestOption;
}
```

Listado 5.3: Función con la que se calcula el mejor nodo

Establecer la mejor opción como visitada

Una vez se ha calculado cual es la posición sobre la que debe avanzar el vehículo se actualiza la variable $visited = 1$, De esta forma se evita volver a pasar por el mismo nodo en futuros movimientos.

Inicialización del bucle para calcular la trayectoria

Los pasos mencionados anteriormente se han aplicado para inicializar la búsqueda, a continuación se debe establecer un bucle hasta que se cumpla la condición de que el nodo actual corresponda a la posición correspondiente al destino.

El proceso que se lleva a cabo en el bucle es el mismo, existe una pequeña variación ya que la posición que se envía como argumento a los métodos es distinta en función del avance de la trayectoria. Por ello cada iteración del bucle genera unos nuevos nodos adyacentes a la posición actual que se añaden a la lista de nodos abiertos que debe valorar el algoritmo.

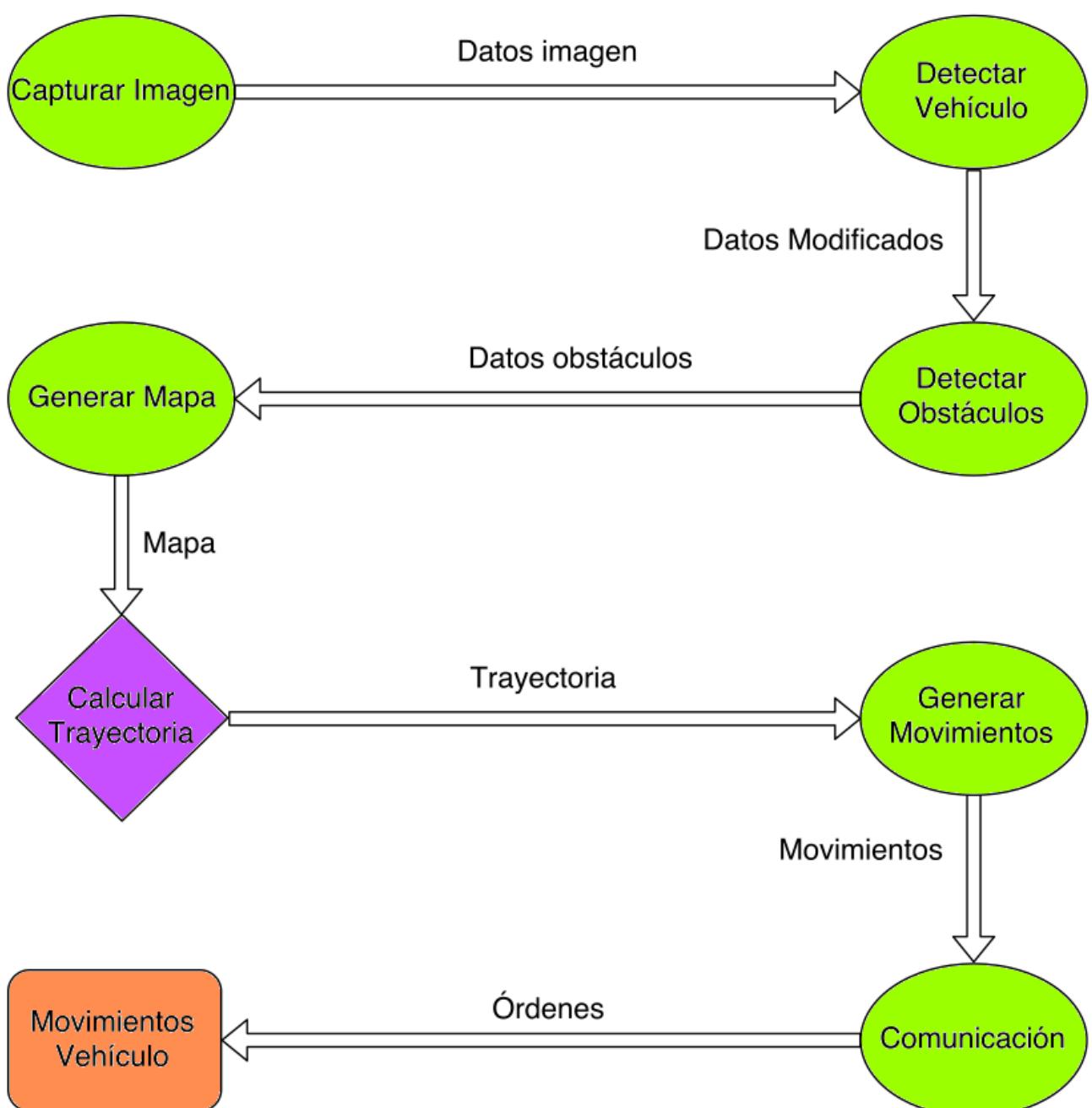
5.11 Integración del sistema en la placa *Zedboard*

Para realizar la integración del sistema en la placa *Zedboard* se ha seguido la estructura del diagrama de procesos de la figura 5.11. Como se puede observar se ha decidido embeber únicamente el algoritmo de cálculo de trayectoria en FPGA. Mediante el uso de la herramienta *Petalinux* se ha realizado una compilación cruzada desde un ordenador diferente a la placa. La compilación de la aplicación en el propio procesador ARM es muy compleja. Esto es debido a que la librería *OpenCV* tiene muchas dependencias en cuanto al uso de librerías externas y al utilizar un procesador ARM tan específico el soporte de la distribución es escaso. Como solución se decidió crear un directorio sobre el cual se pudiera emular la distribución *Debian* que se ha utilizado en la placa *Zedboard*. De esta manera mediante el uso de un fichero *Makefile* y la ayuda de la herramienta *Petalinux* se puede realizar la compilación de la aplicación del sistema generando el archivo binario que se debe trasladar a la placa *Zeboard* para su ejecución.

Con la herramienta *Vivado HLS* se genera el componente IP con la funcionalidad del algoritmo A*. Este componente IP será cargado al catálogo IP de la herramienta *Vivado* para realizar su posterior integración en la placa *Zedboard*. Finalmente, con la herramienta SDK se invocan funciones necesarias para la utilización del componente referente al algoritmo de cálculo de trayectoria.



Diagrama de procesos del sistema



Capítulo 6

Resultados

A Continuación se va a proceder a mostrar los resultados de cada uno de los algoritmos que se han desarrollado para el sistema. Como se han realizado optimizaciones para el uso de hardware reconfigurable, al final, se realizará una comparación acerca de los tiempos de ejecución que se han obtenido.

6.1 Detección de objetos

En la figura 6.1 se pueden observar los resultados de la aplicación del algoritmo de detección de objetos. Este algoritmo tiene dos fases. La resta de imágenes correspondiente a la figura 6.1b y la delimitación de objetos correspondiente a la figura 6.1c.



Figura 6.1: Ejemplo de aplicación del algoritmo de detección de objetos

6.2 Detección del vehículo frente a los objetos

En la figura 6.2 se puede observar como funciona el algoritmo de detección del vehículo frente a los objetos. El círculo verde que aparece sobre el coche ha sido incluido por el algoritmo. La imagen original es exactamente igual salvo con el círculo verde. En la imagen de la izquierda se puede observar un círculo blanco sobre la imagen que corresponde a la posición del vehículo. Se calcula el punto medio del círculo y se genera el fichero correspondiente que guarda las coordenadas.

6.3 Modificación algoritmo A* [FB310]

En la figura 6.3 se puede observar el cálculo de la trayectoria para el escenario con objetos 6.3a. La línea de color verde corresponde a la ruta.



Figura 6.2: Ejemplo de detección de la posición del vehículo en el entorno.

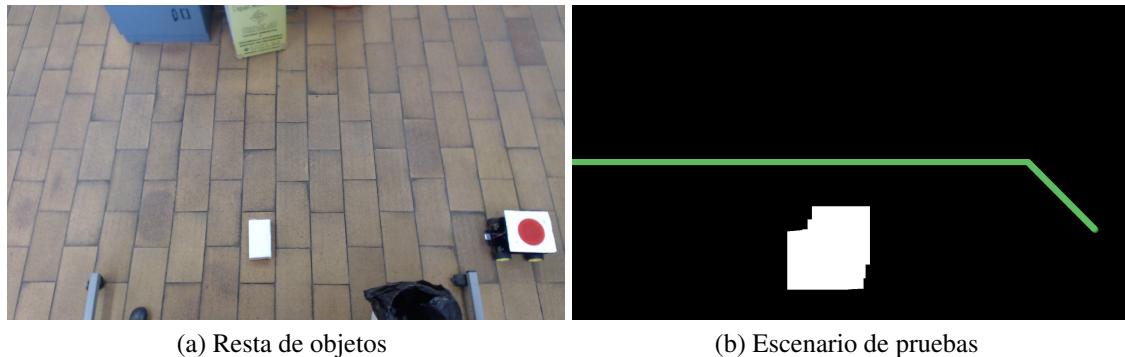


Figura 6.3: Ejemplo de cálculo de trayectoria

6.4 Comunicación

La comunicación por el puerto serie tiene como resultado los cuatro casos distintos que aparecen en la figura 6.4.

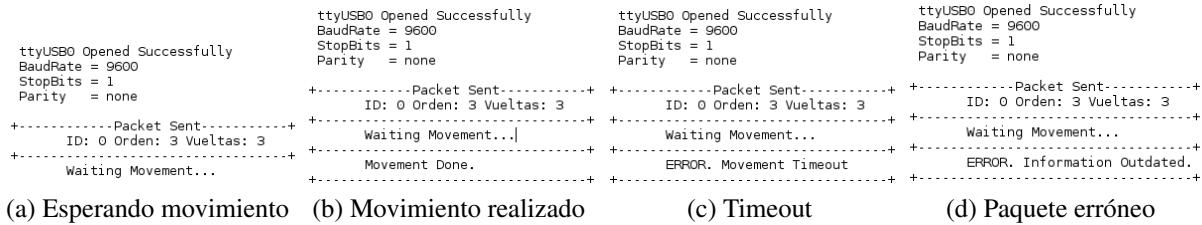


Figura 6.4: Comunicación por el puerto serie

6.5 Coordinador de los movimientos del vehículo

Para los resultados del coordinador de los movimientos del vehículo se ha decidido mostrar la figura 6.5 que corresponde a la trayectoria que debe seguir el vehículo y una captura de pantalla de los paquetes que han sido enviados por el puerto serie para realizar el movimiento.

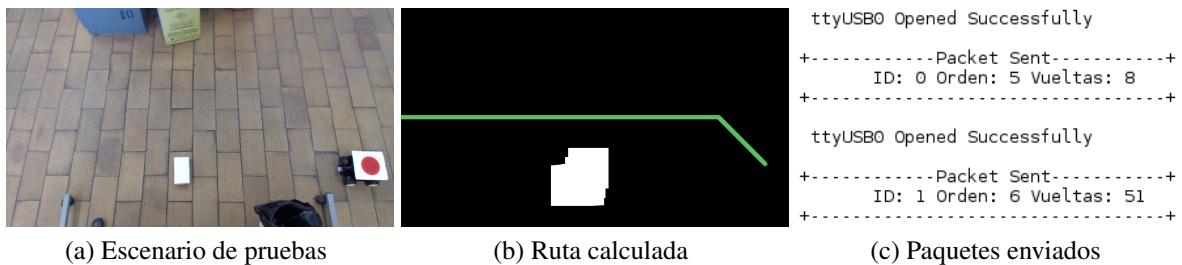


Figura 6.5: Ejemplo de envío de movimientos al vehículo

6.6 Integración del sistema

A continuación se han expuesto en el cuadro 6.1 los tiempos de ejecución del sistema para el escenario de la figura 6.6. Como se puede observar el cuello de botella de nuestra aplicación es el algoritmo de cálculo de trayectoria, el cual puede llegar a limitarnos, en algunas circunstancias, la ejecución en tiempo real debido a sus largos tiempos de cálculo. Por ello, se ha decidido realizar una serie de optimizaciones en el sistema para mejorar los tiempos de ejecución y la versatilidad en cuanto a los formatos de imagen que se utilizan como argumentos.



Figura 6.6: Escenario con el que se ha realizado la medición de los tiempos

Tiempos de ejecución del sistema	
Detección vehículo	0.30 seg
Detección objetos	0.50 seg
Cálculo de trayectoria	7.66 min

Cuadro 6.1: Tiempos de ejecución del sistema sin optimizar

6.7 Optimizaciones realizadas

Se va a proceder a enumerar los resultados obtenidos tras la aplicación de las optimizaciones realizadas en el sistema. En el cuadro 6.2 se muestran los tiempos de ejecución del sistema optimizado para el caso de estudio de la figura 6.6.

Tiempos de ejecución del sistema	
Detección vehículo	0.30 seg
Detección objetos	0.41 seg
Cálculo de trayectoria	0.0030 seg

Cuadro 6.2: Tiempos de ejecución del sistema optimizado

6.7.1 Tratamiento de imágenes en varios formatos

Originalmente el sistema estaba diseñado para utilizar imágenes en formato BMP, pero mediante la utilización de la librería *OpenCV* se ha logrado ampliar el número de formatos permitidos a BMP, JPEG, TIFF y PNG entre otros. Actualmente se utilizan imágenes en formato JPG debido a que el tamaño de las imágenes varía de 3,7 MB por imagen *FullHD* a color en formato BMP a unos 300 KB por imagen *FullHD* a color en formato JPG. Además, el uso de la función *imread* genera una matriz tipo *Mat* que permite interactuar con muchos tipos de funciones para tratamiento de imágenes.

6.7.2 Detección de obstáculos con *OpenCV*

Tras haber desarrollado el algoritmo de delimitación de obstáculos se realizaron una serie de pruebas y se comprobó que en algunos casos no daba los resultados esperados. Es por ello, que se buscó una solución más fiable a la delimitación de objetos mediante el uso de la librería *OpenCV*. Se decidió utilizar la función *dilate* ya que nos permite ampliar los objetos detectados en función de un tamaño como parámetro.

6.7.3 Segmentación de la imagen en macrobloques

Ya que el algoritmo de cálculo de trayectoria es la limitación principal en cuanto a la posibilidad de ejecutar el sistema en tiempo real se decidió buscar una optimización para mejorar el tiempo de ejecución del algoritmo. Para realizar una comparación entre algoritmos vease los cuadros 6.1 y 6.2.

6.7.4 A* optimizado para *Vivado HLS*

Debido a que el algoritmo de cálculo de trayectoria es el cuello de botella del sistema, se decidió embeber el algoritmo en hardware reconfigurable para mejorar el tiempo de ejecución. Para ello ha sido necesario realizar una nueva implementación del algoritmo que cumpla las restricciones de la herramienta *Vivado HLS*. Una vez lograda la implementación, se realizó la síntesis del algoritmo y estos son los datos más relevantes que nos ha mostrado la herramienta.

6.8 Integración del sistema en la placa *Zedboard*

Para esta sección se van a mostrar las estadísticas que ha generado la síntesis del algoritmo A* en la herramienta *Vivado HLS*.

CAPÍTULO 6. RESULTADOS

Como se puede observar en la figura 6.7 el tiempo ciclos de reloj estimado es de 8,48 ns. Gracias a que la suma entre el tiempo estimado e incertidumbre es menor que el objetivo se puede llevar a cabo el uso de una frecuencia de 100 MHz.

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.48	1.25

Figura 6.7: Periodo de reloj estimado para el algoritmo A*

A continuación se puede observar en la figura 6.8 el número de iteraciones por bucle que contiene el algoritmo A*. El bucle 4 y 5 son indeterminados debido a que la condición de finalización no es determinista. Ya que están relacionadas con la resolución del cálculo de la ruta.

Loop Name	Latency		Iteration Latency	Initiation Interval		Trip Count	Pipelined
	min	max		achieved	target		
- ForinitNodes	6912	6912	3	-	-	2304	no
- FOR1	32	88	4 ~ 11	-	-	8	no
- FORcalculateBestOption	4608	4608	2	-	-	2304	no
- Loop 4	?	?	4655 ~ 4719	-	-	?	no
+ FOR2	32	96	4 ~ 12	-	-	8	no
+ FORcalculateBestOption	4608	4608	2	-	-	2304	no
- Loop 5	?	?	4	-	-	?	no

Figura 6.8: Latencias para cada bucle del algoritmo A*

Por último se van a mostrar los recursos necesarios para la ejecución del algoritmo en la placa *Zedboard*, véase la figura 6.9.

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	878
FIFO	-	-	-	-
Instance	-	-	328	910
Memory	50	-	0	0
Multiplexer	-	-	-	650
Register	-	-	948	-
Total	50	0	1276	2438
Available	280	220	106400	53200
Utilization (%)	17	0	1	4

Figura 6.9: Recursos necesarios para la ejecución del algoritmo A*

Capítulo 7

Conclusiones

EN el trabajo realizado a lo largo de este TFG se ha cumplido el objetivo principal del mismo que consistía en conseguir que un vehículo llegue desde un punto de origen a un punto de destino esquivando los obstáculos que aparezcan en el escenario en tiempo real, mediante el uso de hardware reconfigurable.

Como consecuencia del desarrollo de este proyecto se ha llegado a la conclusión de que el uso de hardware reconfigurable ha sido un acierto en cuanto a mejora de rendimiento. Ya que gracias a su uso se ha mejorado 654,9 veces el tiempo de ejecución en software de la aplicación.

Para la realización de este TFG ha sido necesario integrar los conocimientos recibidos de las siguientes asignaturas de la carrera: Diseño de Sistemas Basados en Microprocesador, Sistemas Inteligentes, Ingeniería del Software, Redes, Sistemas Operativos, Gestión y Administración de Redes entre otras.

7.1 Objetivos logrados

Se han logrado cumplir todos los objetivos específicos relacionados con Detección de obstáculos; Distinción y cálculo de posición del vehículo; Cálculo de trayectoria; Generación de movimientos mediante datos de la trayectoria; Comunicación con el vehículo; Movimiento del vehículo; Funcionamiento del sistema en tiempo real.

7.2 Trabajo futuro

Ahora bien, gracias al trabajo realizado en este proyecto se ha podido pensar en varias optimizaciones que se pueden realizar en el sistema para mejorar su funcionamiento. Estas son algunas de ellas:

1. Mejorar la detección de obstáculos. La resta de imágenes funciona bien para la mayoría de casos. Pero el umbral con el cual se elimina el ruido que se ha generado en la toma de las fotografías A y B no es dinámico. Es decir, es necesario realizar una configuración para cada escenario en el cual se instala el sistema. Este hecho limita mucho la compatibilidad entre configuraciones de distintos escenario. Se propone añadir un algoritmo que calcule la configuración en función de la altura y la luminosidad ya que

si para el mismo escenario se sitúa la cámara a una altura distinta el umbral que se había configurado para el escenario probablemente no de unos buenos resultados para todos los casos. Al igual que el hecho de ejecutar la aplicación en días distintos en los que el tiempo cambie puede dar lugar a que el umbral no funcione como se espera debido al cambio de luz del entorno.

2. Mejorar la aplicación del algoritmo de segmentación de la imagen en macro-bloques. El algoritmo funciona correctamente en cuanto a calcular el número de objetos que aparecen en un bloque. El problema es que se ha simplificado el algoritmo debido a que se pueden dar situaciones que den lugar a una colisión. Es decir, el pre-procesamiento que se realiza del mapa en resolución *FullHD* simplifica el resultado de tal manera que si el número de objetos por bloque es mayor que uno, pone ese bloque como ocupado. Aunque el bloque esté mayormente vacío. Como consecuencia la ruta que calcula posteriormente el algoritmo A* no es óptima en cuanto a distancia aunque la naturaleza del algoritmo sea calcular la ruta más corta posible. Esto es debido a que el hecho de simplificar los 900 valores posibles (30×30) a 2 (Ocupado o libre) evita que se puedan visitar nodos que son una mejor opción en cuanto a distancia más corta. Por ello, como mejora se invita a analizar si los bloques que no están ocupados completamente son transitables. Ya que se puede dar la circunstancia en la que una pequeña línea atraviese el bloque de tal forma que no se pueda pasar por el, aunque el bloque esté prácticamente libre de objetos. Si se consigue analizar si un bloque es transitible se puede dejar el valor calculado previamente por el algoritmo y así se mejorará la posterior aplicación del algoritmo A* que calculará una ruta más corta para el mismo escenario.
3. Implementar todo el procesamiento de imágenes en la FPGA de la placa *Zedboard*. En este TFG solo se ha implementado en FPGA el algoritmo de cálculo de trayectoria. Si se consiguiera utilizar hardware reconfigurable para todos los algoritmos del proyecto se podría mejorar el tiempo de ejecución del sistema, permitiendo así poder aumentar la velocidad de movimiento del vehículo a causa de su mejor capacidad de reacción frente a cambios en el escenario.

ANEXOS

Anexo A

Componentes que forman el vehículo

Como brazo ejecutor del sistema se ha decidido integrar una serie de componentes con los cuales se pueda llevar a cabo la conducción, comunicación, y traducción de la información en movimientos. A continuación se va a proceder a explicar en cada uno de los apartados el porqué se han elegido dichos componentes.

A.1 *Arduino UNO R3 ATMEGA328*

Ejerce de plataforma computacional con la que se controlan todos los elementos de prototipado que, en su conjunto, forman el vehículo. Lo bueno del uso de esta placa es que nos permite integrar el uso de sensores, motores y módulos de comunicación. Lo cual es perfecto ya que nos permite diseñar un vehículo en función de nuestras necesidades.

A.2 Controlador *Sparkfun Ardumoto* y chasis

El laboratorio *ARCO* concedió para este proyecto un chasis en el cual vienen instalados cuatro motores de corriente continua de 9V conectados a un controlador Ardumoto de la marca *Sparkfun*. Este artilugio permite obtener resultados muy rápido ya que mediante el uso de estos cuatro pines digitales se puede emular el movimiento de un coche:

- DIRA 12 y DIRB 13: Con el valor de salida 0 las ruedas giran en sentido de las agujas del reloj y con la salida a 1 en sentido antihorario.
- PMWA 3 y PMWA 11: Permiten configurar la velocidad de rotación de los motores. El rango de velocidad se encuentra entre 0 (parado) y 255 (máxima velocidad).

A.3 Módulos *Xbee* de la marca *Digi*

Mediante el uso de estos módulos de comunicación por radio se puede establecer una comunicación por puerto serie entre *Arduino* y la placa *Zedboard*. Igual que con el controlador ardumoto, la aplicación de este módulo nos proporciona rápidos resultados. Esto es debido a que gracias al manejo de un adaptador microusb de la marca *Sparkfun* y el software *XCTU* de la marca *Digi* se realiza una configuración entre ambos módulos que permite establecer la comunicación entre ambos y simplifica la interacción con la placa *Arduino* al uso de 2 pines, obviando los pines de alimentación:

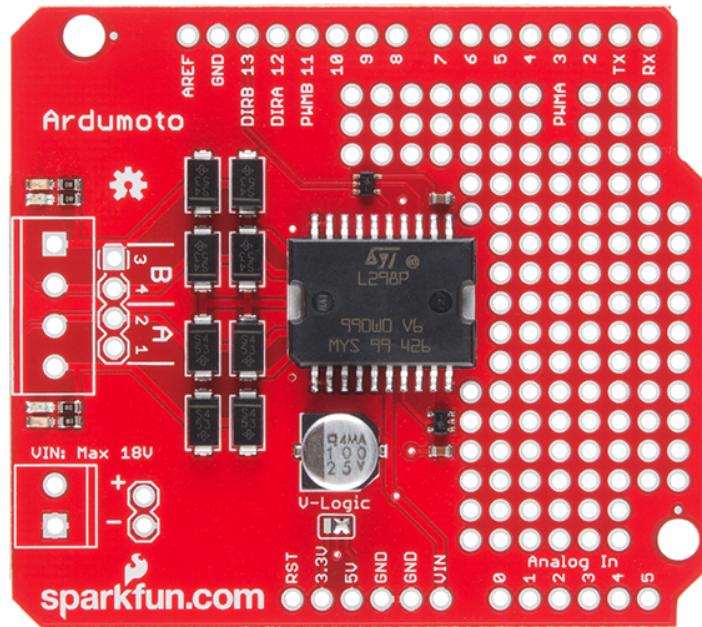
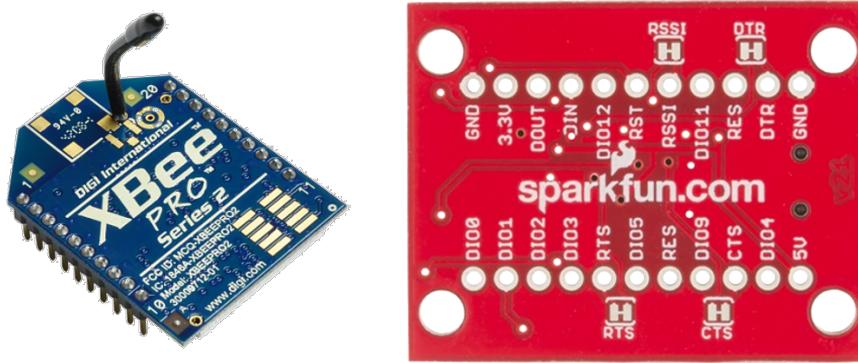


Figura A.1: ArduMoto

- DIN y DOUT mediante los cuales se establece la comunicación de entrada y salida por el puerto serie.



(a) Módulo Xbee

(b) Adaptador microusb

Figura A.2: Módulo de comunicación del sistema

A.4 Batería recargable *Odec* 9V

Se ha decidido utilizar pilas recargables *Odec* 9V de 600mAh de salida debido a que por el número de componentes que forman el vehículo, el consumo del coche es grande. En un principio se utilizaron pilas de 9V no recargables y se consumían en un día de pruebas, lo cual generaba un coste innecesario debido a que se debían comprar pilas muy frecuentemente.

A.5 Osciloscopio y multímetro

El laboratorio ARCO posee un multímetro BEHA 93423 con el cual se han realizado medidas de voltaje sobre el prototipado del proyecto. Este artilugio ha sido de gran utilidad ya que

en un momento del proyecto se cometió un fallo de alimentación en uno de los componentes. El descuido consistió en que debido a la utilización de una base *Shield* de la marca *Seed*. Véase la figura A.3. Todos los pines VCC de la placa estaban configurados en 5V pero el módulo *XBee* estaba utilizando el pin de 3.3V. A causa de esta negligencia el módulo *XBee* dejó de funcionar ya que se alimentaba de un voltaje mayor del que soportaba. Hubo un momento en el cual no se podía ver a simple vista que ocurría en el coche. Ya que la comunicación no funcionaba. Y en un principio no se sabía que componente era el causante del mal funcionamiento. Debido a que no se producía ningún movimiento se pensó que era debido al módulo de comunicación, pero el led del adaptador microusb para *XBee* funcionaba. En primer lugar se comprobó si era por algún error en la configuración, pero tras comprobar que la configuración era correcta. Se procedió a medir los voltajes de los componentes. Fue en ese momento cuando se comprobó que el voltaje de entrada del módulo *XBEE* era erróneo. El módulo *XBee* fue sustituido por uno nuevo y se tomó una mayor precaución en el futuro para evitar este tipo de descuidos.

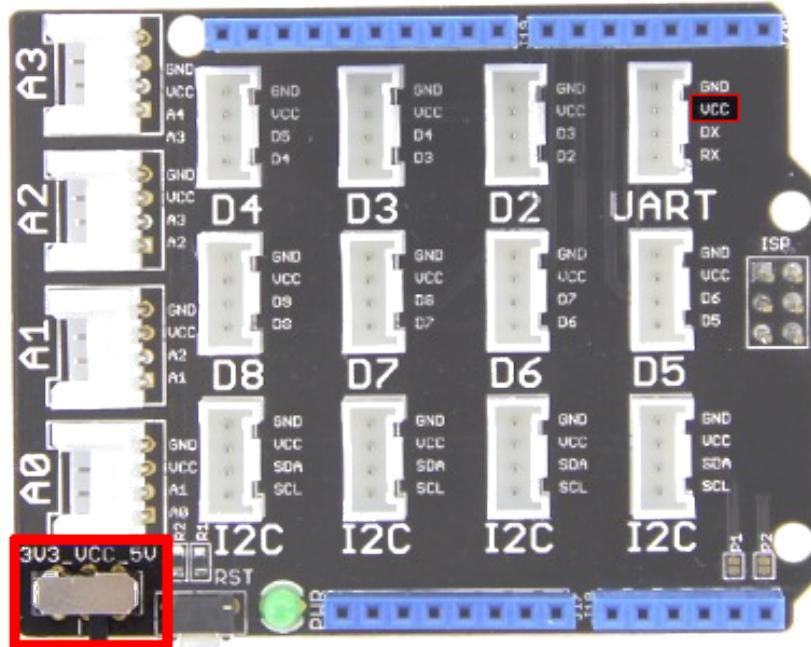


Figura A.3: Placa base

A.6 Placa de pruebas para Arduino

Sobre esta tabla se establece la instalación de los sensores, leds y resistencias del coche. Esta placa de pruebas se ha utilizado como maqueta para realizar una serie de pruebas correspondientes a activar los motores, detectar el efecto Hall, encender leds y comprobar el funcionamiento del protocolo de comunicación establecido entre la placa *Zedboard* y *Arduino*. Además, se utiliza en la versión final del vehículo como lugar de instalación de los sensores de efecto Hall, leds y resistencias que permiten el funcionamiento de los sensores y leds.

A.7 Arduino shield

Imprescindible para poder utilizar los pines de *Arduino* que no utiliza el controlador ArduMoto. Ya que si instalaramos la placa arduMoto sobre *Arduino* inutilizaría los pines de *Arduino* que no se utilizan. Véase la figura A.4. Mientras que con el uso de la base *Shield*, véase la figura A.3, se pueden utilizar todos los pines que ocupa *ArduMoto*.

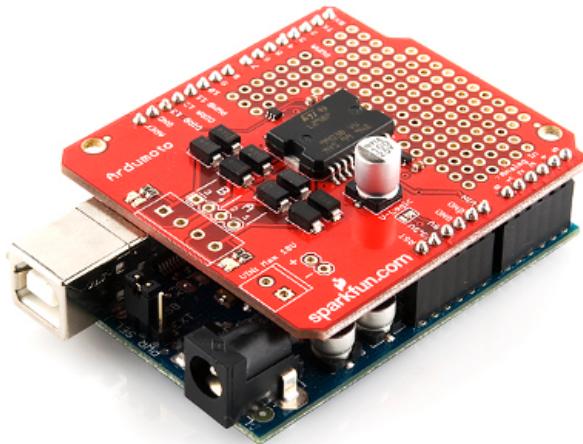


Figura A.4: Ardumoto instalado sobre *Arduino UNO*

A.8 Sensor efecto Hall A3144

Debido a que los motores de corriente continua carecen de encoders¹. Se ha decidido emular el funcionamiento de un encoder mediante el uso de un sensor de efecto Hall². El sensor de efecto Hall permite detectar la variación producida en un campo eléctrico mediante la aproximación de un campo magnético. Si fijamos un imán de neodimio en el lateral de la rueda e instalamos un sensor pegado a esta. Cada vez que el imán pase cerca del sensor A3144, este detectará la variación del campo eléctrico. Si el imán se acerca al sensor devuelve el valor 0 por el pin digital, mientras que si por el contrario el imán se encuentra lejos, devuelve un 1.

El uso de este sensor nos permite controlar el número de vueltas que realiza una rueda. Ya que si contamos el número de veces que el sensor devuelve un 0 por el pin digital que está conectado a *Arduino*, podremos mantener un control acerca de cuánto se mueve el vehículo. Como en los casos de ejemplo la cámara cenital se coloca a un par de metros, se ha decidido situar cuatro imanes en los laterales de las ruedas que poseen la tracción, en este caso las

¹Los Encoders convierten el movimiento en una señal eléctrica que puede ser leída por algún tipo de dispositivo de control en un sistema de control de movimiento, tal como un mostrador. El encoder envía una señal de respuesta que puede ser utilizada para determinar la posición, contar, velocidad o dirección.

²Se conoce como efecto Hall a la aparición de un campo eléctrico por separación de cargas, en el interior de un conductor por el que circula una corriente en presencia de un campo magnético con componente perpendicular al movimiento de las cargas. Este campo eléctrico (campo Hall) es perpendicular al movimiento de las cargas y a la componente perpendicular del campo magnético aplicado. Lleva el nombre de su primer modelador, el físico estadounidense Edwin Herbert Hall.

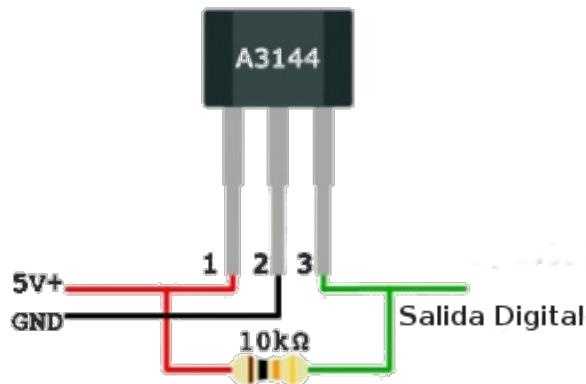


Figura A.5: Sensor Efecto Hall A3144

traseras. De esta forma se puede controlar con mayor precisión la distancia que se quiera recorrer con el vehículo al tener un mayor número de muestras por vuelta completa.

El fundamento teórico de contar el número de vueltas con el sensor A3144 es correcto, pero en la práctica los resultados del uso de este sensor no fueron los esperados. Ya que el sensor consideraba falsas aproximaciones del imán de neodimio. Es decir, el sensor transmitía valores 0 por el pin digital sin que hubiera existido ninguna aproximación del imán. Con el uso del multímetro tampoco se percibía algo extraño en las patas del sensor. El voltaje tenía un valor de 5V, la salida a tierra era correcta, y la resistencia funcionaba correctamente. Se cumplían los requisitos de funcionamiento de la figura A.5. El problema era el ruido³. Este tipo de ruido aparece aleatoriamente por lo que se ha suplido con la introducción de un capacitor 473 y una modificación en software pensada para contemplar la aparición de ruido.

A.9 Estación de soldadura JBM AR5800

A causa de la realización de un diseño propio para el vehículo, la mayoría de componentes que se han introducido en el vehículo se han tenido que soldar con estaño. Por ejemplo, con el uso del adaptador microusb del módulo *XBee* de la marca *Sparkfun* ha sido necesario realizar una soldadura. Ya que por defecto el adaptador venía con un soldadura hecha en el pin de 3.3V pero debido al uso del *Shield* de *Arduino* se necesita un voltaje de entrada de 5V.

A.10 Impresión de marca identificativa para el coche

Con el uso de un trozo de cartón y la impresión de un círculo rojo en dos folios de papel, se ha realizado la marca identificativa del vehículo frente a los demás objetos.

³Por ruido se entiende toda componente de tensión o intensidad indeseada que se superpone con la componente de la señal que se procesa o interfiere con el proceso de medida.

Anexo B

Código de los algoritmos del sistema

B.1 Cabecera del formato BMP

```
/* JoseJimenezRincon */
/* BMPformat.h */

/*BMP_HEADER
2Bytes Type of image format
4Bytes Size of the BMP file in bytes
2Bytes Reserved
2Bytes Reserved
4Bytes The offset, starting address, of the byte where pixel array can be found.

BMP_INFO_HEADER
4Bytes infoSize The number of bytes required by the structure.
4Bytes width The width of the bitmap, in pixels.
4Bytes heighth The height of the bitmap, in pixels. If heighth is positive, the bitmap is a bottom-up
DIB and its origin is the lower-left corner. If it is negative, the bitmap is a top-down DIB and
its origin is the upper-left corner.
2Bytes planes The number of planes for the target device. This value must be set to 1.
2Bytes bitsPerPixel Determines the number of bits that define each pixel and the maximum number os
colors in the bitmap. GREYSCALE
4Bytes compression The type of compression for a compressed bottom-up bitmap (top-down DIBs cannot
be compresses). Value 0.
4Bytes imageSize The size, in bytes, of the image.
4Bytes yPixelParameter The vertical resolution, in pixels-per-meter, of the target device for the
bitmap.
4Bytes xPixelParameter The horizontal resoluion.
4Bytes numColorsPallate The number of color indexes in the color table that are actually used by the
bitmap.
4Bytes mostImpColor The number of color indexes that are required for displaying the bitmap. If this
value is zero, all colors.

RGBQUAD
rgbBlue 0-255
rgbGreen 0-255
rgbRed 0-255
```

```
rgbReserved 0-255*/
```

```
#ifndef _BMPformat_H
#define _BMPformat_H 1
```

```
#pragma pack(1)
```

```
typedef struct {
    unsigned short type;
    unsigned int size;
    unsigned short res1;
    unsigned short res2;
    unsigned int offset;
} BMP_HEADER;
```

```
typedef struct {
    unsigned int infoSize;
    int width;
    int height;
    unsigned short planes;
    unsigned short bitsPerPixel;
    unsigned int compression;
    unsigned int imageSize;
    int yPixelParameter;
    int xPixelParameter;
    unsigned int numColorsPallatte;
    unsigned int mostImpColor;
} BMP_INFO_HEADER;
```

```
typedef struct{
    unsigned char  rgbBlue;
    unsigned char  rgbGreen;
    unsigned char  rgbRed;
    unsigned char  rgbReserved;
} RGBQUAD;
```

```
typedef struct {
    BMP_HEADER bmp_header;
    BMP_INFO_HEADER bmp_info_header;
    RGBQUAD indexedColData[256];
} BITMAP_FILE;
```

```
#endif /* _BMPformat_H */
```

Listado B.1: Cabecera del formato BMP

B.2 Detección de objetos

```
/* Jose Jimenez Rincon */
/* DetectObjects.c */

#include <stdio.h>
#include "SETTINGS.h"

void subtraction ( unsigned char *pixelData_Output, unsigned char *pixelData_Base, unsigned char *
pixelData_Obj, int imageSize){

    int x, subtraction_value;

    for (x=0; x<imageSize; x++){

        subtraction_value = abs(pixelData_Base[x]-pixelData_Obj[x]);

        if ( subtraction_value > THRESHOLD ){

            pixelData_Output[x] = BLACK;

        } else {

            pixelData_Output[x] = WHITE;

        }
    }

} //end subtraction

void square (unsigned char *pixelData_Output, int imageSize, int width, int height){

    int reference_y = HEIGHT_CAR;
    int reference_x = WIDTH_CAR;
    int x,y,z,j;
    int inicializar_x = FALSE;
    int inicializar_y = FALSE;
    int y1 = FALSE;
    int y2 = FALSE;
    int y_last = FALSE;
    int y_now = FALSE;
    int x1 = FALSE;
    int x2 = FALSE;
    int x3 = FALSE;
    int count_objects = 0;
    int objects_now= 0;
    int objects_last = 0;
    int accountant_x = 0;
```

```

int accountant_y = 0;
int accountant_buffer_y = 0;
int accountant_buffer_x = 0;
int objects_coordinates_y[height/HEIGHT_CAR];
int objects_coordinates_x[height/HEIGHT_CAR];
int buffer_y[height/HEIGHT_CAR];
int buffer_x[height/HEIGHT_CAR];

for (x=0; x<width; x++){ //Recorre el eje X hasta el ancho de la imagen

    if ( count_objects > 0 ){ // Si el numero de objetos es mayor que 0
        accountant_buffer_y = 0;

        while ( count_objects > 0 ){ //Mientras el numero de objetos sea mayor que 0

            if ( x1 == TRUE ) { //Si el primer valor de la izq se ha encontrado.

                if ( objects_last != objects_now ){ //Si el numero de
                    objetos de la pasada anterior es distinto del numero de
                    objetos de la pasada actual.

                if ( objects_last < objects_now ) { //Si el numero
                    de objetos de la pasada anterior es menor que la
                    actual implica que hay nuevo x1, y1 e y2.

                if ( x3 == FALSE ){ //Si no hay dos o mas
                    objetos en la misma columna hasta el
                    momento.

                    if ( abs(buffer_y[
                        accountant_buffer_y+(objects_now
                        *2-2)] - y_last) / width >=
                        HEIGHT_CAR && count_objects > 1
                        || x -
                        objects_coordinates_x
                        [accountant_x-1]
                        >= WIDTH_CAR &&
                        count_objects >
                        1 ) {
                            /* Si la distancia entre el
                            valor y2 del objeto de
                            abajo y el valor y1 del
                            objeto de arriba es
                            mayor que la referencia,
                            significa que es un nuevo objeto.*/
                            objects_coordinates_y[
                                accountant_y++] =

```

ANEXO B. CÓDIGO DE LOS ALGORITMOS DEL SISTEMA

```
        buffer_y[
            accountant_buffer_y++];
        objects_coordinates_y[
            accountant_y++] =
            buffer_y[
                accountant_buffer_y++];
        objects_coordinates_x[
            accountant_x++] = x - 1;
        objects_coordinates_x[
            accountant_x++] = x - 1;
        x3 = TRUE;
    }else{
        objects_coordinates_x[
            accountant_x-1] = x2;
        objects_coordinates_y[
            accountant_y-1] =
            buffer_y[
                accountant_buffer_y];
    }
}else{ //x3 == TRUE

    if ( objects_coordinates_y[
        accountant_y-2] < buffer_y[
        count_objects] ){
        objects_coordinates_y[
            accountant_y-2] =
            buffer_y[count_objects];
    }

    if ( objects_coordinates_y[
        accountant_y-1] > buffer_y[
        count_objects-1] ){
        objects_coordinates_y[
            accountant_y-1] =
            buffer_y[count_objects
            -1];
    }

}

}else{ //objects_last > object_now

    y_now = buffer_y[accountant_buffer_y];
    if ( x3 == TRUE ){
        objects_coordinates_x[accountant_x -
        3] = x2;
    }else{


```

```

        objects_coordinates_x[accountant_x -
                               1] = x2;
    }

    x3 = FALSE;
}

}

}else{ // objects_last == objects_now
    if ( x3 == FALSE ) {
        if ( objects_coordinates_y[accountant_y-2] <
            buffer_y[count_objects] ){
            objects_coordinates_y[accountant_y
                               -2] = buffer_y[count_objects];
        }

        if ( objects_coordinates_y[accountant_y-1] >
            buffer_y[count_objects-1] ){
            objects_coordinates_y[accountant_y
                               -1] = buffer_y[count_objects-1];
        }
    }
    x2 = x;
}

}else{ // x1 == FALSE
    objects_coordinates_y[accountant_y++] = buffer_y[
        accountant_buffer_y++];
    objects_coordinates_y[accountant_y++] = buffer_y[
        accountant_buffer_y++];
    if ( x - objects_coordinates_x[accountant_x-1] >= WIDTH_CAR
        ){
        objects_coordinates_x[accountant_x++] = x - 1;
        objects_coordinates_x[accountant_x++] = x - 1;
    }
    count_objects--;
}

x1 = TRUE;

}else{ // Si no hay objetos en la columna.

    if ( reference_x == 0 ) {
        objects_coordinates_x[accountant_x - 1] = x2;
        x1 = FALSE;
    }
}

```

ANEXO B. CÓDIGO DE LOS ALGORITMOS DEL SISTEMA

```
    if ( objects_last > objects_now && x - objects_coordinates_x[accountant_x-1]
        >= WIDTH_CAR && objects_now == 0){
        objects_coordinates_x[accountant_x - 1] = x2;
        x1 = FALSE;
    }
    reference_x--;
}

y_last = y_now;
objects_last = objects_now; //Guardo la variable de la etapa anterior, antes de
actualizarla de nuevo en la fase de captura.

for (y=x; y<imageSize ;y=y+width){

    if ( pixelData_Output[y] == BLACK ){ //Si el pixel es negro.

        reference_x = WIDTH_CAR;
        if ( y1 == FALSE ){ // Si no se ha encontrado ningun objeto hasta el
            momento.
            buffer_y[count_objects++] = y;
            pixelData_Output[y] = BLACK;
            y1 = TRUE;
        }

    }else{ //y1 == TRUE
        pixelData_Output[y] = WHITE;
        y2 = y;
    }

    reference_y = HEIGHT_CAR;

}else{ //pixelData_y == WHITE

    if ( y1 == TRUE ){

        if (reference_y == 0 || y + width >= imageSize){ //Si la
            referencia es igual a 0 o se encuentra en el limite de
            la coordenada Y.
            pixelData_Output[y2] = BLACK;
            buffer_y[count_objects++] = y2;
            reference_y = HEIGHT_CAR;
            y1 = FALSE;
        }
        reference_y--;
    }

}

}
```

B.3. IMPLEMENTACIÓN A* PARA LA HERRAMIENTA VIVADO HLS

```
objects_now = count_objects / 2; //Guardo la variable para que no se destruya en la
                                //fase de analisis.
count_objects = objects_now;
y1 = FALSE;
y2 = FALSE;
}

for (z = 0; z<accountant_x; z=z+2){
    for ( x = objects_coordinates_x[z]; x < objects_coordinates_x[z+1]; x++){
        for ( y = (x+((objects_coordinates_y[z+1]/width)*width)); y < (
            objects_coordinates_y[z]/width*width)+(objects_coordinates_x[z+1]); y=y+
            width){
            pixelData_Output[y] = BLACK;
        }
    }
}

} // end square
```

Listado B.2: Detección de objetos

B.3 Implementación A* para la herramienta *Vivado HLS*

```
/*
// * aStarStatic.cpp
// *
// * Created on: 12 jun. 2017
// *      Author: jose
// */

#include "aStarStatic.h"
#include "../SETTINGS.h"
#include <fstream>

void initNodes(int horizontal, int vertical, int map[MACROWIDTH][MACROHEIGHT]){
    int size = horizontal*vertical;
    int x=0;
    int y=0;

    for (int i = 0; i < size; i++) {
        openNodes[i]=1;//Initialize openNodes

        x=i%horizontal;
        if (x==0 && i!=0) y++;

        if(map[x][y]!=1){
```

ANEXO B. CÓDIGO DE LOS ALGORITMOS DEL SISTEMA

```
        node[i].coordinates[0]=x;
        node[i].coordinates[1]=y;
        node[i].visited=0;
    }else{
        node[i].visited=1;
    }
}

void calculateFather(int position, int xFather, int yFather){
    int positionFather = xFather+(yFather*MACROWIDTH);
    node[position].father= positionFather;
}

void openNode(int position){
    openNodes[position]=0;
}

void closeNode(int position){
    openNodes[position]=1;
    node[position].visited=1;
}

void adjacentNodes(int x, int y){
    int initPosition = x+(y*MACROWIDTH);
    //Left
    adjacentPosition[0][2] = initPosition-1; //Position
    //Right
    adjacentPosition[1][2] = initPosition+1; //Position
    //Top
    adjacentPosition[2][2]= initPosition-MACROWIDTH; //Position
    //Bot
    adjacentPosition[3][2] = initPosition+MACROWIDTH; //Position
    //Upper Left Diagonal
    adjacentPosition[4][2]= initPosition-MACROWIDTH-1; //Position
    //Upper Right Diagonal
    adjacentPosition[5][2] = initPosition-MACROWIDTH+1; //Position
    //Bottom Left Diagonal
    adjacentPosition[6][2] = initPosition+MACROWIDTH-1; //Position
    //Bottom Right Diagonal
    adjacentPosition[7][2] = initPosition+MACROWIDTH+1; //Position

    //if nodes are not adjacent set -1 on position.
    if(x==0){
        //Left
        adjacentPosition[0][2] = -1; //NotAdjacent
        //Right
    }
}
```

B.3. IMPLEMENTACIÓN A* PARA LA HERRAMIENTA VIVADO HLS

```
adjacentPosition[1][0] = x+1; //X
//Top
adjacentPosition[2][0] = x; //X
//Bot
adjacentPosition[3][0] = x; //X
//Upper Left Diagonal
adjacentPosition[4][2]= -1; //Not adjacent
//Upper Right Diagonal
adjacentPosition[5][0] = x+1; //X
//Bottom Left Diagonal
adjacentPosition[6][2] = -1; //Not adjacent
//Bottom Right Diagonal
adjacentPosition[7][0] = x+1; //X
}else if(x==MACROWIDTH-1){
    //Left
    adjacentPosition[0][0] = x-1; //X
    //Right
    adjacentPosition[1][2] = -1; //NotAdjacent
    //Top
    adjacentPosition[2][0] = x; //X
    //Bot
    adjacentPosition[3][0] = x; //X
    //Upper Left Diagonal
    adjacentPosition[4][0]= x-1; //X
    //Upper Right Diagonal
    adjacentPosition[5][2] = -1; //Not Adjacent
    //Bottom Left Diagonal
    adjacentPosition[6][0] = x-1; //X
    //Bottom Right Diagonal
    adjacentPosition[7][2] = -1; //NotAdjacent
}else{
    //Left
    adjacentPosition[0][0] = x-1; //X
    //Right
    adjacentPosition[1][0] = x+1; //X
    //Top
    adjacentPosition[2][0] = x; //X
    //Bot
    adjacentPosition[3][0] = x; //X
    //Upper Left Diagonal
    adjacentPosition[4][0] = x-1; //X
    //Upper Right Diagonal
    adjacentPosition[5][0] = x+1; //X
    //Bottom Left Diagonal
    adjacentPosition[6][0] = x-1; //X
    //Bottom Right Diagonal
    adjacentPosition[7][0] = x+1; //X
```

ANEXO B. CÓDIGO DE LOS ALGORITMOS DEL SISTEMA

```
}

if(y==0){

    //Left
    adjacentPosition[0][1] = y; //Y
    //Right
    adjacentPosition[1][1] = y; //Y
    //Top
    adjacentPosition[2][2]= -1; //NotAdjacent
    //Bot
    adjacentPosition[3][1] = y+1; //Y
    //Upper Left Diagonal
    adjacentPosition[4][2]= -1; //NotAdjacent
    //Upper Right Diagonal
    adjacentPosition[5][2] = -1; //NotAdjacent
    //Bottom Left Diagonal
    adjacentPosition[6][1] = y+1; //Y
    //Bottom Right Diagonal
    adjacentPosition[7][1] = y+1; //Y

else if(y==MACROHEIGHT-1 ){

    //Left
    adjacentPosition[0][1] = y; //Y
    //Right
    adjacentPosition[1][1] = y; //Y
    //Top
    adjacentPosition[2][1]= y-1; //Y
    //Bot
    adjacentPosition[3][2] = -1; //NotAdjacent
    //Upper Left Diagonal
    adjacentPosition[4][1]= y-1; //Y
    //Upper Right Diagonal
    adjacentPosition[5][1] = y-1; //Y
    //Bottom Left Diagonal
    adjacentPosition[6][2] = -1; //NotAdjacent
    //Bottom Right Diagonal
    adjacentPosition[7][2] = -1; //NotAdjacent

else

    //Left
    adjacentPosition[0][1] = y; //Y
    //Right
    adjacentPosition[1][1] = y; //Y
    //Top
    adjacentPosition[2][1] = y-1; //Y
    //Bot
    adjacentPosition[3][1] = y+1; //Y
    //Upper Left Diagonal
    adjacentPosition[4][1] = y-1; //Y
```

B.3. IMPLEMENTACIÓN A* PARA LA HERRAMIENTA VIVADO HLS

```
//Upper Right Diagonal
adjacentPosition[5][1] = y-1; //Y
//Bottom Left Diagonal
adjacentPosition[6][1] = y+1; //Y
//Bottom Right Diagonal
adjacentPosition[7][1] = y+1; //Y
}

}

int calculateBestOption(){
    int bestOption = 2147483647;
    int positionBestOption;

    for (int i = 0; i < (MACROWIDTH*MACROHEIGHT); i++) {
        if(openNodes[i]==0){ //If it is an open node
            if( node[i].costPlusHeuristic < bestOption ){ //If is a better option...
                bestOption = node[i].costPlusHeuristic;
                positionBestOption = i;
            }
        }
    }

    return positionBestOption;
}

void manhattanHeuristic(int x, int y, int position, int xFinish, int yFinish){
    int xDistance, yDistance;
    xDistance = xFinish - x;
    yDistance= yFinish - y;

    node[position].heuristic = abs(xDistance)+abs(yDistance);
}

void calculateCost(int i, int position){
    if(i<4){
        node[position].cost = node[node[position].father].cost + 10; //Orthogonal
    }else{
        node[position].cost += node[node[position].father].cost + 14; //Diagonal
    }
}

void costPlusHeuristic(int position){
    node[position].costPlusHeuristic = node[position].cost + (node[position].heuristic)*10;
}

int aStar(int xStart, int yStart, int xFinish, int yFinish, int map[MACROWIDTH][MACROHEIGHT]){

```

ANEXO B. CÓDIGO DE LOS ALGORITMOS DEL SISTEMA

```
//Initializes coordinates and sets to closed if the node is an obstacle.  
initNodes(MACROWIDTH,MACROHEIGHT, map);  
  
//Starts the initial node search  
int position = xStart+(yStart*MACROWIDTH);  
node[position].father=-1;  
node[position].cost=0;  
node[position].heuristic=0;  
node[position].costPlusHeuristic=0;  
node[position].visited=1;  
  
//Compute adjacent nodes of the initial node  
adjacentNodes(xStart, yStart);  
  
//Insert Node information like... FatherPosition, cost and heuristic.  
for (int i = 0; i < 8; i++) {  
    if(node[adjacentPosition[i][2]].visited==0 && adjacentPosition[i][2]!=-1 ){  
        openNode(adjacentPosition[i][2]); //Add node to list of open nodes.  
        calculateFather(adjacentPosition[i][2], xStart, yStart);//Set position of  
        father to adjacent node  
        manhattanHeuristic(adjacentPosition[i][0], adjacentPosition[i][1],  
        adjacentPosition[i][2], xFinish, yFinish);  
        calculateCost(i, adjacentPosition[i][2]);//Set cost of movement to adjacent  
        node  
        costPlusHeuristic(adjacentPosition[i][2]);//Cost plus heuristic  
    }  
}  
  
//Calculate position of best option from adjacent nodes.  
position = calculateBestOption();  
//Close node of best position.  
closeNode(position);  
  
//Calculate final position  
int finalPosition = xFinish+(yFinish*MACROWIDTH);  
  
//Recursive pathfinding  
do{  
    adjacentNodes(node[position].coordinates[0], node[position].coordinates[1]);  
    for (int i = 0; i < 8; i++) {  
        if(node[adjacentPosition[i][2]].visited==0 && adjacentPosition[i][2]!=-1){  
            openNode(adjacentPosition[i][2]); //Add node to list of open nodes.  
            calculateFather(adjacentPosition[i][2], node[position].coordinates  
            [0], node[position].coordinates[1]);//Set position of father to  
            adjacent node
```

B.4. DETECCIÓN DEL VEHÍCULO FRENTE A LOS OBJETOS

```
        manhattanHeuristic(adjacentPosition[i][0], adjacentPosition[i][1],
                            adjacentPosition[i][2], xFinish, yFinish);
        calculateCost(i, adjacentPosition[i][2]); //Set cost of movement to
                                         adjacent node
        costPlusHeuristic(adjacentPosition[i][2]); //Cost plus heuristic
    }else{
        //If node is closed or if position is outside map, break. if(node[
        adjacentPosition[i][2]).closed == 1)
    }
}

//Calculate position of best option from adjacent nodes.
position = calculateBestOption();

//Close node of best position.
closeNode(position);

while(position!=finalPosition);

int route[MACROWIDTH*MACROHEIGHT][2];
route[0][0]=xFinish;
route[0][1]=yFinish;
int counter=1;
while(node[position].father!=-1){
    map[node[position].coordinates[0]][node[position].coordinates[1]]=2;
    route[counter][0]=node[position].coordinates[0];
    route[counter][1]=node[position].coordinates[1];
    position=node[position].father;
    counter++;
}
route[counter][0]=xStart;
route[counter][1]=yStart;

map[xFinish][yFinish]=4;

std::ofstream output("trajectory.txt");
while(counter>0){
    output<<"X: "<<route[counter][0]<<" Y: "<<route[counter][1]<<std::endl;
    counter--;
}
output.close();

return 0;
}
```

Listado B.3: Implementación A* para la herramienta *Vivado HLS*

B.4 Detección del vehículo frente a los objetos

/*

ANEXO B. CÓDIGO DE LOS ALGORITMOS DEL SISTEMA

```
* deteccionVehiculo.cpp
*
*   Created on: 19/12/2016
*       Author: jose
*/
#include <opencv2/opencv.hpp>
#include <iostream>
#include <string>
#include <vector>
#include <fstream>
#include "../SETTINGS.h"

using namespace cv;
using namespace std;

int vehicleDetection(const char* path) {

    // Cargamos la imagen de entrada
    Mat bgr_image = imread(path);

    // Clonamos la imagen
    Mat orig_image = bgr_image.clone();

    // Convert input image to HSV
    Mat hsv_image;
    cvtColor(bgr_image, hsv_image, COLOR_BGR2HSV);

    // Threshold the HSV image, keep only the red pixels
    Mat red_vehicule;
    inRange(hsv_image, Scalar(170, 100, 100), Scalar(179, 255, 255), red_vehicule);

    //Gaussian filter to detect better circles
    GaussianBlur(red_vehicule, red_vehicule, cv::Size(9, 9), 2, 2);

    // Use the Hough transform to detect circles in the combined threshold image
    vector<Vec3f> circles;
    HoughCircles(red_vehicule, circles, CV_HOUGH_GRADIENT, 1, red_vehicule.rows/8, 100, 20, 0,
                 0);

    std::ofstream output("vehicle.txt");

    int blockWidth=MAPWIDTH/MACROWIDTH;
    int blockHeight=MAPHEIGHT/MACROHEIGHT;

    // Loop over all detected circles and outline them on the original image
    if(circles.size() == 0) std::exit(-1);
```

```

for(size_t current_circle = 0; current_circle < circles.size(); ++current_circle) {
    Point center(round(circles[current_circle][0]), round(circles[current_circle][1]));
    int radius = round(circles[current_circle][2]);
    circle(orig_image, center, radius, Scalar(0, 255, 0), 5);

    center.x=center.x/blockWidth; //1920
    center.y=center.y/blockHeight; //1080
    output<<center;
}

output.close();

//namedWindow("Imagen", CV_WINDOW_AUTOSIZE);
namedWindow("Imagen", CV_WINDOW_NORMAL);
imshow("Imagen", red_vehicule);
namedWindow("Detected red circles on the input image", CV_WINDOW_NORMAL);
imshow("Detected red circles on the input image", orig_image);

waitKey(0);

return 0;
}

```

Listado B.4: Detección del vehículo frente a los objetos

B.5 Comunicación puerto serie

```

/*
 * communication.cpp
 *
 * Created on: 24 may. 2017
 * Author: jose
 */

#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#include <fcntl.h> /* File Control Definitions */
#include <termios.h>/* POSIX Terminal Control Definitions*/
#include <unistd.h> /* UNIX Standard Definitions */
#include <errno.h> /* ERROR Number Definitions */

int communication(unsigned char byte1,unsigned char byte2, unsigned char byte3){

    int fd; /*File Descriptor*/
    /*----- Opening the Serial Port -----*/

```

ANEXO B. CÓDIGO DE LOS ALGORITMOS DEL SISTEMA

```
/* Change /dev/ttyUSB0 to the one corresponding to your system */

fd = open("/dev/ttyUSB0", O_RDWR, 0_NDELAY);      /* ttyUSB0 is the FT232 based USB2SERIAL
   Converter */
/* O_RDWR Read/Write access to serial port          */
/* O_NOCTTY - No terminal will control the process */
/* O_NDELAY -Non Blocking Mode,Does not care about- */
/* -the status of DCD line,Open() returns immediatly */

if(fd == -1)                                     /* Error Checking */
    printf("\n  Error! in Opening ttyUSB0  ");
else
    printf("\n  ttyUSB0 Opened Successfully ";

/*----- Setting the Attributes of the serial port using termios structure ----- */

struct termios SerialPortSettings;      /* Create the structure */

tcgetattr(fd, &SerialPortSettings);      /* Get the current attributes of the Serial port */

/* Setting the Baud rate */
cfsetispeed(&SerialPortSettings,B9600); /* Set Read  Speed as 9600           */
cfsetospeed(&SerialPortSettings,B9600); /* Set Write Speed as 9600           */

/* 8N1 Mode */
SerialPortSettings.c_cflag &= ~PARENB; /* Disables the Parity Enable bit(PARENB),So No
   Parity */
SerialPortSettings.c_cflag &= ~CSTOPB; /* CSTOPB = 2 Stop bits,here it is cleared so 1
   Stop bit */
SerialPortSettings.c_cflag &= ~CSIZE; /* Clears the mask for setting the data size
   */
SerialPortSettings.c_cflag |= CS8; /* Set the data bits = 8
   */

SerialPortSettings.c_cflag &= ~CRTSCTS; /* No Hardware flow Control
   */
SerialPortSettings.c_cflag |= CREAD | CLOCAL; /* Enable receiver,Ignore Modem Control lines
   */

SerialPortSettings.c_iflag &= ~(IXON | IXOFF | IXANY); /* Disable XON/XOFF flow
   control both i/p and o/p */
SerialPortSettings.c_iflag &= ~ICANON; /* Non Canonical mode */

SerialPortSettings.c_oflag &= OPOST; /*No Output Processing*/
```

```

/* Setting Time outs */
SerialPortSettings.c_cc[VMIN] = 0; /* Read at least 10 characters */
SerialPortSettings.c_cc[VTIME] = 0; /* Wait indefinitely */

//tcflush(fd, TCIFLUSH);

if((tcsetattr(fd,TCSANOW,&SerialPortSettings)) != 0) /* Set the attributes to the termios
structure*/
    printf("\n  ERROR ! in Setting attributes");
else
    printf("\n  BaudRate = 9600 \n  StopBits = 1 \n  Parity   = none");

/*----- Communication -----*/
unsigned char packet[3];
packet[0] = byte1;
packet[1] = byte2;
packet[2] = byte3;

printf("\n\n +-----+Packet Sent-----+ \n\tID: %d Orden: %d Vueltas: %d\n
+-----+\n", byte1, ( byte2 & 0xE0 ) >> 5, byte3);

int bytes_written = 0;
bytes_written = write(fd,packet,sizeof(packet));

unsigned char read_char;

if ( read(fd,&read_char,1) <= 0){
    printf(" +-----+\n\tERROR. Movement Timeout\n
+-----+\n\n");
    return 1;
}else{
    if ( read_char == 85 ){
        printf(" +-----+\n\tMovement Done.\n
+-----+\n\n");
        return 0;
    }

    if ( read_char == 170 ){
        printf(" +-----+\n\tERROR. Information Outdated
.\n +-----+\n\n");
        return 1;
    }
}

close(fd); /* Close the serial port */
}

```

```
//struct termios
//{
//tcflag_t c_iflag; /* input mode flags */
//tcflag_t c_oflag; /* output mode flags */
//tcflag_t c_cflag; /* control mode flags */
//tcflag_t c_lflag; /* local mode flags */
//cc_t c_line;      /* line discipline */
//cc_t c_cc[NCCS]; /* control characters */
//};
```

Listado B.5: Comunicación puerto serie

B.6 Coordinador de los movimientos del vehículo

```
/*
 * coordinatorTrajectory.cpp
 *
 * Created on: 24 may. 2017
 * Author: jose
 */

#include <stdio.h>
#include "communication.h"

#define true 1
#define false 0

#define LIMITE_VUELTAS

unsigned char ARRIBA = 0;
unsigned char DIAGONAL_INFERIOR_DERECHA = 1 << 5;
unsigned char DETRAS = 2 << 5;
unsigned char DIAGONAL_INFERIOR_IZQUIERDA = 3 << 5;
unsigned char ABAJO = 4 << 5;
unsigned char DIAGONAL_SUPERIOR_IZQUIERDA = 5 << 5;
unsigned char DELANTE = 6 << 5;
unsigned char DIAGONAL_SUPERIOR_DERECHA = 7 << 5;

int coordinatorTrajectory(){

    FILE *file = fopen("trajectory.txt","r");
    char line[16];
    int coorX, coorY;
    int coorXold, coorYold;
    int coorXchange, coorYchange;
```

B.6. COORDINADOR DE LOS MOVIMIENTOS DEL VEHÍCULO

```
int direccion;
unsigned char identifier = 0;
unsigned char contadorMovDiagonales = 0;
unsigned char contadorMovClasicos = 0;
unsigned char permisoMovimiento = false;

fgets(line, sizeof(line), file);
sscanf(line, "X: %d Y: %d", &coorXold, &coorYold);

while (fgets(line, sizeof(line), file)) {

    sscanf(line, "X: %d Y: %d", &coorX, &coorY);

    if( coorXold != coorX && coorYold != coorY ){ //DIAGONALES
        contadorMovDiagonales++;
    }else{
        contadorMovClasicos++;
    }

    if ( (contadorMovClasicos != 0 && contadorMovDiagonales != 0) || contadorMovClasicos
        == LIMITE_VUELTAS-1 || contadorMovDiagonales == LIMITE_VUELTAS-1){
        permisoMovimiento = true;
        printf("\n\nXold= %d, Yold= %d\tX= %d, Y= %d\n", coorXold, coorYold, coorX,
               coorY);
    }
}

if ( permisoMovimiento == true || permisoMovimiento == true ){

    if(coorXold != coorXchange && coorYold != coorYchange){
        if ( coorXchange > coorXold ){
            if( coorYchange > coorYold ){

                //DIAGONAL_SUPERIOR_IZQUIERDA
                communication(identifier,
                               DIAGONAL_SUPERIOR_IZQUIERDA,
                               contadorMovDiagonales);
                direccion=DIAGONAL_SUPERIOR_IZQUIERDA;

            }else{ // coorYchange < coorY

                //DIAGONAL_INFERIOR_IZQUIERDA
                communication(identifier,
                               DIAGONAL_INFERIOR_IZQUIERDA,
                               contadorMovDiagonales);
                direccion=DIAGONAL_INFERIOR_IZQUIERDA;

            }
        }else{ // coorXchange < coorXold
    }
}
```

ANEXO B. CÓDIGO DE LOS ALGORITMOS DEL SISTEMA

```
if ( coorYchange > coorYold ){

    //DIAGONAL_SUPERIOR_DERECHA
    communication(identifier, DIAGONAL_SUPERIOR_DERECHA,
        contadorMovDiagonales);
    direccion=DIAGONAL_SUPERIOR_DERECHA;

}else{ //coorYchange < coorY

    //DIAGONAL_INFERIOR_DERECHA
    communication(identifier, DIAGONAL_INFERIOR_DERECHA,
        contadorMovDiagonales);
    direccion=DIAGONAL_INFERIOR_DERECHA;
}

}

}else{

    if ( coorXchange != coorXold && coorYchange == coorYold ){ //DCHA/
IZQ
    if ( coorXchange > coorXold ){

        //IZQUIERDA DELANTE
        communication(identifier, DELANTE,
            contadorMovClasicos);
        direccion=DELANTE;

    }else{ //coorXchange < coorXold

        //DERECHA DETRAS
        communication(identifier, DETRAS,
            contadorMovClasicos);
        direccion=DETRAS;
    }
}

if ( coorXchange == coorXold && coorYchange != coorYold ){ //ARRIBA/
ABAJO
    if ( coorYchange > coorYold ) {

        //Arriba
        communication(identifier, ARRIBA,
            contadorMovClasicos);
        direccion=ARRIBA;

    }else{ //coorYchange < coorY

        //Abajo
    }
}
```

B.6. COORDINADOR DE LOS MOVIMIENTOS DEL VEHÍCULO

```
        communication(identifier, ABAJO, contadorMovClasicos
                      );
        direccion=ABAJO;
    }
}

}

coorXchange = coorXold; coorYchange = coorYold;
coorXold = coorX; coorYold = coorY;

if ( permisoMovimiento== true){
    contadorMovClasicos = 0;
    contadorMovDiagonales = 0;
    permisoMovimiento = false;
    identifier++;
}

}

if(contadorMovClasicos>0){
    if ( coorXchange != coorXold && coorYchange == coorYold ){ //DCHA/IZQ
        if ( coorXchange > coorXold ){

            //IZQUIERDA DELANTE
            communication(identifier, DELANTE, contadorMovClasicos);
            direccion=DELANTE;

        }else{ //coorXchange < coorXold

            //DERECHA DETRAS
            communication(identifier, DETRAS, contadorMovClasicos);
            direccion=DETRAS;
        }
    }

    if ( coorXchange == coorXold && coorYchange != coorYold ){ //ARRIBA/ABAJO
        if ( coorYchange > coorYold ) {

            //Arriba
            communication(identifier, ARRIBA, contadorMovClasicos);
            direccion=ARRIBA;

        }else{ //coorYchange < coorY

            //Abajo
            communication(identifier, ABAJO, contadorMovClasicos);
            direccion=ABAJO;
        }
    }
}
```

ANEXO B. CÓDIGO DE LOS ALGORITMOS DEL SISTEMA

```
        }

    }

}

if(contadorMovDiagonales>0){

    if(coorXold != coorXchange && coorYold != coorYchange){

        if ( coorXchange > coorXold ){

            if( coorYchange > coorYold ){

                //DIAGONAL_SUPERIOR_IZQUIERDA
                communication(identifier, DIAGONAL_SUPERIOR_IZQUIERDA,
                               contadorMovDiagonales);
                direccion=DIAGONAL_SUPERIOR_IZQUIERDA;

            }else{ // coorYchange < coorY

                //DIAGONAL_INFERIOR_IZQUIERDA
                communication(identifier, DIAGONAL_INFERIOR_IZQUIERDA,
                               contadorMovDiagonales);
                direccion=DIAGONAL_INFERIOR_IZQUIERDA;
            }
        }else{ // coorXchange < coorXold

            if ( coorYchange > coorYold ){

                //DIAGONAL_SUPERIOR_DERECHA
                communication(identifier, DIAGONAL_SUPERIOR_DERECHA,
                               contadorMovDiagonales);
                direccion=DIAGONAL_SUPERIOR_DERECHA;

            }else{ //coorYchange < coorY

                //DIAGONAL_INFERIOR_DERECHA
                communication(identifier, DIAGONAL_INFERIOR_DERECHA,
                               contadorMovDiagonales);
                direccion=DIAGONAL_INFERIOR_DERECHA;
            }
        }
    }
}

fclose(file);

return 0;
}
```

Listado B.6: Coordinador de los movimientos del vehículo

B.7 Dilatación de obstáculos mediante *OpenCV*

```
/*
 * deteccionObjetosOpenCV.cpp
 *
 * Created on: 25 may. 2017
 * Author: jose
 */

#include <opencv2/opencv.hpp>
#include <iostream>
#include <string>
#include <vector>
#include <fstream>
#include "../SETTINGS.h"

using namespace cv;
using namespace std;

int objectsDetectionOpenCV(const char * scenario_img,const char * scenarioObjects_img){

    int macromap[MACROWIDTH][MACROHEIGHT];
    int blockWidth=MAPWIDTH/MACROWIDTH;
    int blockHeight=MAPHEIGHT/MACROHEIGHT;
    int x,y;

    // Cargamos la imagen de entrada
    Mat scenario;
    scenario = imread(scenario_img, CV_LOAD_IMAGE_GRAYSCALE);
    Mat scenarioObjects;
    scenarioObjects = imread(scenarioObjects_img, CV_LOAD_IMAGE_GRAYSCALE);
    Mat subtract;
    subtract(scenario, scenarioObjects, subtract);
    Mat dilated;

    for (x = 0; x < subtract.rows; x++) {
        for (y=0; y < subtract.cols; y++) {
            if(subtract.at<uchar>(x,y) > THRESHOLD){
                subtract.at<uchar>(x,y) = 255;
            }else{
                subtract.at<uchar>(x,y) = 0;
            }
        }
    }

    Mat element = getStructuringElement(MORPH_RECT, Size(200, 200) ); //MORPH_ELLIPSE //
    MORPH_CROSS
    dilate(subtract, dilated, element);
}
```

```

    std::ofstream output("map.txt");

    for (int y = 0; y < MACROHEIGHT; y++) {
        for (int x = 0; x < MACROWIDTH; x++) {
            macromap[x][y] = 0;
        }
    }

    for (int y_macromap = 0; y_macromap < MACROHEIGHT; y_macromap++) {
        for (int x_macromap = 0; x_macromap < MACROWIDTH; x_macromap++) {
            for (int y=0; y < blockHeight ; y++) {
                for (int x=0; x < blockWidth; x++) {
                    macromap[x_macromap][y_macromap]+=(dilated.at<uchar>(
                        blockHeight*y_macromap+y ,blockWidth*x_macromap+x)&0
                        x0001);
                }
            }
            //output<<macromap[x_macromap][y_macromap]<<'\t';
            if(macromap[x_macromap][y_macromap] > 0){
                output<<1;
            }else{
                output<<0;
            }
        }
        output<<endl;
    }

    output.close();

    namedWindow("Scenario", CV_WINDOW_NORMAL);
    imshow("Scenario", scenario);
    namedWindow("Scenario Objects", CV_WINDOW_NORMAL);
    imshow("Scenario Objects", scenarioObjects);
    namedWindow("Subtract", CV_WINDOW_NORMAL);
    imshow("Subtract", subtract);
    namedWindow("Dilated", CV_WINDOW_NORMAL);
    imshow("Dilated", dilated);
    imwrite("dilated.jpeg", dilated);

    waitKey(0);
}

```

Listado B.7: Dilatación de obstáculos mediante OpenCV

Anexo C

GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <<http://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version. 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

C.5. COLLECTIONS OF DOCUMENTS

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

5. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

6. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

7. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

8. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

9. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

10. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

Referencias

- [Alg12a] Artículo acerca del funcionamiento del algoritmo de cálculo de trayectoria BFS. <https://jariasf.wordpress.com/2012/02/27/algoritmo-de-busqueda-breadth-first-search/>, 2012.
- [Alg12b] Artículo acerca del funcionamiento del algoritmo de cálculo de trayectoria Djikstra. <https://jariasf.wordpress.com/2012/03/19/camino-mas-corto-algoritmo-de-dijkstra/>, 2012.
- [Alg15] Algoritmo de la librería de OpenCV acerca de la detección de objetos en función de su color. <http://www.pyimagesearch.com/2015/06/22/install-opencv-3-0-and-python-2-7-on-ubuntu/>, 2015.
- [App] Artículo acerca del proyecto de un coche eléctrico de Apple. <https://www.movilzona.es/2016/06/12/el-coche-electrico-de-apple-la-gran-amenaza-para-tesla-que-llegara-demasiado-tarde...>.
- [APr] Página web que explica el funcionamiento del algoritmo A*. <https://www.policyalmanac.org/games/aStarTutorial.es.htm>.
- [Bai] Artículo acerca de que la empresa Baidu se compromete a adelantar a Tesla. <https://mundo.sputniknews.com/tecnologia/201606281061336803-tecnologias-automoviles-futuro-ya-qin-zhang/>.
- [Can09] Artículo sobre el funcionamiento del algoritmo de Canny. <http://oefa.blogspot.com.es/2009/04/deteccion-de-bordes-algoritmo-de-canny.html>, 2009.
- [CCP] Artículo del Project Management Institute acerca de la mejora del rendimiento del proyecto con la metodología CCPM.
- [CMM] Página web acerca de la metodología de gestión de proyectos CMMI. <http://cmmiinstitute.com/>.
- [Dig] Página Web sobre los productos que ofrece la empresa Digi. <https://www.digi.com/products>.

- [dt17] OpenCV development team. OpenCV API Reference. <http://docs.opencv.org/2.4/modules/refman.html>, 2017.
- [FB310] FB36. A-star Shortest Path Algorithm (C++ recipe). <http://code.activestate.com/recipes/577457-a-star-shortest-path-algorithm/>, 2010.
- [For16] Artículo acerca de un algoritmo de detección de formas geométricas. <http://acodigo.blogspot.com.es/2014/05/deteccion-de-figuras-geometricas.html>, 2016.
- [ImR] Documentación de la lectura y escritura de imágenes en OpenCV. http://docs.opencv.org/2.4/doc/tutorials/introduction/display_image/display_image.html.
- [LEA] Página web de LEAN Enterprise Institute acerca de qué es la metodología LEAN. <http://www.lean.org/WhatsLean/>.
- [Les03] Patrick Lester. A* Pathfinding para Principiantes. <http://www.policyalmanac.org/games/articulo1.htm>, 2003.
- [Mat] Documentación del objeto Mat de la librería OpenCV. http://docs.opencv.org/2.4/modules/core/doc/basic_structures.html.
- [Mer] Artículo acerca de la semiautonomía del Mercedes-Benz Clase E. <http://www.20minutos.es/noticia/2721018/0/mercedes-benz/clase-e/autonomo/>.
- [Met] Artículo acerca de las metodologías de gestión de proyectos más utilizadas. <https://www.sinnaps.com/blog-gestion-proyectos/metodologias-herramientas-gestion>.
- [Mic] Artículo acerca de la intención de Microsoft en desarrollar software para vehículos autónomos. <https://hipertextual.com/2016/06/microsoft-coche-autonomo>.
- [Ope05] Práctica de la Universidad de Jaen que habla sobre la detección de bordes en una imagen. http://www4.ujaen.es/~satorres/practicas/practica3_vc.pdf, 2005.
- [Pat10] Tema acerca del reconocimiento de patrones. <http://grupo.us.es/gtocoma/pid/tema7.pdf>, 2010.
- [Pet] Página oficial de la herramienta Petalinux. <https://www.xilinx.com/products/design-tools/embedded-software/petalinux-sdk.html>.
- [PMB] Página web del Project Management Institute que supervisa el libro PMBOK. <https://www.pmi.org/pmbok-guide-standards>.

REFERENCIAS

- [Res] Funcionaes a las que da soporte la herramienta Vivado HLS de la librería OpenCV. <http://www.wiki.xilinx.com/HLS+Video+Library>.
- [SAEa] Estandar sobre los 6 niveles de autonomía por el organismo SAE. http://standards.sae.org/j3016_201401/.
- [SAEb] Página oficial de la Sociedad de Ingenieros Automotrices. <http://www.sae.org/>.
- [Scr] Página web acerca de cómo funciona la tecnología Scrum. <https://proyectosagiles.org/como-funciona-scrum/>.
- [SMA] Página web acerca de la metodología de gestión de proyectos SMART. <http://smartmethodology.org/>.
- [Tes] Página oficial de Tesla acerca de sus modelos de coche. <https://www.tesla.com/presskit/autopilot>.
- [THCS01] Ronald L. Rivest Thomas H. Cormen, Charles E. Leiserson y Clifford Stein. *Introduction to Algorithms, Second Edition*, capítulo 24.1: The Bellman-Ford algorithm, páginas 588–592. 2001.
- [Viva] Página web acerca de la herramienta Vivado Design Suite de la empresa Xillinx, Inc. <https://www.xilinx.com/products/design-tools/vivado.html>.
- [Vivb] Página web acerca de la herramienta Vivado HLS de la empresa Xi-llinx, Inc. <https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html>.
- [Vivc] Página web acerca de la herramienta Vivado Software Development Kit de la empresa Xillinx, Inc. <https://www.xilinx.com/products/design-tools/embedded-software/sdk.html>.
- [Why] Informe de SEI acerca de la utilización conjunta de las metodologías de desarrollo Scrum y CMMI. http://resources.sei.cmu.edu/asset_files/TechnicalNote/2008_004_001_14924.pdf.
- [XCT] Plataforma de configuración de módulos XBee XCTU. <https://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu>.
- [Xu16] Xueqiao (Joe) Xu. Librería visual acerca de algoritmos de búsqueda. <https://qiao.github.io/PathFinding.js/visual/>, 2016.

Este documento fue editado y tipografiado con L^AT_EX empleando
la clase **esi-tfg** (versión 0.20170705) que se puede encontrar en:
https://bitbucket.org/arco_group/esi-tfg

[respeta esta atribución al autor]

