Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño de un sistema de guiado de un robot móvil basado en Arduino

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Brandon Andrés Lara Erazo

Tutores: Joaquín Gracia Morán

Juan Carlos Ruiz García

Septiembre de 2018

Resumen

Este proyecto trata de desarrollar un sistema de guiado para un robot basado en Arduino, en el cual no solamente se ha desarrollado este sistema de guiado, si no que además se ha aprendido el funcionamiento de varios componentes compatibles con Arduino entre los cuales se encuentran los componentes de, así como el lenguaje de programación de Arduino, y su entorno de desarrollo. Una vez aprendidas estas tecnologías se ha desarrollado el sistema de guiado en sí, el cual consiste en un sistema que sea capaz de guiar un coche sobre un circuito dibujado en el suelo. El sistema de guiado se ha lleva a cabo mediante una serie de componentes hardware, y una placa SunFounder compatible con Arduino, los cuales serán controlados y gestionados mediante el software que se ha desarrollado en este proyecto, el cual esta escrito en el lenguaje de programación Arduino.

Palabras clave: Hardware, Arduino, desarrollo, aprendido, SunFounder, Software, lenguaje, programación, proyecto, guiado, robot, coche.

Índices

Contenido

Índice de ilustraciones

Índice de Tablas

1. Introducción

Actualmente, hay un amplio catálogo de microcontroladores, y estos a su vez nos ofrecen gran cantidad de posibilidades a la hora de desarrollar un proyecto, ya sea en el ámbito profesional o académico.

Una de las compañías de diseño y fabricación de placas de desarrollo, es Arduino

1.1. Motivación

La motivación que ha llevado a la elaboración de este proyecto es el ampliar aún más los estudios realizados en el Grado de Ingeniería Informática, ya que este tipo de proyecto no se ve en ninguna asignatura de la carrera. A medida que se obtienen conocimientos en informática, la sensación de que cada vez tenemos más posibilidades por aprender aumenta. Este hecho puede ser motivador para algunos o frustrante para otros. Lo que sí es cierto es que puede llevar a un estudiante a aprender algo que nunca se podría haber imaginado, y que se le abran nuevas posibilidades dentro del campo de la informática.

En este sentido, durante los estudios de grado, fuera de las asignaturas optativas no se aprende robótica. Este proyecto ha sido una buena oportunidad para conseguir estudiar algo nuevo e interesante que, además, está relacionado con la ingeniería informática.

1.2. Objetivos

El objetivo principal de este proyecto ha sido desarrollar un sistema de guiado para un robot basado en Arduino. Este robot es un coche "inteligente" compuesto por varias placas, microcontroladores, motores, y sensores que, conectados entre sí, y mediante el entorno de desarrollo de Arduino, será capaz de seguir, de forma autónoma, un circuito dibujado en el suelo.

Además, también existen varios objetivos secundarios que ayudarán a conseguir el objetivo principal. Uno de estos objetivos secundarios ha sido el aprender el funcionamiento de todos los elementos hardware de los que consta el robot inteligente. Además, también se busca alcanzar el conocimiento necesario en el entorno de desarrollo de Arduino para poder programar el sistema de guiado, no sin antes haber conseguido entender cómo funciona la comunicación del hardware con nuestro programa en el IDE de Arduino. Una vez entendido el funcionamiento, tanto de los elementos hardware como de los elementos software, el siguiente objetivo ha sido diseñar y desarrollar el programa del sistema de guiado. Finalmente, una vez terminado el programa, se han realizado diferentes pruebas para comprobar su correcto funcionamiento, y corregirlo en caso de encontrar errores, para completar el desarrollo de forma adecuada.

1.3. Metodología

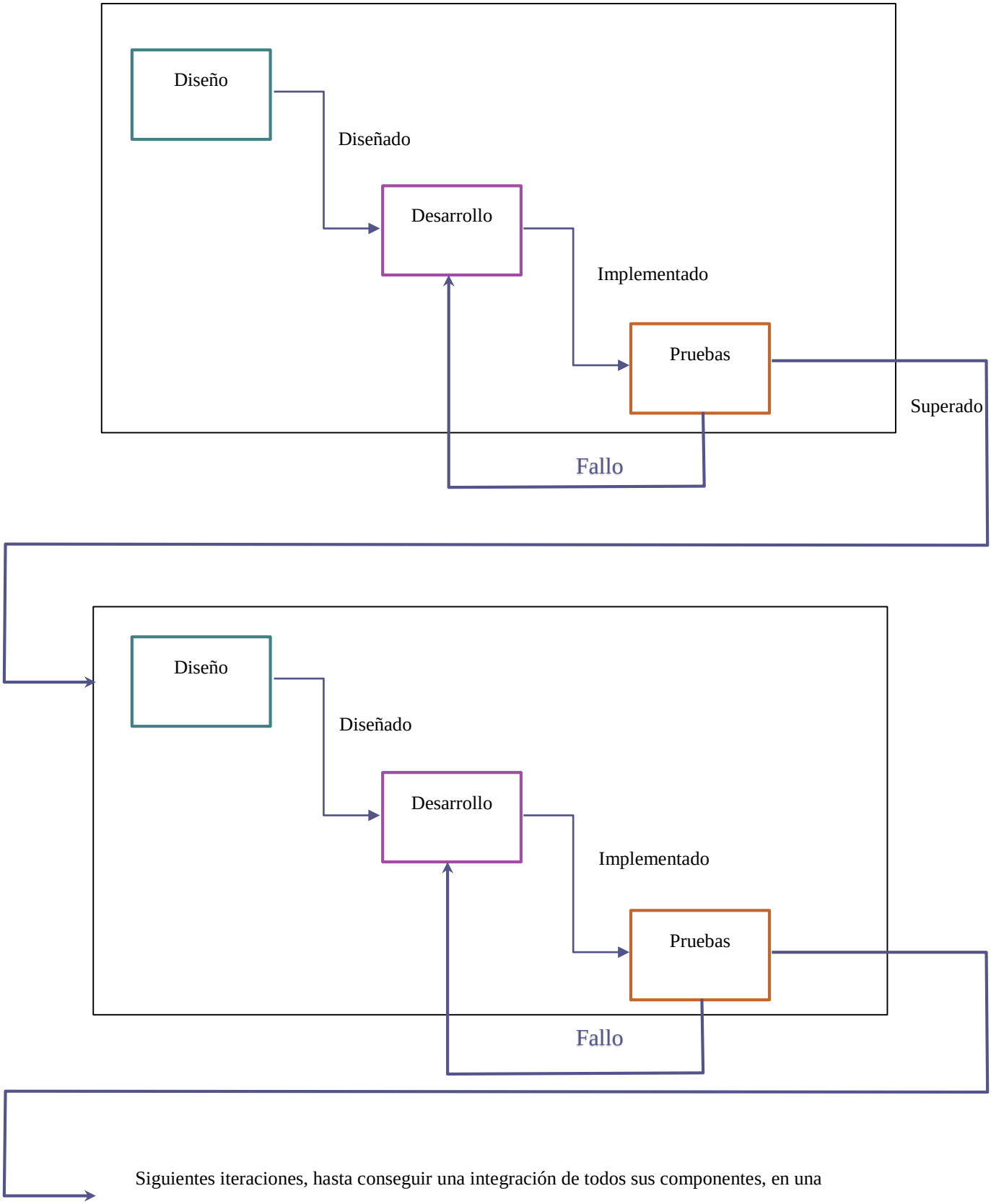
Para llevar a cabo este proyecto, se ha intentado tener un equilibrio entre productividad, agilidad, y cumplimiento de plazos. Por ello se ha elegido realizar este proyecto de forma iterativa incremental, en la que se basa el desarrollo ágil. Se ha decidido reducir las fases de cada iteración ya que no necesitamos hacer un análisis previo al comenzar con el diseño. Por lo tanto, las fases que se han establecido por cada iteración son las siguientes:

- **Diseño**. En esta fase se ha especificado cuál será el diseño de los componentes software y hardware de nuestro sistema de guiado, y cómo van a ser las interacciones entre el hardware del coche inteligente y el software a desarrollar. En esta fase también se ha hecho una estimación del alcance y del coste temporal del proyecto. Además, en esta fase se analizan las especificaciones para cada

iteración, lo que nos lleva a analizar el software a construir desde diversos puntos de vista, pero sin llegarlo a construir. Por lo tanto, nos da una medida, por cada iteración, que nos sirve como base para el desarrollo y las pruebas.

- **Desarrollo**. En esta fase se ha procedido a implementar todo aquello que se ha pensado y se ha especificado previamente. A partir de esta fase se ha procedido de forma incremental con las siguientes fases para desarrollar el programa completo, con una calidad óptima, con cada incremento. Para que esto no suponga un gran impacto temporal en el proyecto, se ha desarrollado poco a poco el sistema, es decir con cada incremento se iban añadiendo pequeños componentes en los que se ha dividido el sistema, que pasaban a fase de pruebas en cuanto se tenía una versión con el componente integrado en el sistema. Sin embargo, si había un error o problema en las pruebas, no se hacía un incremento en el proyecto. Se seguía con el desarrollo y pruebas sobre el componente del incremento actual.
- **Pruebas**. En esta fase se han realizado diversas pruebas sobre el sistema de guiado, dada la forma de proceder, incremental mediante varias iteraciones dependiendo de los componentes del sistema. Para cada incremento se necesitaban unas pruebas específicas dependiendo del componente que se añadiese con el incremento. Por lo tanto, cada incremento tenía asociado una batería de pruebas, las cuales se han utilizado en incrementos posteriores para asegurarse que los cambios no han afectado a otros componentes que ya funcionaban correctamente en anteriores iteraciones. Si un desarrollo no conseguía pasar la fase de pruebas sin errores o problemas, volvía a la anterior fase de desarrollo para aplicar las correcciones, y después volver a la fase de pruebas para probar que después de las correcciones todo funciona correctamente y está listo para pasar al siguiente incremento.

Además, antes de empezar con los incrementos en nuestro proyecto, había que planificarlos plazos, objetivos e iteraciones. También, como no se tenía conocimiento acerca de la tecnología, en los incrementos se han incluido varias iteraciones, en las cuales se aprendía el lenguaje utilizado, así como la tecnología y el hardware. Todo esto de forma práctica y con desarrollos de ejemplo que servían para comprender mejor el trabajo que se va a realizar, y así planificarlo correctamente e ir definiendo los objetivos de los incrementos. En la ilustración 1 se puede ver un esquema de este proceso.



Siguientes iteraciones, hasta conseguir una integración de todos sus componentes, en una versión final del proyecto de muy alta calidad.

Ilustración 1 Modelo iterativo incremental

Ventajas de utilizar esta metodología de trabajo:

Para empezar, utilizando este modelo, vamos obteniendo una versión completa y refinada a medida que vamos incrementando las iteraciones realizadas sobre nuestro proyecto. En este sentido, no es necesario que especifiquemos los requisitos del proyecto inicialmente, es decir, se puede empezar a desarrollar bastante pronto en el ciclo de vida del proyecto, lo que implica que podemos tener primeras versiones unitarias en el proceso de pruebas en un momento muy temprano, lo cual favorece el aprendizaje de las tecnologías, así como la gestión del tiempo y riesgos. Además, la planificación del proyecto resulta más sencilla, y en nuestro caso más aun debido a que tenemos menos fases en cada iteración. Por último, una de las ventajas más importantes es que el resultado de nuestro trabajo es de más calidad, debido a que continuamente se está probando al terminar cada desarrollo, lo que permite minimizar los errores e introducir mejoras continuas.

Desventajas:

Se requiere la elaboración de objetivos claramente especificados, ya que de otra forma se pierde la ventaja de poder gestionar más fácilmente el tiempo, los recursos y las versiones del proyecto poniendo en riesgo los plazos y recursos necesarios. Por otra parte, tenemos versiones del proyecto bastante tempranas, y con ello podemos ir sacando información acerca del proyecto, y mejorándolo constantemente. Sin embargo, se requiere tiempo para hacer varias veces las fases de diseño, desarrollo y pruebas, lo que puede provocar que nuestro proyecto se alargue más en el tiempo.

2. Estado del arte

En la actualidad, el guiado automático mediante microcontroladores, sensores y software ha cobrado una especial relevancia en la industria del automóvil. Una prueba es que grandes empresas no solo del sector automovilístico, sino también del sector tecnológico han apostado fuertemente por este tipo de tecnologías. En este sentido, estamos viviendo una época en la cual se están produciendo grandes avances en el desarrollo de vehículos autónomos. Todo esto se debe a que los accidentes de tráfico son una de las principales causas de mortalidad en el mundo, existiendo una fuerte necesidad por mejorar los sistemas de seguridad de los automóviles. Una vez más, la tecnología va a tener un papel fundamental en la mejora e innovación de los sistemas de seguridad de un vehículo.

Últimamente se ha conseguido un alto nivel de autonomía en los vehículos. Incluso ya están disponibles para su compra vehículos que son capaces de conducir de forma automática. Suelen estar compuestos de sensores, radares, cámaras y sistemas informáticos, que, funcionando de forma conjunta, permiten que el coche funcione de forma autónoma, sabiendo dónde está el coche, cuál es su entorno e incluso conocer el estado en el que se encuentra el conductor. Todo esto sirve para anticiparse a posibles accidentes, mejorar la seguridad, reducir los fallos humanos, o simplemente para no tener que conducir.

Un elemento de seguridad en este tipo de vehículos sería el de guiado mediante sensores para evitar salirse de la vía. El hecho de tener localizados los carriles por los cuales está circulando el vehículo es algo similar a lo que se ha desarrollado en el presente Trabajo Final de grado. Sin embargo, aunque parezca similar, el estado actual de la tecnología que utilizan los vehículos autónomos es mucho más avanzado de lo que se ha implementado en este TFG, aunque la idea sea la misma. Hoy en día este tipo de guiado se hace mediante cámaras y sensores de imagen que crean imágenes en tres dimensiones de los alrededores del vehículo, y mediante una avanzada inteligencia artificial, se pueden interpretar las imágenes que producen las cámaras y sensores, con el fin de al vehículo por la vía o carril adecuado.

Ilustración 2 Predicción de disponibilidad de coches autónomos.

Arduino se ha convertido actualmente en uno de los mayores referentes tecnológicos, así como de la comunidad Open Source. Esto es debido a la facilidad que se tiene para adquirir a buen precio las placas Arduino y sus kits de desarrollo. Los módulos y sensores son muy fáciles de encontrar en las principales tiendas online en todo el mundo, así como en la web de Arduino. También el desarrollo de proyectos Arduino se ve beneficiado por una plataforma de desarrollo sencilla, y que además cuenta con una gran comunidad detrás, activa en páginas webs, foros y redes sociales, donde podemos buscar información, ayuda, o consejo para comenzar a desarrollar proyectos con Arduino, o llevar nuestras ideas a otro nivel. Así mismo, Arduino actualmente nos ofrece una versatilidad muy amplia, no solo debido a que es un proyecto de código abierto, sino que desde 2005 se han desarrollado diferentes placas, módulos, sensores y proyectos que permiten implementar casi cualquier cosa que nos podamos imaginar.

En concreto, uno de los proyectos más extendidos, y que está estrechamente relacionado con los vehículos autónomos y las tecnologías que utilizan éstos, son los

robots automáticos, o inteligentes, que mucha gente lleva desarrollando desde hace mucho tiempo. En la actualidad existen kits de desarrollo enfocados a este tipo de robots, con placas, sensores, módulos y motores diseñados específicamente para este tipo de proyectos. También existe mucha documentación, que permite conocer las posibilidades que nos ofrece el desarrollo en esta plataforma, así como tutoriales que ayudan a la etapa de aprendizaje, y gran cantidad de manuales que nos sirven de guía en la construcción del robot y su sistema de guiado. Ahora mismo, el desarrollo de este tipo de proyectos nos sirve como introducción a las tecnologías empleadas en los vehículos autónomos. Sin embargo, con los grandes avances que hay en el campo de la domótica y los asistentes virtuales, también podríamos utilizar esta tecnología en un uso más doméstico, como por ejemplo el de un robot que, mediante unas líneas en el suelo de casa, fuera capaz de realizar diferentes tareas cotidianas o de apoyo al usuario.

A pesar de que hoy en día Arduino está muy extendido alrededor del mundo, con una gran comunidad detrás, también es cierto que en un inicio se pensaba que todo lo que envolvía a Arduino iba a ser algo mucho más grande, conocido y que iba triunfar mucho más de lo que en realidad lo ha hecho. Con esto no se da a entender que Arduino no haya alcanzado un buen nivel de éxito, si no que podría haber sido una plataforma más exitosa aún.

Este problema puede haber sido causado por no haber podido llegar a más tipos de público. También ha tenido que compartir cierto público con Raspberry pi, una placa similar a Arduino, pero con una potencia de cálculo mayor. Por último, tal vez, en la mayoría de las ocasiones se le da un enfoque educativo. Este hecho provoca que se limiten sus casos de uso, y por ello no crecer, no solo como plataforma, sino como comunidad.

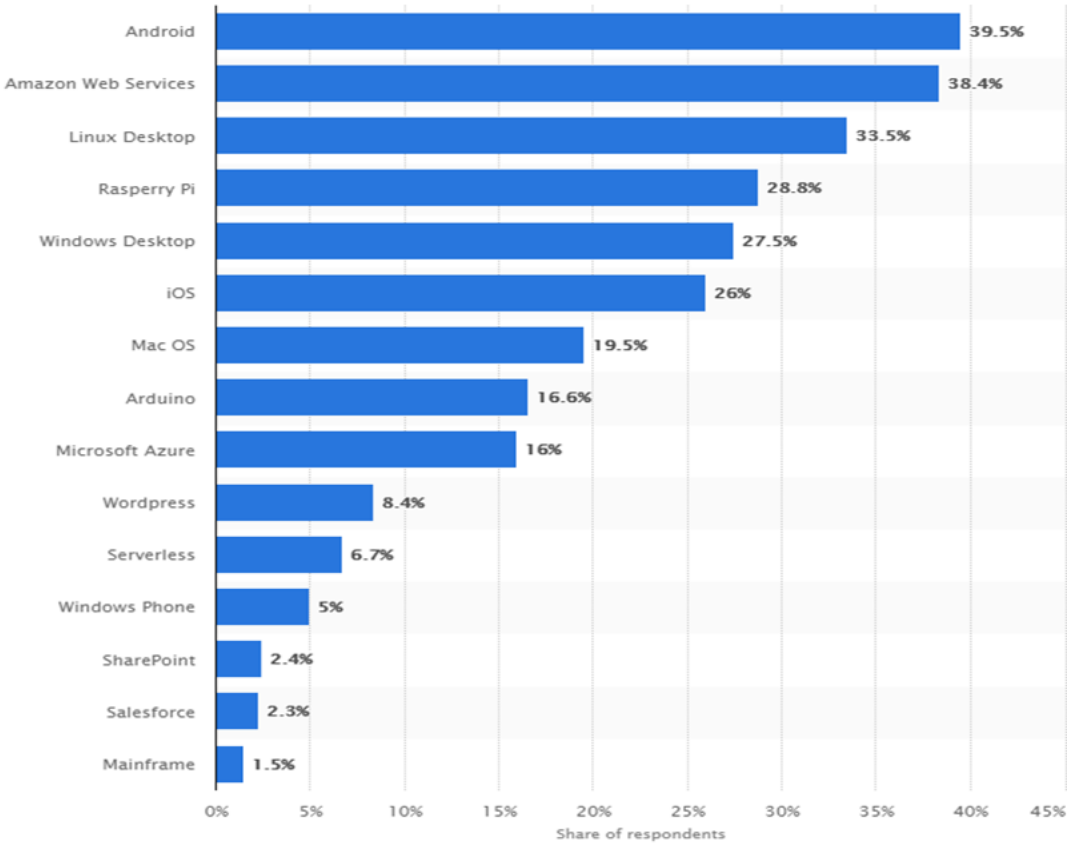


Ilustración 3 Gráfico de las plataformas más solicitadas en todo el mundo

3. Análisis del problema

En este capítulo se va a analizar en profundidad el problema al que hay que enfrentarse a la hora de desarrollar un sistema de guiado, así como se explicará la solución a estos problemas. En este sentido, vamos a analizar todas las oportunidades que nos ofrecen las tecnologías de las que se ha hablado anteriormente, y las que se van a utilizar para el desarrollo de este proyecto.

La ilustración 4 muestra un esquema de los diferentes escenarios en los que se va a encontrar el coche cuando haga uso del sistema de guiado. El coche circulará por una pista dibujada en el suelo, y que seguirá en función de unas ciertas métricas de calidad, ya sea la distancia de alejamiento de la pista, la fluidez en la circulación, o la ausencia de errores. Dicho esto, dentro de este escenario el coche ha de superar las diferentes circunstancias que se le presenten a lo largo del recorrido. Es decir, en principio, el coche simplemente va a tener que seguir una línea dibujada en el suelo. Sin embargo, para realizar este seguimiento, el coche ha de ser capaz que girar a la derecha o a la izquierda si hay alguna curva en esas direcciones, o seguir recto si está circulando sobre una recta. Estos giros en diferentes direcciones no solamente van a ser izquierda o derecha, sino que además van a poder hacerse mediante un ángulo de giro, y así poder adaptarse a los diferentes tipos de curvas que se pueden haber dibujado en el suelo, ya que unas pueden ser más cerradas que otras. Por otro lado, también debe de ser capaz de cambiar la velocidad.

Todo este funcionamiento va a ser controlado mediante una placa SunFounder compatible con Arduino, y para detectar las diferentes circunstancias en las que puede encontrarse el coche, se utilizarán diferentes módulos sensores.

Ilustración 4 Posibles escenarios de circulación del coche

Identificar los requerimientos del sistema de guiado, no solo ayuda a implementar correctamente la solución del problema que se nos presenta en el análisis de este, sino que además va a ayudar en otras facetas del proyecto, como, por ejemplo, la planificación y estimación de costes temporales de desarrollo, conseguir cierto grado de calidad a la hora de desarrollar el proyecto, o en su resultado final.

Así pues, se han identificado varios requisitos que hemos agrupado en requisitos funcionales y requisitos no funcionales.

Requisitos funcionales

Estos requisitos se refieren a las funcionalidades que se esperan del sistema, y las acciones que va a tener que realizar. En este caso tenemos diferentes tipos de componentes: hardware y software, de los cuales podemos determinar los siguientes requisitos:

- El sistema debe ser capaz de guiar al coche por una pista dibujada en el suelo.

- El sistema utilizará diferentes sensores para poder seguir el trazado.
- La pista presentará diferentes formas y trazados, que el sistema va a tener que seguir.
- El sistema debe ser capaz de funcionar en una placa SunFounder basada en Arduino.

Requisitos no funcionales

Este tipo de requisitos trata de las propiedades intrínsecas que va a tener que cumplir el sistema, para poder satisfacer ciertos criterios de calidad en el resultado final:

- Se requiere que el sistema tenga un cierto grado de fiabilidad, para que pueda circular durante un tiempo determinado sin que el nivel de desempeño se vea afectado.
- El sistema tiene que cumplir todos los requisitos funcionales. Por lo tanto, se espera un alto grado de funcionalidad.
- Se necesita un buen nivel de eficiencia en el proyecto para proporcionar un rendimiento adecuado consumiendo los recursos mínimos.
- Al ser un sistema con varios componentes tanto software como hardware, y que va a estar sometidos a pruebas continuamente, se espera que el sistema de guiado tenga un muy buen nivel de mantenibilidad.

3.1. Análisis de soluciones

Hay varias soluciones posibles para el sistema de guiado. Por ejemplo, este proyecto podría llevarse a cabo mediante una placa Raspberry pi, sustituyendo la placa SunFounder. Sin embargo, uno de los requisitos del sistema de guiado es que se tiene que utilizar una placa SunFounder compatible con Arduino. La Raspberry pi es más parecido a un pequeño ordenador que a un microcontrolador. Además, Arduino es más idóneo para un proyecto de robótica, ya que tiene una multitud de conexiones, sensores y actuadores compatibles, lo cual facilita el trabajo con dispositivos electrónicos, y su posterior programación.

Ya dentro de Arduino, para llevar a cabo este proyecto, tenemos varios sensores o tecnologías con las cuales es posible desarrollar un sistema de guiado de estas

características. En primer lugar, mediante un led que emita luz y un sensor fotosensible podríamos crear un sistema de guiado. También mediante un sensor fotoeléctrico, el cual nos permite saber el color de la superficie que hay justo debajo de este sensor. Esta solución es más elegante que la anterior y además puede llegar a ser más sencilla de implementar, debido a la naturaleza del propio sensor que puede facilitar enormemente el trabajo a realizar. Finalmente, una solución bastante sofisticada es utilizar una cámara integrada en el coche, la cual permita procesar las imágenes que va recibiendo en tiempo real, y así poder conocer el entorno sobre el que está circulando el coche. Posiblemente esta solución sea muy costosa y compleja de llevar a cabo, pero puede ser la que tenga una tecnología superior a la del resto. Sin embargo, la relación beneficio/coste no es tan alta como para contemplar esta solución.

3.2. Solución propuesta

Como el título del presente Trabaja de Fin de Grado indica, la solución que se ha propuesto para la problemática es la de un sistema de guiado de un robot basado en Arduino, en el cual se van a emplear una serie de sensores fotoeléctricos que servirán al sistema de guiado para poder saber sobre qué superficie está circulando el coche en cualquier momento, y de esta forma poder guiar al robot por donde queramos. En este robot también debería haber otros componentes que puedan ser controlados mediante un microcontrolador basado en Arduino, en este caso una placa de desarrollo SunFounder, compatible con Arduino. Por lo tanto, todos los componentes que puedan cambiar el comportamiento del coche al circular tienen que poder ser controlados mediante el software que se tiene que desarrollar para el sistema de guiado, el cual va a ser desarrollado mediante el entorno de desarrollo de Arduino, y su lenguaje de programación. Esta solución se ha escogido porque además de ser la más sencilla de llevar a cabo, está entre las soluciones más efectivas, lo que significa que, si se realiza correctamente, el resultado puede ser excelentes con unos costes temporales inferiores.

3.3. Plan de trabajo

La planificación se divide en dos partes diferenciadas: la etapa de aprendizaje y la etapa de desarrollo del proyecto. Dentro de estas etapas se establecen diferentes objetivos, que se alinearán con el modelo iterativo e incremental que vamos a seguir para el desarrollo del proyecto, además de definir plazos referentes a los objetivos y las fases de cada iteración.

La etapa de aprendizaje, además de servir para aprender cómo funcionan las diferentes herramientas, tanto software como hardware, que se van a utilizar, también ha servido como aprendizaje para hacer la planificación de la etapa de desarrollo del proyecto más precisa y realista.

Etapa de aprendizaje

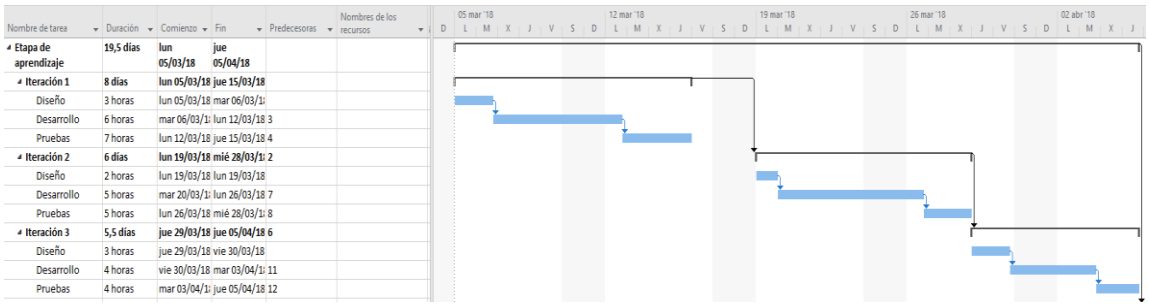
Esta etapa se había pensado que debería durar 48 horas durante un mes y medio, en las cuales se establecería una introducción de un par de semanas con el fin de aprender las partes teóricas de las tecnologías utilizadas. Además, se aprovecharía esta introducción para analizar el robot, de qué elementos está compuesto, y cómo afrontar el aprendizaje sobre cada componente del robot. En concreto, de estas dos semanas se dedicarían cinco días a la semana, una hora al día, haciendo un total de 10 horas para la introducción a Arduino. A partir de aquí todo sería práctico, siguiendo el modelo propuesto. La primera iteración se dedicaría a los motores y servomotores que utiliza el robot para desplazarse y cambiar de dirección y sentido, desde el punto de vista del hardware (conexiones, interacción de la placa Arduino con estos componentes), y el software que desarrollaremos. Esta iteración se planificaría para la tercera y cuarta semana de la etapa de aprendizaje, dedicando dos horas al día, cuatro días a la semana, un total de 16 horas.

En la siguiente iteración habría que centrarse en los sensores. Se trata de un sistema de guiado el cual sigue una línea en el suelo, para lo que se necesita un sensor que haga un seguimiento del suelo sobre el que está desplazándose el robot. Sin embargo, existen muchos otros sensores que se utilizan para el guiado de nuestro robot Arduino, y que pueden ser de ayuda para la comprensión del conjunto de piezas del robot, de cómo

interactúan, y cómo se programan. Para ello se ha planificado una pequeña iteración por cada sensor de cuatro horas cada una. El objetivo era practicar con los sensores, fotosensibles, de proximidad y de seguimiento del suelo. Por lo tanto, habiendo tres iteraciones, se dedicarían doce horas. En total, trabajando dos horas al día, tres días a la semana durante dos semanas.

Finalmente, se ha planificado una iteración de la etapa de aprendizaje en la cual se va a practicar, aprender y experimentar con los sensores, motores y servos que se utilizarían en el proyecto. Esta iteración duraría una semana de la cual se trabajarían cinco días, dos horas al día, haciendo un total de diez horas.

Recapitulando, en total se habría pensado hacer tres iteraciones durante la etapa de aprendizaje, que durarían cuatro semanas, de las cuales se trabajarían aproximadamente diecinueve días, tal y como se indica en el siguiente gráfico, donde además se indican las horas que se dedicarían a las fases de cada iteración.



Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
Etapa de aprendizaje	**19,5 días**	**lun 05/03/18**	**jue 05/04/18**		
Iteración 1	**8 días**	**lun 05/03/18**	**jue 15/03/18**		
Diseño	3 horas	lun 05/03/18	mar 06/03/1		
Desarrollo	6 horas	mar 06/03/1	lun 12/03/18	3	
Pruebas	7 horas	lun 12/03/18	jue 15/03/18	4	
Iteración 2	**6 días**	**lun 19/03/18**	**mié 28/03/1**	**2**	
Diseño	2 horas	lun 19/03/18	lun 19/03/18		
Desarrollo	5 horas	mar 20/03/1	lun 26/03/18	7	
Pruebas	5 horas	lun 26/03/18	mié 28/03/1	8	
Iteración 3	**5,5 días**	**jue 29/03/18**	**jue 05/04/18**	**6**	
Diseño	3 horas	jue 29/03/18	vie 30/03/18		
Desarrollo	4 horas	vie 30/03/18	mar 03/04/1	11	
Pruebas	4 horas	mar 03/04/1	jue 05/04/18	12	

Ilustración 5 Planificación de la Etapa de aprendizaje

Etapa de desarrollo

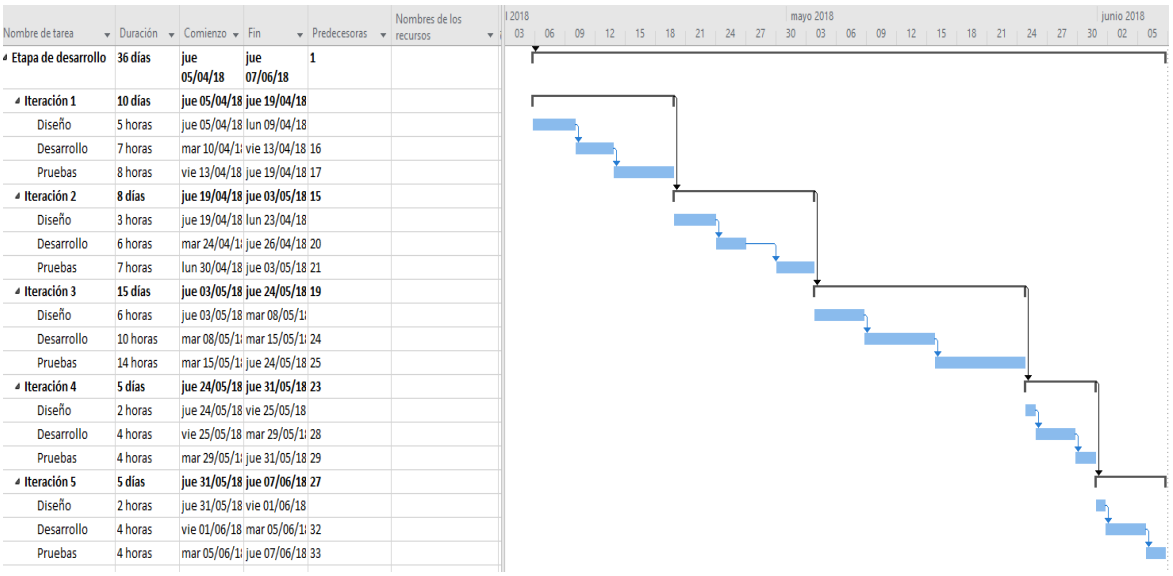
Una vez terminada la etapa de aprendizaje, se ha planificado la etapa de desarrollo. Gracias a las conclusiones sacadas de la etapa de aprendizaje, esta planificación podrá ser más acertada y completa. La etapa de desarrollo se planificó con una duración de aproximadamente dos meses, en los cuales se terminaría por completo el desarrollo del sistema de guiado del proyecto. La primera iteración se dedicaría al tema de sensores, montarlos y programarlos correctamente. Para ello utilizaríamos dos semanas, cinco días cada semana, dos horas al día, lo que haría un total de 20 horas para este incremento.

En el segundo incremento programaríamos los motores y servos según la disposición que hemos elegido para las conexiones con la placa Arduino. Para esta iteración se utilizarían una semana, cuatro días a la semana, dos horas al día. Por lo tanto, el total de horas para esta iteración es de dieciséis horas.

En la tercera iteración se programarían en conjunto los motores, servos y sensores para que nuestro robot siga una línea dibujada en el suelo. Sin embargo, esta sería una primera iteración funcional de nuestro proyecto. Por lo tanto, no sería definitiva, pero nos serviría como una primera aproximación al resultado final. Esta iteración tendría una duración de tres semanas, de las cuales se trabajaría cinco días a la semana, dos horas diarias, lo que nos daría un total de treinta horas.

Una vez obtenida la primera aproximación al sistema de guiado, se debería refinar en versiones cada vez más correctas y de mejor calidad. Para ello, se han planificado dos iteraciones más, que se pueden extender con más iteraciones al final de la etapa de desarrollo. Estas iteraciones deberían tener una duración de una semana cada una, de las cuales se dedicarían cinco días a la semana, y trabajando dos horas al día, daría una duración de diez horas cada iteración.

Un resumen de la duración de esta etapa se puede ver en la siguiente ilustración.



Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
Etapa de desarrollo	**36 días**	**jue 05/04/18**	**jue 07/06/18**	**1**	
Iteración 1	**10 días**	**jue 05/04/18**	**jue 19/04/18**		
Diseño	5 horas	jue 05/04/18	lun 09/04/18		
Desarrollo	7 horas	mar 10/04/1	vie 13/04/18	16	
Pruebas	8 horas	vie 13/04/18	jue 19/04/18	17	
Iteración 2	**8 días**	**jue 19/04/18**	**jue 03/05/18**	**15**	
Diseño	3 horas	jue 19/04/18	lun 23/04/18		
Desarrollo	6 horas	mar 24/04/1	jue 26/04/18	20	
Pruebas	7 horas	lun 30/04/18	jue 03/05/18	21	
Iteración 3	**15 días**	**jue 03/05/18**	**jue 24/05/18**	**19**	
Diseño	6 horas	jue 03/05/18	mar 08/05/1		
Desarrollo	10 horas	mar 08/05/1	mar 15/05/1	24	
Pruebas	14 horas	mar 15/05/1	jue 24/05/18	25	
Iteración 4	**5 días**	**jue 24/05/18**	**jue 31/05/18**	**23**	
Diseño	2 horas	jue 24/05/18	vie 25/05/18		
Desarrollo	4 horas	vie 25/05/18	mar 29/05/1	28	
Pruebas	4 horas	mar 29/05/1	jue 31/05/18	29	
Iteración 5	**5 días**	**jue 31/05/18**	**jue 07/06/18**	**27**	
Diseño	2 horas	jue 31/05/18	vie 01/06/18		
Desarrollo	4 horas	vie 01/06/18	mar 05/06/1	32	
Pruebas	4 horas	mar 05/06/1	jue 07/06/18	33	

Ilustración 6 Planificación de la etapa de desarrollo

4. Diseño de la solución

A continuación, se van a explicar el diseño de la solución y las decisiones sobre cómo hay que realizar el sistema de guiado para el robot, tanto por la parte software como hardware, y todo esto desde el punto de vista de la arquitectura del sistema y también desde un punto de vista del diseño en detalle.

En un primer lugar se ha dividido el sistema en dos partes bastante diferenciadas: la parte hardware, y la parte software. A continuación, se puede ver una ilustración de la división en dos partes del sistema de guiado, y también de sus componentes, tanto software como hardware.

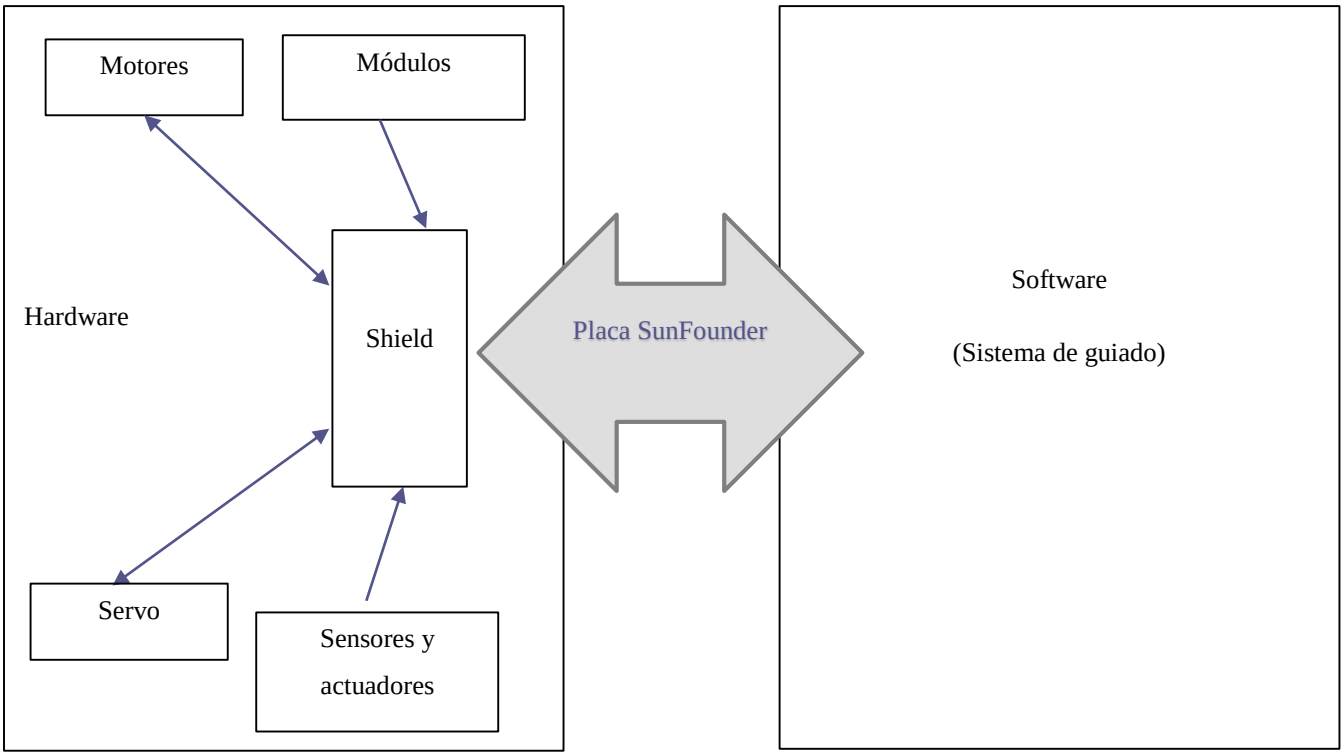


Ilustración 7 Partes del sistema de guiado

En la Ilustración 7 se puede observar la división en dos bloques del sistema de guiado y de los componentes de la parte hardware, y cómo el software del sistema de guiado es un componente único, el cual controla los componentes del hardware a través de la placa compatible con Arduino, SunFounder. En la parte de hardware se puede apreciar que los componentes están conectados entre sí. Los módulos y sensores solo tienen una dirección, que es la de salida hacia el shield. Sin embargo, los motores y el servo tienen tanto salidas como entradas, y todas están conectadas directamente con el shield, a excepción de los motores, los cuales tienen de intermediario entre los motores y el shield un módulo controlador. Algunas de estas salidas y entradas, según el componente concreto, van a poder ser controlables por el shield.

En los siguientes apartados se va a explicar el diseño de la arquitectura del sistema tanto de la parte software, y sus componentes, como el hardware, así como su diseño en detalle.

4.1. Arquitectura del sistema

En este subapartado se va a detallar el diseño de la arquitectura del sistema de guiado, describiendo las dos partes, la parte de hardware, la parte software, y además la parte de nexo entre estas dos partes.

Hardware

Mediante los componentes que se nos proporcionaba, se ha diseñado la arquitectura hardware de tal forma que tuviésemos un shield de interfaz entre la placa SunFounder y el resto de los componentes. Como se ha comentado antes, también hay que tener en cuenta que los motores tienen un módulo controlador, el cual se tiene que ubicar entre los motores y el shield.

Por otro lado, también se tienen los sensores que recogen información del medio en el que se encuentra el robot. Esta información, después de ser procesada, se envía directamente al shield, para que mediante el software subido a la placa SunFounder, el robot cambie su comportamiento, ya sea dirección, velocidad o sentido de la marcha.

En la siguiente ilustración de detalla la arquitectura hardware del robot.

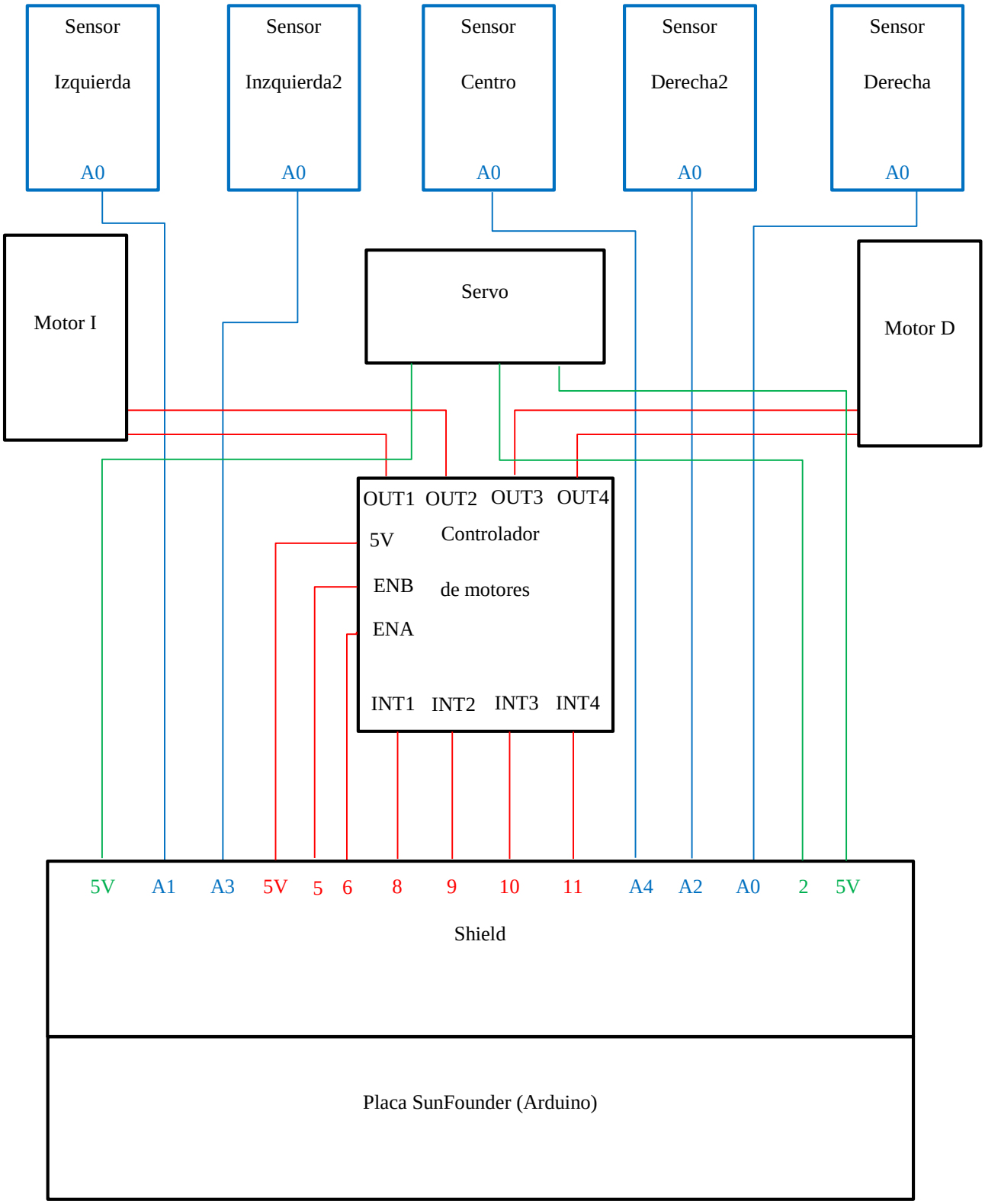


Ilustración 8 Diseño arquitectura hardware

A continuación, se va a explicar la arquitectura hardware que se ha establecido para el sistema de guiado, que es la que se muestra en la ilustración 8. En este esquema, se han obviado diversos componentes hardware triviales del robot, como son el interruptor de encendido y apagado del robot, así como el conversor que se sitúa entre la fuente de alimentación del robot y la placa SunFounder, para adaptar el voltaje que recibe la placa SunFounder de las baterías.

Se han utilizado cinco sensores de seguimiento fotoeléctricos, los cuales se han conectado directamente con el shield apilado encima de la placa SunFounder. En la ilustración 8 se han simplificado estas conexiones. En esta ilustración parece que hay una conexión por cada sensor cuando en realidad físicamente existen tres conexiones por cada sensor. Sin embargo, para que la ilustración tuviese un aspecto lo más limpio posible, se han simplificado estas conexiones. Además, se han resumido las salidas de 5V y GND de cada sensor en su salida analógica A0.

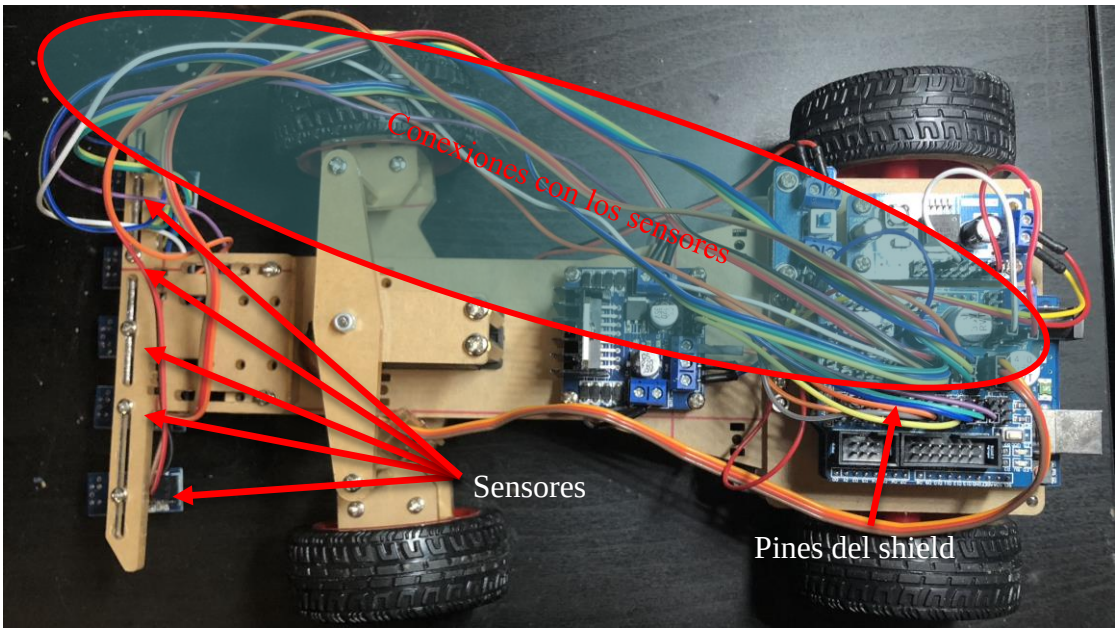


Ilustración 9 Conexiones entre los sensores y el shield

Por lo tanto, las salidas analógicas de cada sensor se han conectado con las entradas analógicas del shield A0, A1, A2, A3 y A4, y los pines de 5V y GND de cada sensor con su correspondiente pin de 5V y GND, asociado a la entrada analógica del shield, ya que cada entrada analógica tiene además su propio pin de 5V y GND. Se han asignado las entradas analógicas por orden de sensor más exterior a más interior,

empezando por la derecha, para que a la hora de desarrollar el sistema de guiado, sea más intuitivo dar valor a las variables, y que sea más sencillo analizar problemas, en caso de que los hubiera.

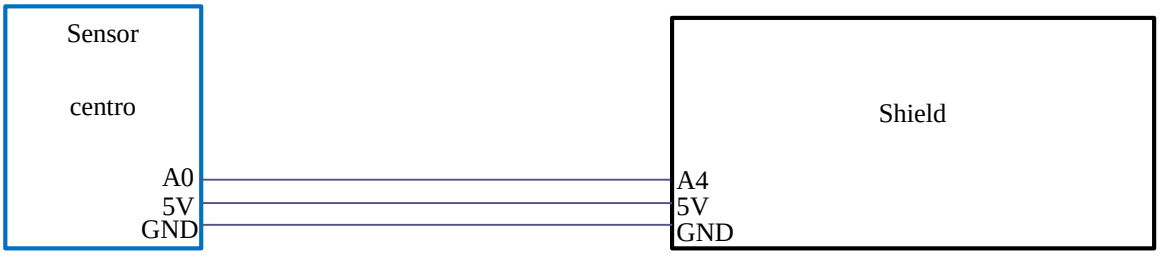


Ilustración 10 Ejemplo de conexión completa entre el shield y el sensor central

Después se han diseñado las conexiones entre el shield, el controlador de motores, y los motores. En este caso se ha puesto el controlador de motores entre el shield y los motores.

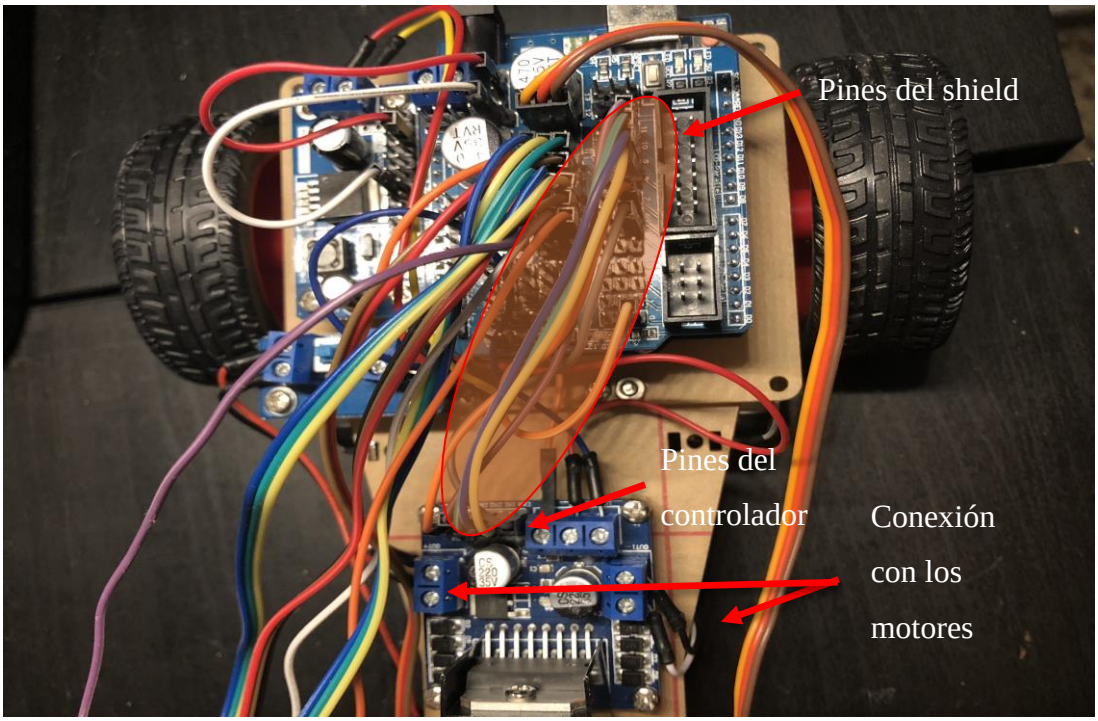


Ilustración 11 Conexiones entre el shield y el módulo controlador de motores

Por lo tanto, las conexiones serían del shield al controlador de motores, y de éste a los motores. Para el controlador de motores se han reservado los pines del shield 5V, 5, 6, 8, 9, 10 y 11, de los cuales el pin de 5V del shield se conectará con el pin de 5V del controlador, y después los pines 5 y 6 se conectan con los pines de habilita A (ENA) y

habilita B (ENB) respectivamente, y que sirven para controlar la velocidad. A continuación, los pines 8, 9, 10 y 11 del shield se ha decidido conectarlos con las entradas 1, 2, 3 y 4 del controlador de motores (INT1, INT2, INT3 e INT4) respectivamente. Finalmente, las salidas del controlador de motores se conectarán directamente a los motores. Específicamente, se ha decidido conectar las salidas 1 y 2 (OUT1 y OUT2) al motor izquierdo, y de la misma manera las salidas 3 y 4 (OUT3 y OUT4). De esta forma cada vez que en el pin OUT1 se tenga un valor de alto nivel, y en el OUT2 uno de bajo nivel, el motor izquierdo va a girar en sentido antihorario. Si estos valores se invierten, el sentido de giro también lo hace. Por lo tanto, poniendo valores de alto o bajo nivel en estos cuatro pines, se provoca el cambio en el sentido de la marcha del coche. Estos pines se han conectado directamente con el motor.

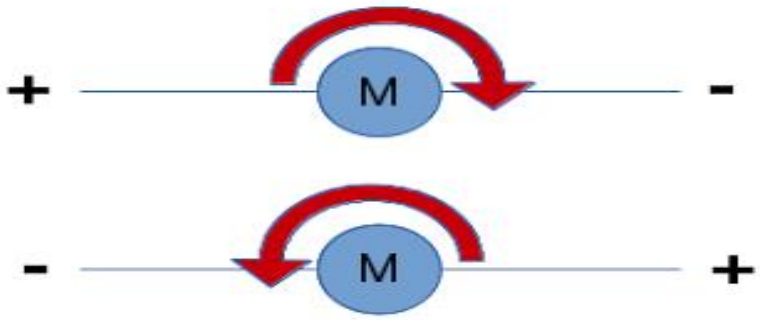


Ilustración 12 Giro de los motores

En resumen, todo este circuito de conexiones se hace para controlar fácilmente la velocidad de rotación de los motores, escribiendo en los pines 5 y 6 las señales analógicas que va a recibir el controlador de motores, y este controlador, según los valores analógicos que se hayan recibido, establecerá la velocidad deseada. El sentido de giro de los motores se controla mediante los pines del shield 8, 9, 10 y 11. En concreto, con los pines 8 y 9 se controla el motor izquierdo y con los pines 10 y 11 se controla el motor derecho.

Finalmente, se va a explicar el diseño de las conexiones entre el servo y la placa SunFounder, que se utiliza para poder controlar la dirección del robot, en función de los diferentes factores que se tienen en cuenta para diseñar el sistema de guiado. La conexión de este componente es sencilla. Simplemente se han conectado los tres pines que posee el servo con el conjunto de pines asociados al pin número 2 del shield, de tal manera que el pin 2 se ha conectado con la entrada analógica del servo, mientras que los pines de 5V y GND se han conectado con los respectivos pines de 5V y GND en el servo.

Software

Habitualmente el código Arduino tiene dos funciones, la función de bucle loop() y la función de inicialización setup(). La funcionalidad de la primera es un bucle infinito, mientras el robot esté conectado, mientras que la función setup() contiene todo el código que se va ejecutar antes que cualquier otra función al encenderse el robot. Estas dos funciones se han diseñado como dos bloques diferentes dentro de la arquitectura. Sin embargo, para el funcionamiento correcto del bucle, la función de inicialización tiene que preparar todas las variables, componentes software y constantes, con el fin de evitar conflictos cuando se ejecute el bucle y los demás componentes software.

La inicialización de variables y constantes ha de tener coherencia con el diseño de las conexiones de los componentes hardware del sistema de guiado, ya que, de otra forma, el sistema de guiado no hubiese tenido el resultado deseado. Además, en esta preparación se tiene en cuenta la dirección, el sentido y la velocidad que va a tener el robot al principio. Esta parte de preparación es importante, ya que aquí se establecen los nexos entre los componentes hardware y software.

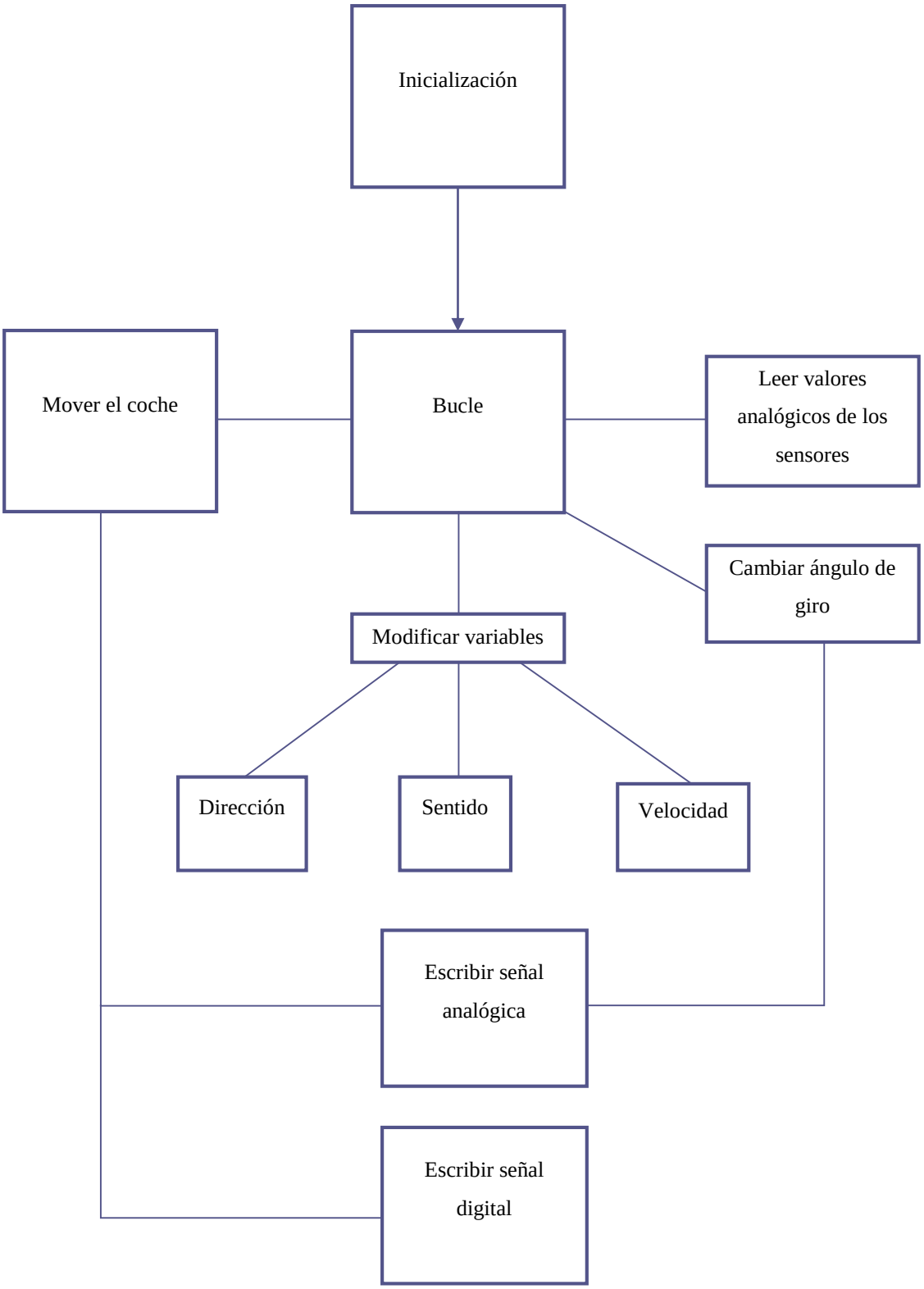


Ilustración 13 Arquitectura software

Como se puede observar en la ilustración 13, la arquitectura del software del sistema de guiado es bastante sencilla. En concreto, la parte que tiene más funciones es la del bucle, ya que en este bucle se realizan todas las acciones del sistema de guiado. En el siguiente capítulo, se profundiza más en el nivel de detalle de los diferentes componentes software.

El bucle tiene cuatro componentes principales agrupados por su funcionalidad, que son: mover el coche, leer los valores analógicos de los sensores, cambiar el ángulo de giro y modificar las variables. A su vez, estas funcionalidades, están compuestas por otro tipo de acciones. De esta manera, mover el coche está formado por escribir señal analógica y escribir señal digital, ya que el controlador de motores acepta dos tipos de señales: la analógica para la velocidad de giro del motor, y la digital para saber el sentido en el que se va a mover. Además, toda la lógica referente a cómo se va a mover el coche está contemplada dentro de este componente

El segundo elemento que se ha diseñado es el de leer las señales analógicas, el cual va a leer constantemente los valores de las señales analógicas que tendrán en cada momento cada uno de los sensores del robot. En función de estos valores, el comportamiento del resto de los componentes del sistema de guiado varía.

A continuación, se utiliza el componente de cambiar el ángulo de giro del servo para poder cambiar dirección del robot. Para ello se necesitaría escribir un valor analógico directamente en el servo. Es por ello que este elemento tiene el componente de escritura analógica.

Finalmente, para escribir señal analógica o digital, se hacen a través de variables, las cuales cambian en función de lo que se lea de los sensores, En este caso este elemento está compuesto de las variables que se cambian: velocidad, dirección y sentido.

Arquitectura hardware-software

Para terminar este capítulo, se va a explicar la arquitectura de las interacciones entre hardware y software. Habiendo detallado la arquitectura de estas dos partes previamente, se ha diseñado a continuación la arquitectura del nexo entre estos dos elementos. Como se puede apreciar en la ilustración 13, podemos observar que la unión de estas dos partes viene dada por la placa SunFounder compatible con Arduino. Es en

esta placa en la que cargamos el software del sistema de guiado y, por lo tanto, la que va a controlar todos los elementos hardware según el funcionamiento de los componentes software. Sin embargo, para que la arquitectura sea más clara, se han diseñado las relaciones e interacciones de forma directa, es decir, se ha omitido el papel del shield o placa SunFounder en la arquitectura conjunta, tal y como se puede ver en la ilustración 13.

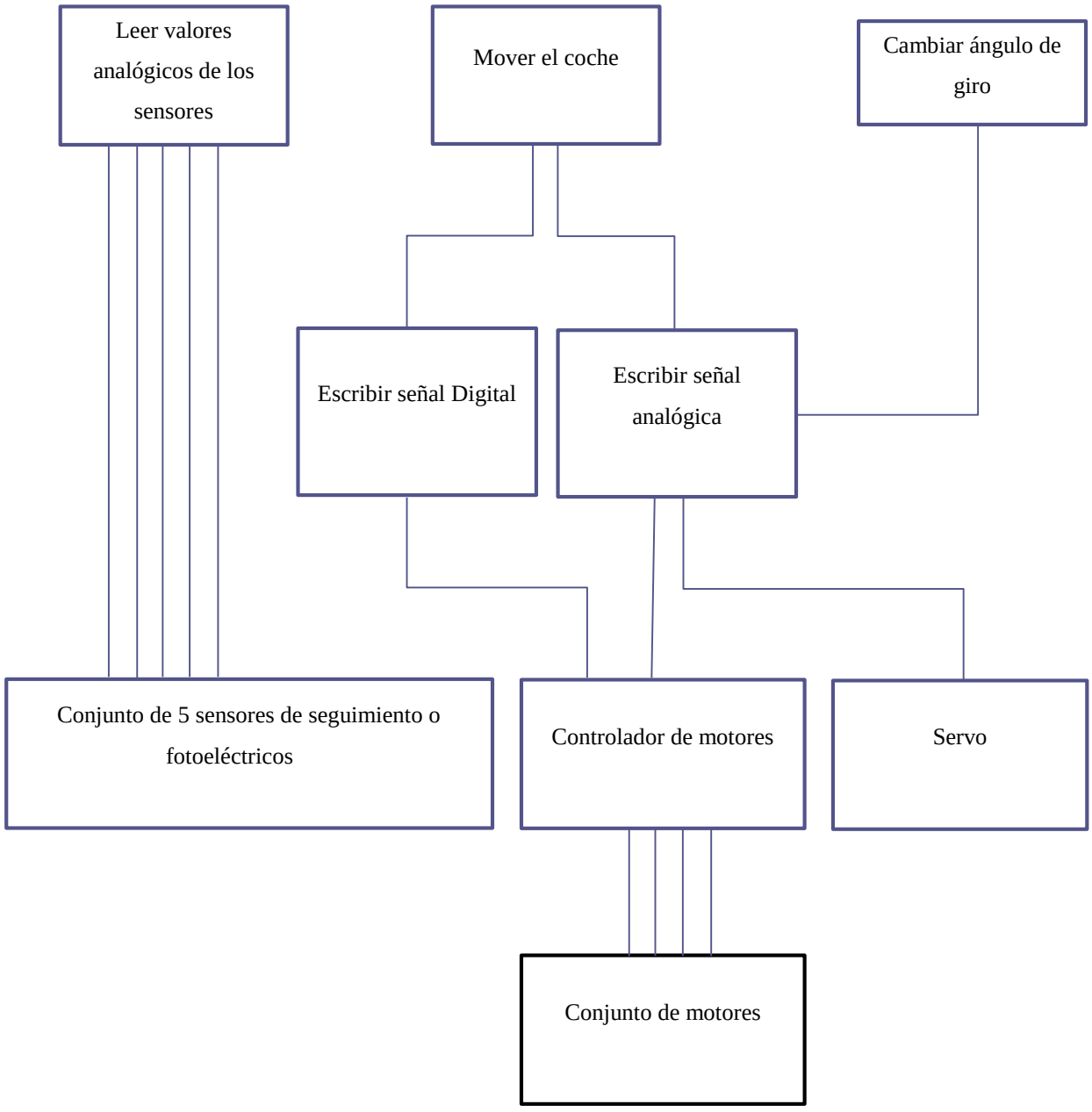


Ilustración 14 Arquitectura de las interacciones hardware y software

Como se puede ver en la ilustración anterior, las interacciones entre los componentes hardware y software vienen dadas cuando se producen escrituras tanto de señales analógicas como de señales digitales, así como también lecturas en el hardware del robot, aunque para el sistema de guiado solamente se necesitan hacer lecturas constantemente de las señales analógicas de todos los sensores.

Además, se puede observar que se realizan tanto escrituras de señales analógicas como de señales digitales en el controlador de motores. En este controlador es posible escribir estos dos tipos de señales en pines diferentes, para poder controlar dos funciones diferentes de los motores. En resumen, la señal analógica serviría para controlar la velocidad a la cual, rota el motor, y en el caso de las señales digitales, servirían para controlar el sentido en el que rota el motor.

Finalmente, el servo se controla escribiendo un valor analógico en éste mediante el componente de cambiar el ángulo de giro. Este nuevo valor analógico será el ángulo con el que va a girar el servo. Cada componente software específico contiene la lógica que controla lo que se escribe en cada componente hardware con el que están relacionados, o con los que interactúan.

4.2. Diseño en detalle

En este apartado se entra en más detalle sobre las decisiones y el diseño que se ha llevado a cabo, tanto de la parte software, como del hardware. Además, se detallará todo lo expuesto anteriormente destinado al diseño del sistema de guiado. Es decir, a partir de ahora se pierde un nivel de abstracción que se venía empleando hasta ahora en los anteriores capítulos del diseño del sistema de guiado. Por lo tanto, este capítulo no solo incluye la explicación más detallada de los componentes, relaciones, funciones, interacciones, sistemas o partes de forma estática, sino que además se profundizará sobre el funcionamiento que se espera durante la ejecución del sistema de guiado.

El funcionamiento de software, en tiempo de ejecución, viene dado por todas las variables que se corresponden con diversos aspectos físicos del hardware, como la

velocidad, dirección o sentido de la marcha. Sin embargo, todo el funcionamiento viene condicionado por los valores que tienen los 5 sensores que tiene el robot en el frontal. Como hemos explicado anteriormente, según cuales sean estos valores, la lógica cambia el comportamiento del resto de componentes hardware.

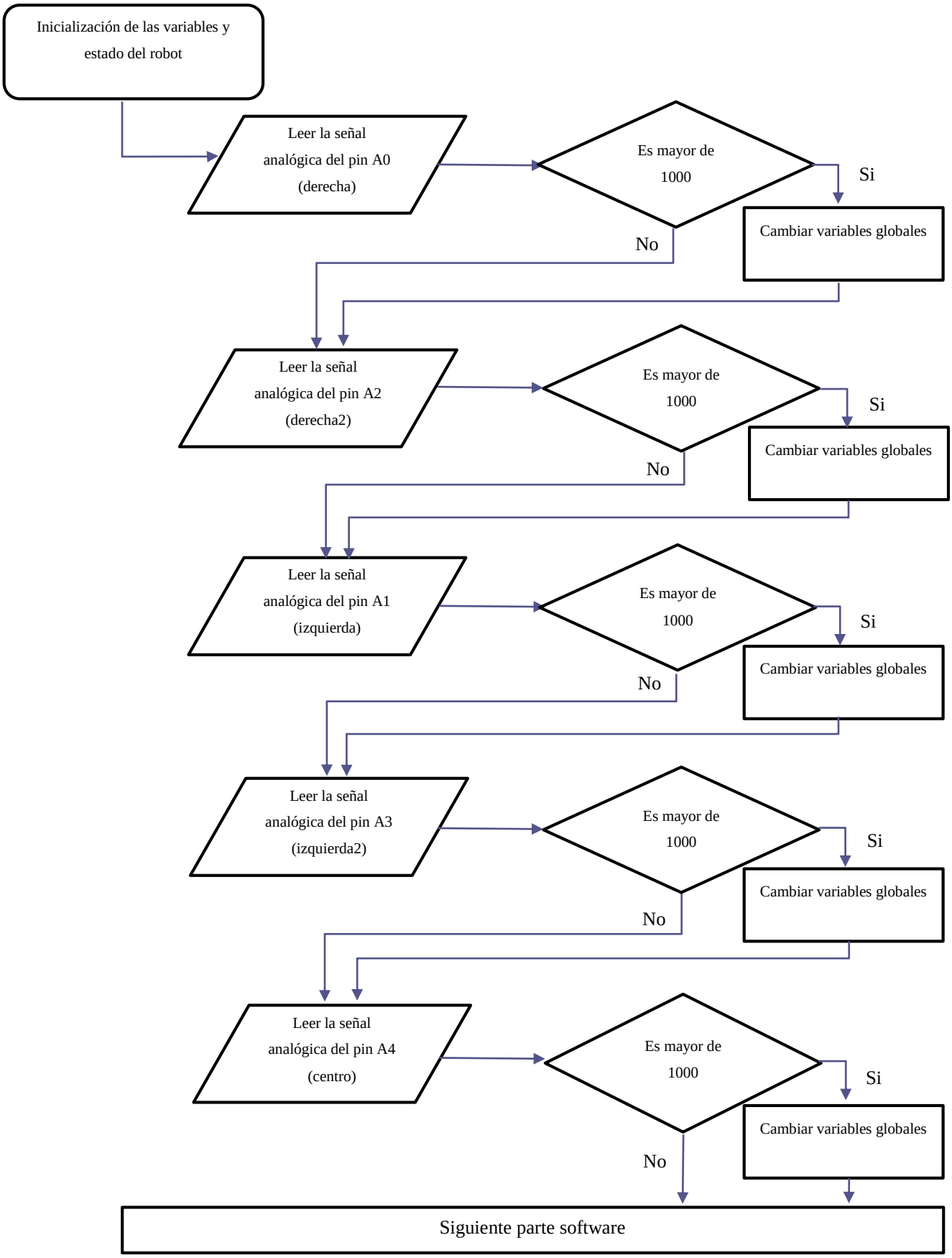


Ilustración 15 Primera parte del diagrama de flujo software.

En la ilustración 15 se puede observar la primera parte del diagrama de flujo, que se corresponde con las lecturas de las señales analógicas de todos los sensores y el cambio en el valor de las variables en función de los valores que se hayan obtenido previamente. En esta lectura de los cinco sensores que componen el sistema de guiado, el orden en el que se realizan las lecturas es importante para el correcto funcionamiento del sistema de guiado. Primero se hacen las lecturas sobre los dos sensores exteriores de la derecha, y después sobre los dos exteriores de la izquierda. Finalmente, se hace sobre el sensor central. Esto es debido a que el cambio de las variables se realiza justo a continuación de la lectura de la señal analógica de cada sensor, si fuese necesario. Estas variables se corresponden a la velocidad, tanto del motor izquierdo como del motor derecho, y la dirección o ángulo de giro que tendría el robot en un momento determinado.

La condición de que el valor leído del sensor tiene que ser mayor que mil se debe a que el sensor emite en su salida un valor mayor que mil cada vez que detecta una superficie de color negro. Por lo tanto, si este valor es mayor de mil para un sensor determinado, se cambian las variables para cambiar la dirección, velocidad o sentido, si fuese necesario, para seguir la línea, que es de color blanco. Si no es mayor que mil significa que el sensor no ha detectado ninguna superficie de color negro, y es por ello por lo que no se necesitan cambiar las variables para ajustar la dirección o velocidad. Sin realizar ningún cambio, se pasa a la siguiente comprobación, y si fuese necesario, al siguiente cambio de variables.

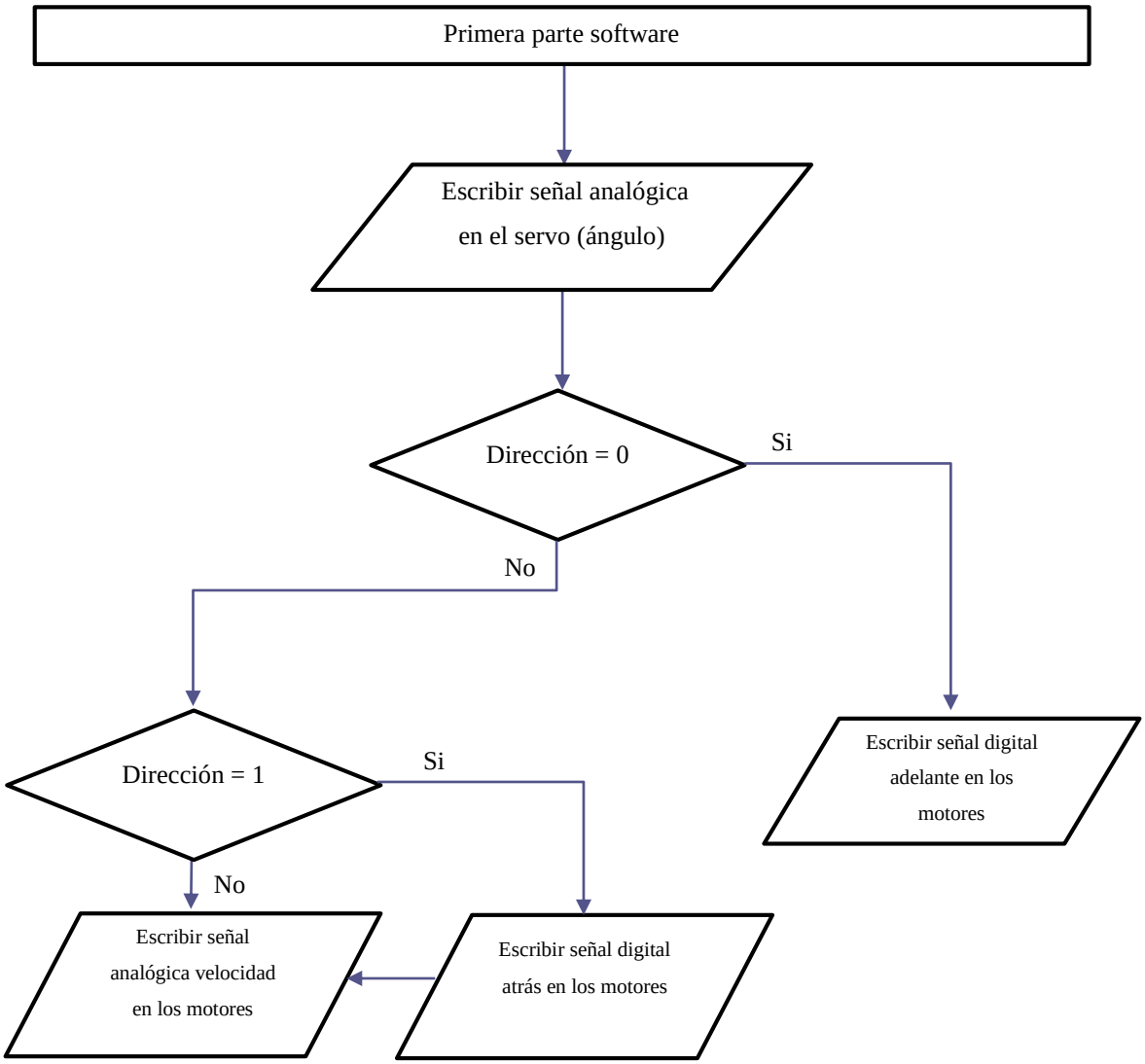


Ilustración 16 Ultima parte del diagrama de flujo del software

Una vez establecidas las variables, se procede a cambiar el comportamiento del robot según sus valores. En primer lugar, se escribe el valor del ángulo en el servo para cambiar la dirección. Después de establecer la dirección se cambia tanto la velocidad como el sentido de giro de los motores mediante la función *moverCoche()*. Esta función recibe las variables sentido (0 hacia delante, 1 hacia atrás) y velocidad. Finalmente, después de evaluar las condiciones asociadas al sentido en el que tiene que circular el coche, se hace una escritura de un valor analógico en los dos motores referente a la velocidad, que se representa mediante la variable velocidad.

Después de explicar el flujo de las acciones que se realizan en el software, y como afectan éstas al hardware, se especifican los valores que van a adoptar las diferentes variables que se utilizan para cambiar el funcionamiento del coche bajo las circunstancias que se han detallado en los diagramas de flujo. Para comenzar, las variables tienen una serie de valores límites, es decir unos mínimos y máximos, y por lo tanto se tiene que trabajar entre estos dos valores en cada variable para el que funcionamiento del coche sea el adecuado. Así se han establecido un mínimo de 95, y un máximo de 140 para la velocidad, ya que por debajo de 95 el coche no podría avanzar debido a su propio peso, y por encima de 140 el coche circularía tan rápido que en la mayoría de ocasiones, al trazar las curvas, acabaría saliéndose del circuito. De la misma manera se han establecido límites para el ángulo de giro del servo, los cuales son 95 y 155. Si el ángulo que se escribe es menor que 95, al girar a la izquierda en las curvas más cerradas las ruedas delanteras del coche acabarían tocando el chasis del coche, y si se superase el valor de 155 ocurriría lo mismo pero esta vez girando hacia la derecha, con un ángulo demasiado cerrado que provocaría que las ruedas tocasen el chasis del coche. Que las ruedas toquen el chasis provoca un efecto de frenado en el coche, lo cual no es deseado, ya que éste puede llegar a detenerse en las curvas más cerradas. El diseño del cambio de valores se ha hecho dentro de estos límites para el correcto funcionamiento del coche.

Finalmente, en el diseño del sistema de guiado se ha tenido en cuenta qué es lo que tiene que hacer el hardware en diversas circunstancias, tal y como se explica en la siguiente tabla:

Circunstancia	Como detectarla	Acción requerida
El coche circula sobre una recta.	Si el sensor central está constantemente enviando señales mayores que 1000	Poner la dirección recta del servo y la velocidad del motor a la máxima establecida por diseño.
El coche circula sobre una curva abierta hacia la derecha	Si el sensor de la derecha 2 (interior) detecta que se está circulando sobre la línea.	Poner el ángulo de la dirección del servo en un punto intermedio entre el ángulo recto y el máximo hacia la derecha y, además establecer la velocidad del motor a la velocidad máxima.

Tabla 1 Comportamiento del hardware, según las circunstancias de circulación

El coche circula sobre una curva abierta hacia la izquierda	Si el sensor de la izquierda 2 (interior) detecta que se está circulando sobre la línea.	Poner el ángulo de la dirección del servo en un punto intermedio entre el ángulo recto y el máximo hacia la izquierda y, además establecer la velocidad del motor a la velocidad máxima.
El coche circula sobre una curva cerrada hacia la derecha.	Si el sensor de la derecha (exterior) detecta que se está circulando sobre la línea.	Poner en el ángulo de la dirección del servo el valor máximo hacia la derecha permitido por diseño y, además, establecer la velocidad del motor a la velocidad mínima.
El coche circula sobre una curva cerrada hacia la izquierda.	Si el sensor de la izquierda (exterior) detecta que se está circulando sobre la línea.	Poner en el ángulo de la dirección del servo, el valor máximo hacia la izquierda permitido por diseño y, además, establecer la velocidad del motor a la velocidad mínima.

A partir del comportamiento del hardware definido en la tabla 1 se han diseñado los diferentes componentes software, tal y como se ha visto en los diagramas de flujo.

Además, se han realizado diferentes comprobaciones a nivel de software para detectar las circunstancias en las que se encuentra el coche, y su correcto funcionamiento. El cambio de valor de las variables y la escritura de estos valores en el hardware del robot se corresponden con la acción requerida para que el sistema de guiado funcione. En conclusión, se ha intentado que el diseño en detalle del comportamiento tanto hardware como software, a pesar de haberlos dividido en dos partes diferenciadas, sea coherente entre las dos partes, para conseguir el funcionamiento deseado del sistema de guiado.

4.3. Tecnologías y componentes

En este proyecto hemos hecho uso fundamentalmente de Arduino, tanto por la parte software, como por la parte hardware. Por esto es importante conocer en profundidad qué es Arduino, cuáles son sus componentes, y su papel en el proyecto que hemos llevado a cabo. Además, en el proyecto se han usado varias tecnologías hardware compatibles con la placa Arduino.

Arduino

Arduino es una plataforma Open Source tanto hardware como software, que en sus inicios fue ideada para ámbitos educativos, pero que ha ido evolucionando hacia proyectos más complejos. Principalmente se conoce a Arduino por sus placas hardware, que se comercializan a todo tipo de público. La más conocida es la Arduino UNO, que es la placa que hemos utilizado para el desarrollo del sistema de guiado. Arduino presenta un entorno de desarrollo, un lenguaje de programación, y un microcontrolador. Sin embargo, la evolución de Arduino ha permitido crear una herramienta económica y sencilla de utilizar, asequible tanto para estudiantes como para profesionales. Ofreciendo estas características, Arduino ha ido creciendo a lo largo de los años, lo cual significó un incremento en la gama de placas disponibles, así como de Shields, o módulos de expansión para la placa Arduino. También se abrió un abanico de periféricos y componentes hardware que podrían ser utilizados con Arduino.

Además del entorno de desarrollo software de Arduino, en el cual podemos programar el código del sistema de guiado, también se ha utilizado la placa hardware Arduino Uno. Sin embargo, los módulos que se han utilizado para desarrollar este sistema no son Arduino, sino que han sido desarrollados por la marca SunFounder. Todos estos componentes hardware se explican a continuación.

Sensor de seguimiento o fotoeléctrico

El sistema de guiado del robot ha utilizado este sensor de seguimiento (o sensor fotoeléctrico de infrarrojos). Básicamente, lo que este sensor detecta son los colores de la pista que están más próximas al sensor. Para ello, este sensor constantemente lanza luz infrarroja mediante su tubo transmisor de infrarrojos, y desde el tubo negro receptor, se absorbe la luz. Cuando el tubo transmisor lanza luz infrarroja sobre una superficie de color negra, la luz infrarroja reflejada es menor, y por lo tanto la luz infrarroja recibida por los tubos receptores es menor. Esto provoca que la salida sea a nivel alto, y el LED que tiene este sensor se apague. Cuando la superficie es de color blanco, funciona de la

misma forma, pero al revés, es decir, una vez que se recibe la luz, ésta es mayor. Por lo tanto, la salida será a nivel bajo, y el LED se encenderá.

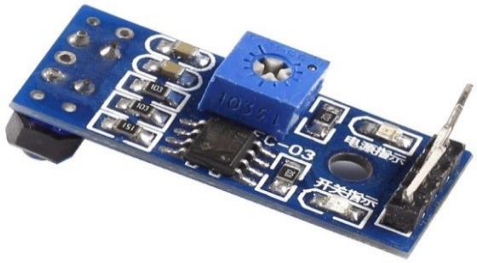


Ilustración 17 Sensor fotoeléctrico

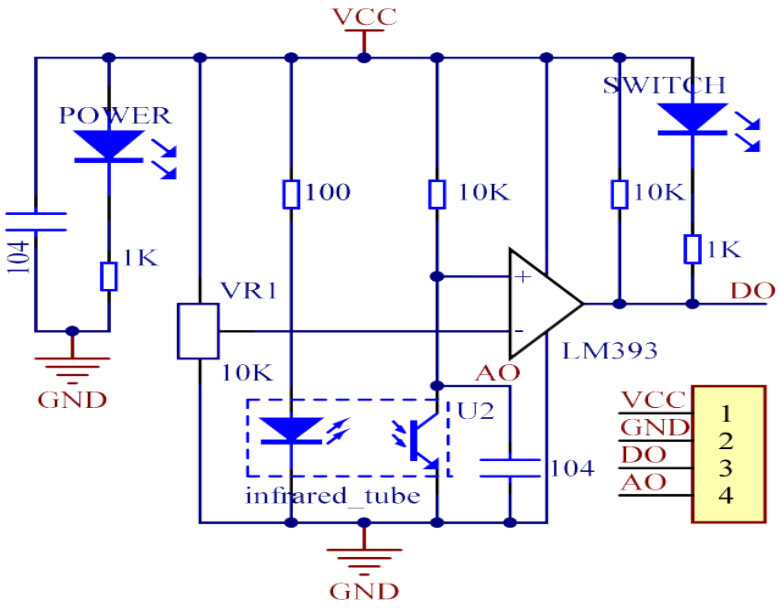


Ilustración 18 Circuito interno del sensor de seguimiento

Se puede observar en el circuito el funcionamiento de los tubos de recepción y emisión, y cómo en función de la comparación de la luz que se emite y recibe, los pines

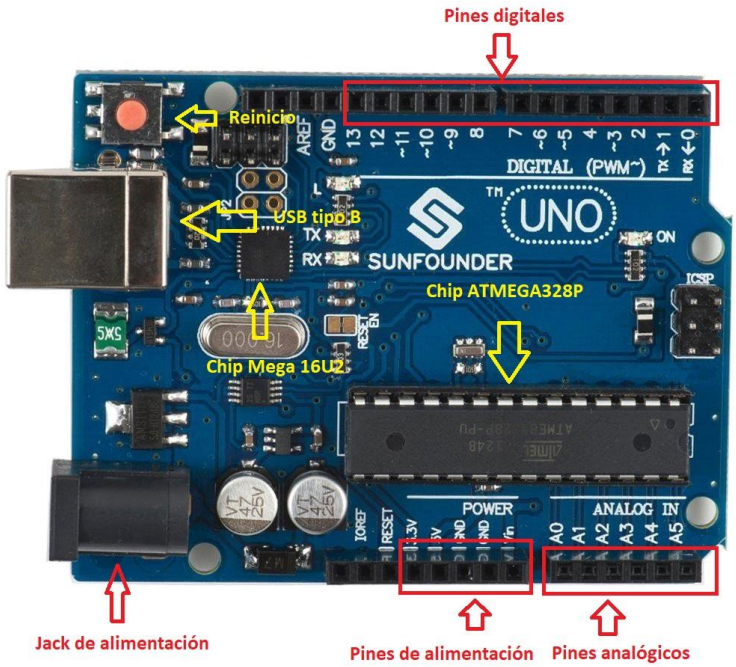
A0 y D0 van a enviar una señal de bajo nivel o de alto nivel en función de los resultados de estas comparaciones. En la siguiente tabla se explican las funciones de los pines y/o componentes del sensor de seguimiento fotoeléctrico.

Pin	Nombre	Descripción
1	VCC	Fuente de alimentación
2	GND	Masa
3	D0	Salida de señal de conmutación TTL
4	A0	Salida de señal analógica

Tabla 2 Pines del sensor de seguimiento

Placa SunFounder UNO R3

Realmente esta no es una placa de la propia Arduino, sino que se trata de la SunFounder Uno R3, la cual es una placa completamente compatible con la placa Arduino UNO. Existen muchas placas compatibles con Arduino en el mercado, ya que al ser Open Source, permite a la gente diseñar sus propias placas basadas en Arduino para diferentes propósitos, incluso comerciales. En la siguiente ilustración se pueden ver las conexiones que tiene



Shield

Habitualmente, los diferentes chips que forman un circuito electrónico se colocan sobre una breadboard o placa de pruebas, ya que así se puede tener una base electrónica para hacer varios tipos de circuitos. Este shield realiza una función similar. Por lo tanto, las interconexiones entre motores, sensores o el servo, y la placa Arduino se harían sobre esta placa, la cual estará conectada directamente con la placa Arduino, y ésta haría de interfaz entre los componentes y la placa Arduino. De esta forma se simplifican las interconexiones, ya que solo hay que conectar los componentes con el shield, permitiendo desarrollar el proyecto rápidamente. En la parte del código se acceden a las conexiones de este shield fácilmente, pudiendo programar los componentes que se han conectado en este shield de forma sencilla. Este periférico de Arduino es uno de los más utilizados habitualmente.

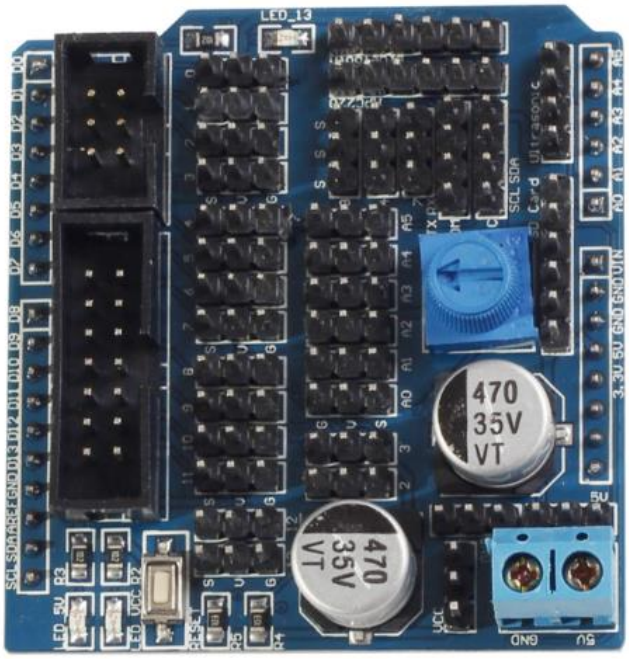


Ilustración 19 Shield

En concreto, este shield se conecta directamente con la placa Arduino, apilando el shield encima de la placa SunFounder, formando así un stack o pila entre la placa controladora y el shield. En la siguiente ilustración se puede observar todas las conexiones

que ofrece el shield que usaremos de interfaz entre los componentes del robot y la placa Arduino donde cargaremos el código del sistema de guiado.

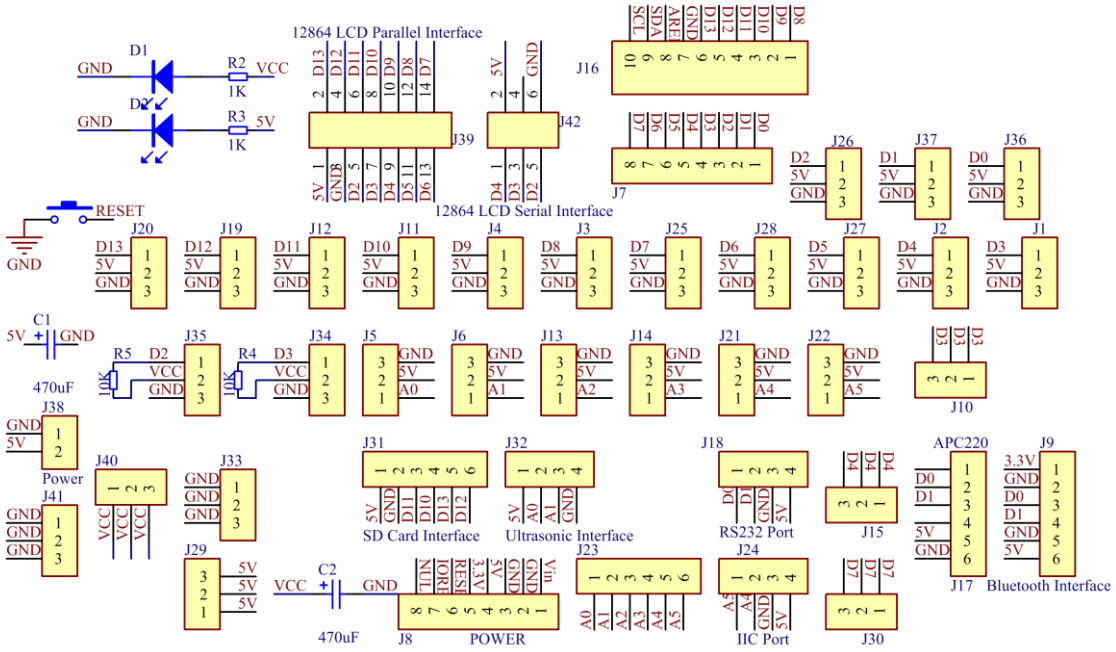


Ilustración 20 Circuito interno del shield

Controlador de motores

Este módulo nos sirve de interfaz entre los motores y la placa Arduino, facilitando el desarrollo del software utilizado para controlar los motores en función de lo que se desea construir finalmente.

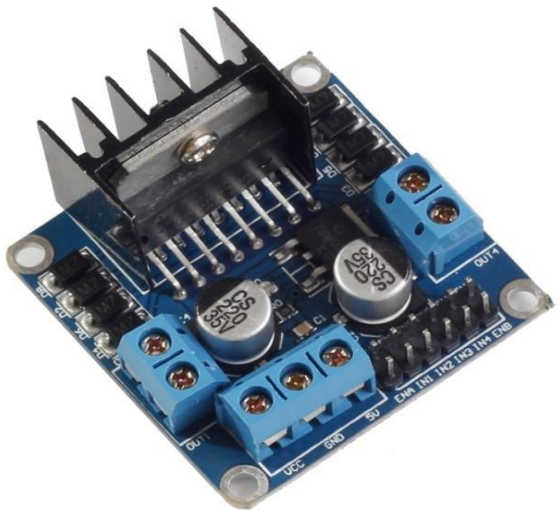


Ilustración 21 Controlador de motores

En concreto, este módulo se utiliza para controlar la rotación de los motores del robot. Sus principales características son las siguientes:

- Puede utilizarse con altos voltajes (Hasta 40V).
- Alta corriente de salida, con picos de hasta 3A.
- Tiene una potencia de 25W.
- Dos puentes H incorporados, alto voltaje, alta corriente, controlador completo del puente, que puede ser usado para controlar motores DC, relés, y otras cargas inductivas.
- Usa señal de nivel lógico estándar para control.
- Capaz de controlar un motor de dos o cuatro fases, y motores DC de dos fases

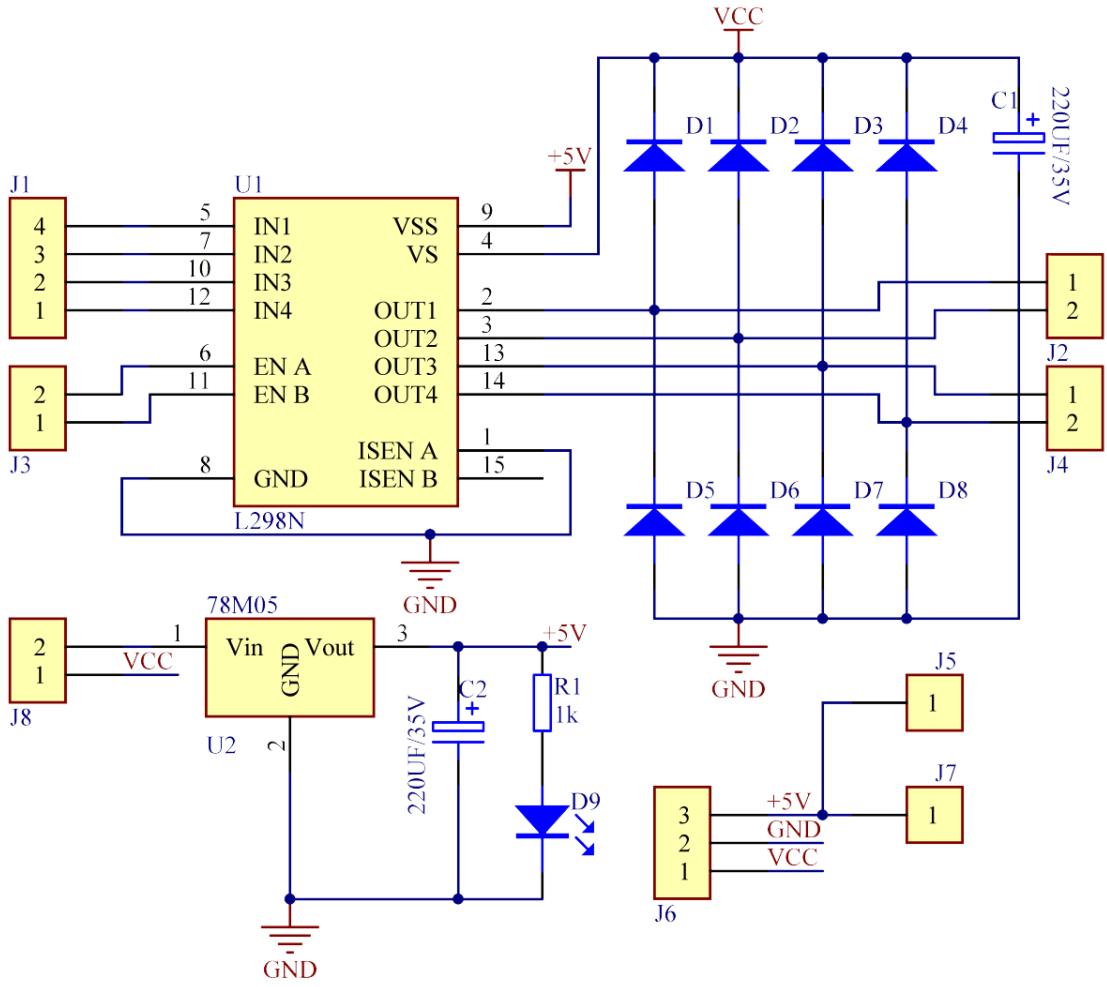


Ilustración 22 Circuito interno del módulo controlador de los motores

Las funciones de los diferentes pines se muestran en la siguiente tabla:

Pin	Nombre	Descripción
1 15	Sensor A Sensor B	El sensor resistor se conecta entre este pin y masa, para controlar la corriente de la carga.
2 3	Salida 1 (OUT1) Salida 2 (OUT2)	Salidas del puente A. La corriente que fluye por la carga conectada entre estos dos pines se monitoriza en el pin 1
4	Vs	Voltaje suministrado para las etapas de salida de potencia. Se debe conectar a un condensador no inductivo de 100nF entre este pin y masa.
5 7	Entrada 1 (INT1) Entrada 2 (INT2)	Entradas compatibles con TTL del puente A.
6 11	Habilita A Habilita B	Entrada habilitada compatible con TTL. El estado L deshabilita el puente A (Habilita A) y/o el puente B (Habilita B)
8	GND	Masa
9	Vss	Fuente de voltaje para los bloques lógicos. Se debe conectar a un condensador no inductivo de 100nF entre este pin y masa
10 12	Entrada 3 (INT3) Entrada 4 (INT4)	Entradas compatibles TTL del puente B.
13 14	Salida 3 (OUT3) Salida 4 (OUT4)	Salidas del puente B. La corriente que fluye a través de la carga conectada entre estos dos pines se controla en el pin 15

Tabla 3 Pines del módulo controlador de motores

En la tabla 4 se explica el comportamiento del motor DC según los valores para los pines ENA (Habilita A) y los pines IN1 e IN2 (Entradas 1 y 2). Esta tabla es bastante interesante, ya que muestra el funcionamiento del motor con los valores que se introducen, ayudando bastante a la hora de programar el comportamiento de los motores del robot, según las circunstancias en las que se encuentre en cada momento.

ENA	IN1	IN2	El estado del motor A DC
0	X	X	Parar
1	0	0	Frenar
1	0	1	Rotar en sentido horario
1	1	0	Rotar en sentido antihorario
1	1	1	Frenar

Tabla 4 Funcionamiento del módulo controlador de motores

Conversor de bajo DC-DC

Este módulo se utiliza para reducir el voltaje de entrada, y proporciona un voltaje de salida de 5V. Es capaz de controlar una corriente de 3A con una buena regulación de la carga. Es un diseño que prioriza la simplicidad en el diseño de una fuente de alimentación para los componentes del robot.

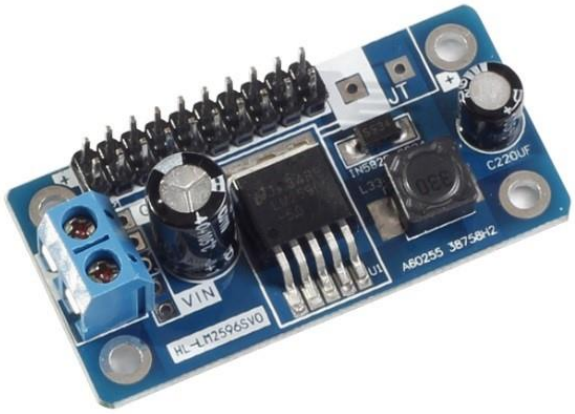


Ilustración 23 Módulo conversor de bajo DC-DC

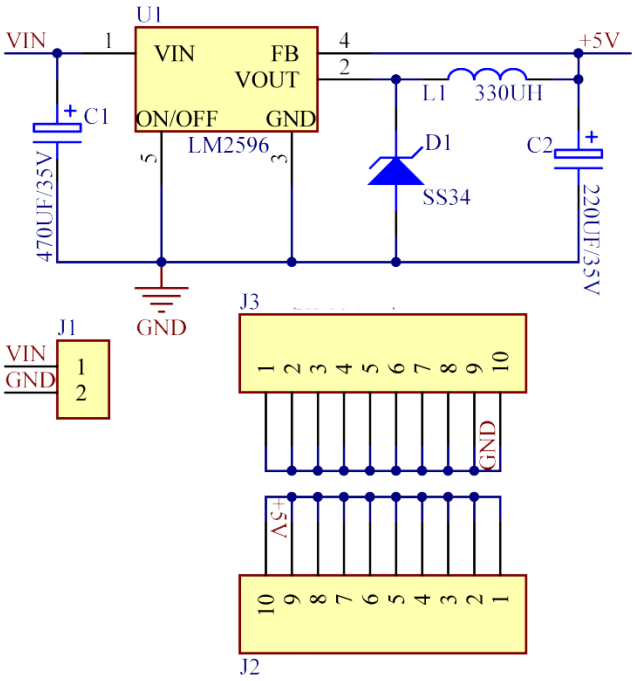


Ilustración 24 Circuito interno del módulo de conversión bajo DC-DC

Sus principales características son las siguientes:

- Voltaje de salida estable a 5V
- Corriente de salida a 3ª
- Amplio rango de voltaje de entrada, hasta 40V
- Frecuencia del oscilador interno a 150 kHz
- Capacidad de TTL cerrado
- Modo ‘ahorro de energía’, normalmente 80 μA
- Apagado térmico y límite de corriente

A continuación, se describen las funciones de los pines del conversor:

Pin	Nombre	Descripción
1	VIN	Esta es la fuente de entrada positiva para el regulador, que sirve para reducir el voltaje y suministrar corrientes de conmutación
2	GND	Masa
3	SALIDA	Interruptor interno
4	Realimentación	Detecta el voltaje de salida para completar el circuito de realimentación
5	Encendido/apagado	Permite apagar el circuito regulador, utilizando señales de nivel lógico, reduciendo la corriente de entrada. Bajando el voltaje de entrada de este pin por debajo de 1.3V se enciende el regulador, y subiendo el voltaje por encima de 1.3V se apaga el regulador. Si no se necesita esta función se puede conectar este pin a masa.

Motor

Ilustración 25 Motor

Se trata de un motor eléctrico, el cual se conecta con dos cables al controlador de motores, estos dos cables según el voltaje que tenga cada uno, teniendo bajo nivel uno y alto nivel otro, va a girar en un sentido determinado, y si estos voltajes se invierten, el sentido del giro también.

Servo

Ilustración 26 Micro servo

El servo es un conjunto de sistemas de control compuesto de motores DC, un conjunto de engranajes de reducción, sensores y circuitos de control. El eje de salida del servo solo puede rotar dentro en un cierto rango. En este caso se ha utilizado para controlar la dirección del coche.

Interruptor

Ilustración 27 Interruptor

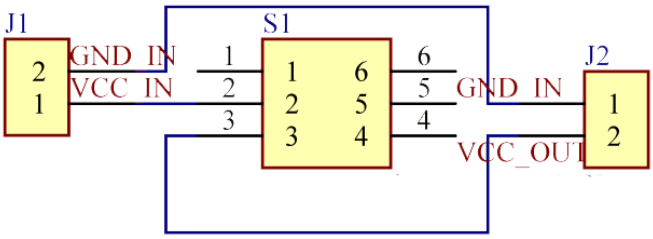


Ilustración 28 Circuito interno del interruptor

Nuestro coche también tiene un interruptor para poder encenderlo y apagarlo. Sin embargo, tiene que conectarse correctamente a la placa de conversión de corriente.

5. Desarrollo de la solución

Una vez diseñada la solución, y especificando cómo va a ser el comportamiento del coche, se ha procedido a la implementación del sistema de guiado para el robot. Para empezar, se deben conectar todos los pines tal y como se había especificado en la arquitectura del hardware, para que luego a la hora de desarrollar el software del sistema fuese más sencillo. Las conexiones entre módulos y placas se han realizado mediante cables de diferentes colores, que facilitan las conexiones entre los diferentes componentes que tiene el robot.

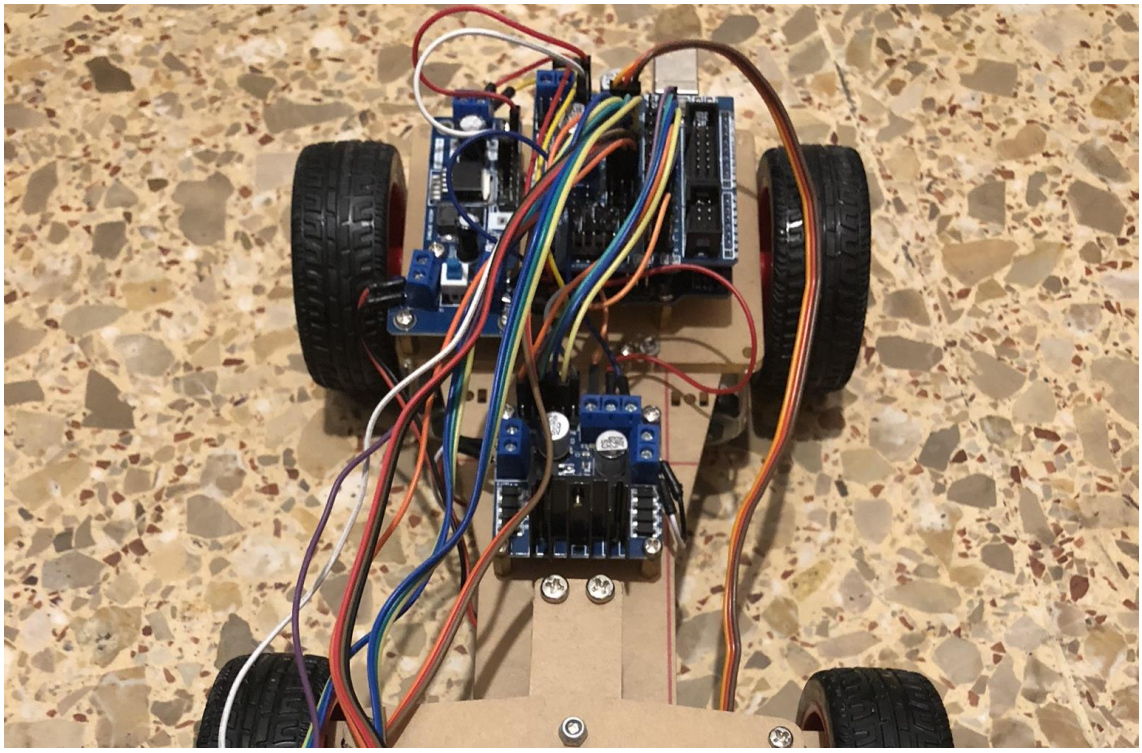


Ilustración 29 Conexiones del shield

En la ilustración 29 se pueden ver las conexiones de los componentes con el shield. También se puede observar como el shield está apilado encima de la placa SunFounder, conectando los diferentes pines de la placa SunFounder con el shield.

También tenemos las conexiones entre el conversor, tanto de voltaje como de corriente. Estas conexiones se han realizado a los pines de alimentación del shield, que a su vez van a servir de alimentación para la placa SunFounder, el servo y los sensores de seguimiento. La fuente de alimentación en este caso proviene de dos pilas colocadas

debajo del coche. Estas dos pilas se conectan con un interruptor para controlar cuando se enciende o apaga el coche. Este interruptor se conecta con el conversor, el cual finalmente se conectará con el shield.

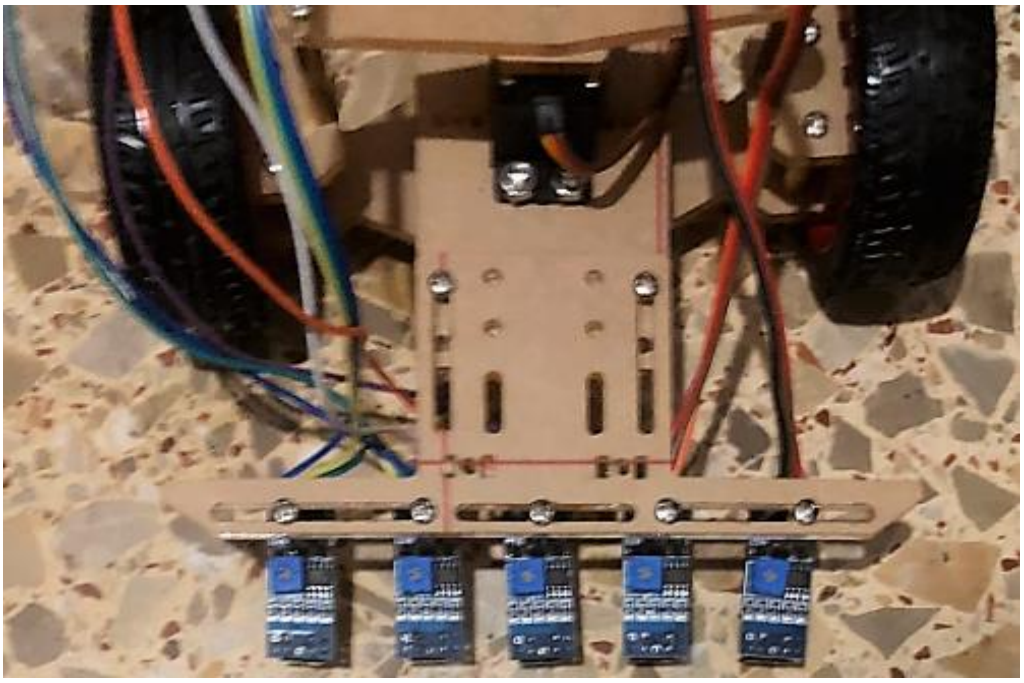


Ilustración 30 Parte frontal con los sensores

En la ilustración 30 se aprecia como los sensores se han colocado en la parte frontal del coche para poder detectar sobre qué superficie está circulando.

Una vez explicada cual ha sido la construcción del hardware, se va a pasar a explicar el desarrollo del sistema de guiado a nivel de software. En primer lugar, hay que saber que el sistema software está compuesto de dos partes. En la primera se inicializa el coche, estableciendo cual es el modo de los pines de velocidad del coche, y los pines del sentido en el que circula el coche. También hay que decir que en esta parte no se inicializan las variables, ya que esto se hace cuando se declaran. Esta función de inicialización se llama *setup()*, y se ejecuta siempre al principio, justo después de encender el coche. Después de esta función se ejecuta un bucle infinito mientras el coche esté encendido, y es aquí donde se ejecutará constantemente la lógica del sistema de guiado.

```
#include <Servo.h>

Servo servo;

#define ADELANTE 0
#define ATRAS    1
#define DERECHA  A0
#define IZQUIERDA  A1
#define DERECHA2  A2
#define IZQUIERDA2 A3
#define CENTRO A4
#define PIN_VELOCIDAD_A   5
#define PIN_VELOCIDAD_B   6
#define MOTOR_IZQ_1 8
#define MOTOR_IZQ_2 9
#define MOTOR_DER_1 10
#define MOTOR_DER_2 11

int velocidadD = 125;
int velocidadI = 125;
unsigned char angulo = 125;

void setup()
{
  Serial.begin(9600);
  pinMode(PIN_VELOCIDAD_A, OUTPUT);
  pinMode(PIN_VELOCIDAD_B, OUTPUT);
  pinMode(MOTOR_IZQ_1, OUTPUT);
  pinMode(MOTOR_IZQ_2, OUTPUT);
  pinMode(MOTOR_DER_1, OUTPUT);
  pinMode(MOTOR_DER_2, OUTPUT);
  servo.attach(2);
}
```

Ilustración 31 Inicialización del sistema de guiado

Para empezar, se necesita especificar los ángulos de giro que va a necesitar el coche para realizar los diferentes trazados que se le propone. El primer caso es circular sobre una línea recta. En este caso, el ángulo tiene que ser recto, lo que se corresponde con el valor 125. Además, al ir en línea recta, el coche puede tener la velocidad alta sin riesgo a que se salga del trazado.

A continuación, se ha tenido en cuenta el caso en el que la curva no sea cerrada. En este caso, el ángulo de giro no se necesita que sea muy cerrado, tanto para la izquierda, como para la derecha. Es por ello que se han decidido los valores de 135 cuando se tenga que girar a la derecha, y 115 cuando se haga hacia la izquierda. Además, con este ángulo si el coche circula a la máxima velocidad, no tiene riesgo de salirse de la pista.

El último caso que se puede dar es el de una curva cerrada. En este caso los ángulos de giro serán los máximos. En el caso de una curva hacia la derecha el ángulo

será de 155, y en el caso de una curva hacia la izquierda el ángulo será de 95. Además, el coche, al encontrarse ante este caso, reducirá la velocidad al mínimo especificado en el diseño, que será de 95, para facilitar el giro en este tipo de curvas sin salirse del trazado.

Ahora que ya se han especificado los valores que van a tener las variables referentes a la velocidad, y el ángulo de giro del robot, solamente faltaría especificar cuáles serían las condiciones a nivel de software en las que se procedería a hacer cualquier cambio sobre estos valores. Para esto se lee constantemente el valor analógico de los 5 sensores que tiene en el frontal el robot. Si este valor que se lee de los sensores es mayor que mil, significa que este sensor está detectando la línea negra del trazado. Por lo tanto, la condición para cambiar los valores de las variables viene dada por el valor de la señal analógica de cada sensor. Mediante una serie de condiciones, se comprueban los valores que se obtienen de los sensores. En concreto, el primer *if* comprueba si el sensor interior de la derecha devuelve un valor superior a mil. Si se da este caso, significa que el robot se encuentra en una curva abierta hacia la derecha, y por lo tanto se tienen que cambiar el ángulo de giro a 135 y la velocidad a 140, tal y como se ha explicado antes. Después de esto, se hayan cambiado las variables o no, se pasa a comprobar si el sensor más exterior de la derecha, llamado sensor derecha2 tiene un valor mayor que mil. Si esto es así, significa que el coche está circulando sobre una curva cerrada, y por esto la variable del ángulo se tiene que establecer a 155 y la velocidad se tiene que reducir a 125. Una vez comprobados estos dos sensores de la derecha, se hayan cambiado los valores de las variables o no, se ha desarrollado la lógica para comprobar los dos sensores de la izquierda, de la misma forma que se ha hecho con los sensores de la derecha, es decir, si el sensor interior de la izquierda, llamado sensor izquierda, devuelve una señal analógica mayor que mil, significa que el robot está sobre una curva abierta hacia la izquierda, y por lo tanto, la variable del ángulo se cambia a 115 y la variable de velocidad se cambia a 140. Después de esto, se haya ejecutado el código referente al cambio de las variables o no, se tiene que comprobar que si al leer la señal analógica del sensor más exterior de la izquierda (llamado sensor izquierda2), este devuelve un valor mayor que mil, la lógica consideraría que el coche avanza sobre una curva cerrada hacia la izquierda, y se procedería a cambiar el valor de la variable velocidad por 95 y el valor de la variable asociada al ángulo de giro, también por 95. Finalmente se hayan cambiado los variables, o no, se comprueba si al realizar una lectura sobre el sensor central, llamado sensor centro,

este nos devuelve un valor mayor que mil. En caso de que esta condición se evalúe a verdadero, se procederá a cambiar las variables de velocidad por 160 y a establecer el valor de la variable del ángulo en 125. Todos los cambios en las variables se hacen siempre y cuando las condiciones de la lectura del sensor devuelvan un valor mayor que 1000.

Esto se hace de esta forma para que se almacenen las variables que se escriben en los diferentes componentes hardware del robot, y así en caso de fallo, o de salida del circuito, este se mantenga circulando de la misma manera que lo estaba haciendo, ya que el caso típico de salida de pista es en una curva, por lo tanto necesitamos que el coche siga girando aunque no detecte la línea, ya que si sigue girando en esa dirección, llegará el momento en el que se vuelva a encontrar con la pista.

Después de hacer las comprobaciones sobre el sensor central, y cambios en las variables si fuese necesario, se procede a escribir estas variables sobre los componentes que controlan el movimiento del coche. Primero se escribe en el servo el valor de la variable ángulo que se ha establecido anteriormente. Con esto cambiamos el ángulo de giro. Después hay que utilizar la función *delayMicroseconds(1000),* la cual sirve para esperar 1000 milisegundos. Esto se hace debido a los problemas que se pueden originar al realizarse las operaciones de escritura tanto analógicas como digitales, u operaciones de lectura de la señal analógica de los sensores, ya que todas estas operaciones se hacen en un bucle. Por lo tanto, al ejecutarse todas estas operaciones con tan poco margen de tiempo puede provocar problemas en el funcionamiento del sistema de guiado, ya que las operaciones de lectura y escritura podrían realizarse mucho más rápido de lo que cambia el hardware y el entorno por el cual se mueve el robot. Es por ello que necesitamos un retardo.

Así que después de esperar un segundo después de escribir el ángulo en el servo, se cambia también la velocidad, o el sentido en caso de que sea necesario, mediante la función *moverCoche(sentido, velocidadD, velocidadI),* la cual tiene tres argumentos en los cuales se pasan como parámetros las variables de velocidad y la del sentido del coche. Esta función comienza comprobando el parámetro sentido, el cual es un valor de tipo binario (un valor de 0 significa que se requiere que el coche circule hacia adelante, y con un 1 hacia detrás). Cuando el coche tiene que circular hacia delante, la función escribe

dos señales digitales en el controlador del motor, una de alto nivel que va hacia el primer pin del motor izquierdo, y otra de bajo nivel que va hacia el otro pin del motor. Análogamente, se realiza el mismo proceso con el motor de la derecha. En el caso de que el coche tenga que circular hacia detrás, se escribe el nivel bajo en el primer pin de los motores, y el nivel alto en el segundo pin. Después de estas acciones, se procede a cambiar la velocidad a la que giran los motores mediante el valor que se ha pasado por el parámetro llamado velocidadD, que representa la velocidad del motor derecho, y velocidadI para el motor izquierdo. Por lo tanto, haciendo dos escrituras analógicas sobre los pines dirigidos al controlador de motores, se cambia la velocidad del robot.

Tras las primeras iteraciones del sistema de guiado, se observó que el coche tenia ciertas dificultados a la hora de girar en las curvas mas cerradas, por esto, aprovechando que a cada motor se le puede escribir una velocidad diferente, cuando el sistema de guiado detecta que el coche esta circulando por una curva cerrada, este además de reducir la velocidad como se ha especificado anteriormente, dependiendo de hacia donde sea la curva, las velocidades de las ruedas serán diferentes, en concreto, si la curva es hacia la izquierda, la rueda derecha disminuirá a una velocidad de 95 , y la rueda izquierda un poco menos para poder hacer un giro mas cerrado, en este caso 125. Y al girar a la derecha, las velocidades serán las mismas, la diferencia es que la rueda derecha girará a 125 y la izquierda a 95

En conclusión, se trata de un código sencillo, bien optimizado y robusto, que se ha refinado bastante para conseguir un sistema sin errores, así como se ha minimizado la cantidad de código necesario para el funcionamiento del sistema de guiado. Gracias a la metodología utilizada, en la cual en cada iteración que se hacía del sistema de guiado éste iba mejorando poco a poco hasta conseguir el resultado final, se ha conseguido un funcionamiento prácticamente sin errores.

6. Pruebas

Esta parte del proyecto está entre las más importantes. En el proyecto del sistema de guiado, esta fase tiene aún más protagonismo debido a la metodología utilizada para el desarrollo del proyecto (iterativa incremental). Como ya se ha comentado, en esta

metodología se ha decidido que cada iteración contenga tres fases esenciales en los proyectos software, es decir, el diseño, el desarrollo y las pruebas del software. La importancia de esta última fase en esta metodología radica en que, al probar el resultado final de cada iteración, la prueba no certifica el correcto funcionamiento, se vuelve de nuevo a la fase de desarrollo hasta que el funcionamiento sea el correcto y se puedan aprobar todas las pruebas pertinentes de cada iteración, acumulando además las pruebas de las iteraciones anteriores. De esta forma nos aseguramos de que, al hacer cualquier cambio, corrección, o mejora, todo lo realizado en las anteriores iteraciones sigue funcionando a la perfección.

En la primera iteración de la metodología se ha trabajado sobre el movimiento del coche. Para comenzar, se tenía que desarrollar este apartado, por lo tanto, las pruebas en

```
void loop()
{
 for(int i = 100;i<151;i++){
    myservo.write(i);
    delay(100);
    movimiento(ADELANTE,100);
    }
    for(int i = 150;i>99;i--){
      myservo.write(i);
      delay(100);
      movimiento(ATRAS,100);
    }
    char res = analogRead(PinEJEMPLO1);//Prueba de leer pin conectado a los sensores.
    print(res);
}
```

esta iteración fueron enfocadas a esta funcionalidad. Para empezar, se ha probado que el funcionamiento de este tipo de movimiento es el correcto. Para ello se ha elaborado la primera iteración de la funcionalidad moverCoche(sentido,velocidad)

Como podemos ver en el código anterior, esta prueba trataba de ver el correcto funcionamiento de todas las constantes definidas, comprobando también la conexión y comunicación de los diferentes componentes, mediante lecturas o escrituras en los pines. Principalmente la prueba del movimiento consistía en observar que durante 10 segundos el coche avanzaba hacia adelante, y en los 10 segundos siguientes hacia atrás, repitiendo este proceso una y otra vez.

Una vez superada esta prueba, ya podíamos asegurar que la implementación de la funcionalidad que permitía el movimiento del coche era la correcta. Además, al realizar

estas primeras pruebas se puedo sacar la conclusión de que el coche necesitaba un tiempo de espera entre ciertas partes del código. Además, se pudo saber que cuando el código contiene estas funciones de *delay()*, las acciones de los componentes hardware se ralentizan, es decir, que la velocidad elegida en la ilustración es la de 100, pero en la práctica, al poner las funciones *delay()* en los bucles, la velocidad que alcanzaría el coche sería claramente inferior a la que alcanzaría en caso de no tener estas funciones.

Después de estas primeras pruebas, en la siguiente iteración se pasó a implementar el funcionamiento del cambio de dirección. Para ello se hizo que en el código anterior además de desplazarse, el ángulo de giro también cambiase mientras se el robot desplazaba, con lo cual las pruebas para esta iteración se podrían combinar fácilmente con las pruebas de la iteración anterior. De esta forma, a las pruebas de la iteración anterior se añadió la prueba girar a la izquierda mientras se estaba moviendo hacia adelante el coche por un total de 15 segundos, y cuando terminase de realizar estas acciones, el coche retrocediese y además el ángulo de giro cambiase para poner las ruedas hacia la derecha, también durante 15 segundos. Al estar todo esto dentro del bucle principal, se tenía que repetir una y otra vez, sin cambiar el comportamiento.

Las siguientes tres iteraciones se han dedicado por completo al desarrollo del sistema de guiado utilizando los sensores, junto con lo desarrollado hasta ahora por las dos primeras iteraciones. En primer lugar, después de conseguir la primera versión funcional había que probar que el sistema de guiado funcionase. En las siguientes iteraciones nos dedicaríamos a pulir cualquier tipo de imperfección o mejorar lo que ya teníamos. Por lo tanto, la batería de pruebas desarrollada en esta iteración para probar esta funcionalidad, consiste primero en dibujar en el suelo con cinta aislante negra una pista que el coche iba a tener que seguir. Sin embargo, había que poner a prueba cada sensor en específico, ya que cada sensor va a cubrir una circunstancia en concreto. Y no solo se limita a esto, porque como se ha comentado antes, estas pruebas no tienen que probar simplemente lo desarrollado en esta iteración, si no que esta batería tenía que hacerse para además probar lo que se había hecho ya en otras iteraciones.

Como se puede observar en la tabla 5, se ha obtenido una batería de pruebas bastante completa, que cubre el 100% de los componentes hardware y software. Por ello, esta batería de pruebas se ha utilizado también las siguientes dos iteraciones, en las que

se han añadido los casos específicos de las mejoras realizadas durante las últimas iteraciones.

En el caso de la mejora sobre el giro en las curvas cerradas, se probó que, con dos velocidades diferentes en los motores al realizar curvas cerradas, la circulación continuaba siendo fluida, y que al coche no le costaba seguir hacia adelante, tanto para las curvas cerradas a la izquierda como para las curvas cerradas a la derecha, ya que un problema conocido del coche es que sufría de pequeños parones cuando la velocidad era demasiado reducida. En este caso, en las ruedas contrarias a la dirección del giro la velocidad es un poco superior a la velocidad que tiene la otra rueda. Esto se hace para que el coche se comporte mejor en curvas más cerradas. Por lo tanto, las curvas que se dibujarán para esta batería de pruebas deberían ser más cerradas de lo que eran al realizar las pruebas de la primera versión del sistema de guiado.

Finalmente, en la última iteración se ha tratado de mejorar la eficiencia del código. Es por esto que no se requerían pruebas adicionales, porque no se ha cambiado ningún aspecto del funcionamiento, si no que se ha mejorado el que ya se había desarrollado. En este caso, simplemente se deberían de volver a pasar las pruebas que se habían realizado hasta ahora. Sin embargo, había que poner un énfasis en la fluidez de los movimientos del coche y su circulación, ya que se había cambiado la función *delay()* por la *delayMicroseconds()*, bajando el tiempo de espera de 1 segundo a 100 milisegundos. Por lo tanto, iba a aumentar la velocidad al realizar las acciones y ejecutar tantas lecturas como escrituras. Este cambio mejoraba la fluidez del coche, ya que en las últimas iteraciones la fluidez de circulación se había visto comprometida hasta el límite.

Prueba	Descripción	Resultado esperado
Prueba unitaria sobre el sensor central.	Se ha desarrollado un circuito que es una línea recta por el cual tiene que circular el coche.	Se espera que la circulación del coche sea fluida a simple vista. Además, no tiene que observarse ningún tipo de cambio en el ángulo de la dirección del servo (fijado a 125). Tampoco se espera que haya ningún tipo de cambio de velocidad durante la circulación del coche.
Prueba unitaria sobre el sensor interior izquierdo.	Se ha desarrollado un circuito que consiste en un enorme circulo, lo que simula una curva abierta siempre para el coche. Éste tendrá que circular en sentido antihorario.	Se espera que además de que la circulación sea fluida, el coche siempre interprete que está sobre una curva abierta hacia la izquierda, fijando un ángulo de 115 y una velocidad un poco más reducida, en este caso 140. Estos valores no tienen que cambiar en ningún momento.

Prueba unitaria sobre el sensor interior derecho.	Aprovechando el círculo dibujado anteriormente, esta vez se pone el coche a circular en sentido horario, simulando una curva siempre abierta hacia la derecha.	Se espera que la circulación sea fluida y que el coche interprete que está sobre una curva abierta hacia la derecha. Por lo tanto, siempre debe mantener el ángulo de 135 y una velocidad un poco más reducida, en este caso 140. Estos valores no tienen que cambiar en ningún momento.
Prueba unitaria sobre el sensor exterior izquierdo.	Se ha dibujado un circuito más pequeño que el anterior para simular una curva siempre cerrada. Para que esta curva siempre sea a izquierdas el coche tiene que circular en sentido antihorario	Se espera que la circulación del coche continúe siendo fluida, y que además tanto el ángulo de giro como la velocidad no cambien en ningún momento, es decir, que estos siempre sean 95. Además, al tratarse de una curva cerrada también se necesita que el coche no se salga de la pista, o que al menos no lo haga de forma exagerada.
Prueba unitaria sobre el sensor exterior derecho	Aprovechando el circulo anterior, se tiene que poner el coche a circular en sentido horario, para simular una curva cerrada hacia la derecha	Se espera que la circulación del coche continúe siendo fluida, y que además tanto el ángulo de giro, como la velocidad no cambien en ningún momento, es decir que el ángulo siempre sea 155 y la velocidad 95. Además, al tratarse de una curva cerrada también se necesita que el coche no se salga de la pista, o que al menos no lo haga de forma exagerada.
Prueba de integración de todos los sensores.	Finalmente se ha diseñado un circuito lo más irregular posible, en el cual se alternan las rectas y las curvas cerradas y abiertas, hacia la izquierda o derecha, con diferentes ángulos. Además, este circuito se ha hecho en bucle, de tal manera que la prueba fuese automática y solamente se tuviese que observar.	Simplemente se espera que la circulación sea fluida, y que el coche sepa responder automáticamente a las circunstancias especiales de cada tramo del circuito, sin salirse más de 12 centímetros de este, empezando a contar desde el eje central del coche.

Tabla 5 Pruebas sobre la primera versión funcional del sistema de guiado

7. Relación con los estudios

Este proyecto ha abarcado diferentes áreas de conocimiento de la Ingeniería Informática, como pueden ser la gestión de proyectos software, el desarrollo de aplicaciones, tecnologías de computadores, mantenimiento del software, ingeniería del software, arquitectura, o incluso ingeniería del hardware. En concreto, ha sido un proyecto que ha combinado tanto el hardware como el software que va a controlarlo. Es por eso que durante todo el desarrollo del software, siempre se ha puesto en práctica todo el conocimiento que se ha adquirido en asignaturas enfocadas tanto al hardware como al

software. Por ejemplo, para comprender el funcionamiento de varios de los componentes hardware que constituyen el sistema de guiado, han sido de gran ayuda los conocimientos adquiridos en la asignatura de tecnologías de computadores, y no solo esto, si no también toda la terminología acerca de los circuitos integrados, y toda la base que se estableció sobre el hardware en los sistemas informáticos, al cursar esta asignatura.

El haber aprendido gran variedad de lenguajes de programación en varias asignaturas de la carrera ha servido para que resulte más sencillo aprender a programar en este nuevo entorno de programación, así como para facilitar la asimilación de la semántica de este lenguaje y todo su funcionamiento.

Una parte importante del proyecto ha sido su planificación. Han sido varias asignaturas en las cuales se han visto diferentes metodologías de trabajo, que incluyen cómo gestionar el proyecto, cómo gestionar los tiempos o cómo mejorar el proyecto en cada una de sus fases. En este proyecto han tenido un papel importante las asignaturas de ingeniería del software, y las de gestión de proyectos. Las asignaturas de la rama de ingeniería del software me han enseñado a cómo escribir un código de calidad, que cumpla ciertos requisitos, características y propiedades, no solo mostrando cómo es un código de calidad, si no enseñando qué es la calidad, y de las métricas y propiedades de las que se compone.

Finalmente, hay una asignatura optativa que realmente ha sido bastante importante para este proyecto. Se trata de Diseño e Interconexión de Periféricos, en la cual se enseñaba el funcionamiento de Arduino y varias herramientas relacionadas con el mismo. Por lo tanto, esta asignatura ha estado estrechamente relacionada con el proyecto que se ha realizado, y en la cual se ha consultado y ha servido de apoyo para llevar a cabo el proyecto.

8. Conclusiones

Para concluir, desde un punto de vista personal, llevar a cabo este proyecto ha cumplido mis expectativas, aunque ha sido más largo de lo que me podría esperar. Este proyecto ha provocado que ponga en práctica bastantes de los conocimientos que había adquirido durante la carrera, además de tener la oportunidad de explorar un nuevo mundo para mí, y conseguir obtener nuevos conocimientos. En este caso, me ha dado la posibilidad de profundizar en la plataforma Arduino. Al final se ha conseguido construir el sistema de guiado satisfactoriamente, y mediante todo lo que he auto aprendido, esto es probablemente de lo que más satisfacción me otorga de este proyecto. Después de la gran cantidad de horas que le he dedicado a aprender las funcionalidades de los componentes que tenía a mi alcance, he podido hacerlos funcionar y finalizar todo un proyecto por mí mismo.

También, a la hora de pensar la metodología que se iba a utilizar, y cómo se iba a llevar a cabo el trabajo, además de toda la planificación que iba a tener éste en función de las decisiones que se iban tomando, me he dado cuenta de que he sido capaz poner en práctica exitosamente lo aprendido acerca de metodologías de trabajo en proyectos software, y la gestión de proyectos, ya que la planificación la he diseñado yo personalmente, ya que he sido la única persona que ha trabajado en este proyecto. Es por eso que esta planificación se ha adaptado muy bien a mis necesidades, y a pesar de que el proyecto se ha extendido tanto, el desarrollo del mismo se ha hecho llevadero, lo cual es una de las cosas más importantes que he aprendido acerca de la gestión de un proyecto, y el desarrollo del mismo, que no solo importa el fin, si no que los medios a utilizar es algo a tener muy en cuenta. No es lo mismo haber hecho el proyecto a última hora que haberlo extendido a lo largo del tiempo. Eso sí, teniendo claros los plazos de entrega.

Finalmente, este proyecto me ha servido para aumentar mis conocimientos en varios aspectos de la Ingeniería Informática. También me ha dotado de una cierta experiencia a la hora de enfrentarme a cierto tipo de problemas en proyectos de software, y situaciones que se plantean en el día a día del desarrollo software. Además, al utilizar una tecnología desconocida para mí y empezar desde cero aprendiendo esta tecnología, llevar a cabo un proyecto a la vez que he ido aprendiendo es un gran logro, y cumple con

gran parte de los objetivos marcados antes de desarrollar este proyecto. Para mí ha sido una experiencia enriquecedora en muchos aspectos. Tanto es así que un porcentaje muy alto de todo lo que se ha explicado en este documento han sido cosas nuevas para mí, cosas que he aprendido por mí mismo para desarrollar el proyecto y que seguramente me sirvan en un futuro.

9. Referencias

- Arduino.cc. (2018). *Arduino - Introduction*. [online] https://www.arduino.cc/en/Guide/Introduction
- *Arduino*. (2018). *Es.wikipedia.org*. https://es.wikipedia.org/wiki/Arduino
- *Raspberry Pi*. (2018). *Es.wikipedia.org*. de https://es.wikipedia.org/wiki/Raspberry_Pi
- *Smart Car Kit for Arduino*. (2018). *Sunfounder.com*. https://www.sunfounder.com/learn/category/Smart-Car-Kit-for-Arduino.html
- Arduino.cc. (2018). *Arduino Reference*. https://www.arduino.cc/reference/en/
- *Raspberry Pi FAQs - Frequently Asked Questions*. (2018). *Raspberry Pi*. https://www.raspberrypi.org/help/faqs/#introWhatIs
- GODOY, Jorge. Arquitectura de control para la conducción autónoma de vehículos en entornos urbanos y autovías. 2013.
- GARCIA BLÀZQUEZ, Daniel. Projectes amb Arduino: Creació d'una documentació d'Arduino per l'assignatura de Robòtica, Llenguatge i Planificació.
- RUI-YAN, C. A. I. Principle and application of Arduino [J]. Electronic Design Engineering, 2012, vol. 16, p. 047.
- Introducción a Arduino. Hardware y Programación What is it ? (n.d.).
- De, I. T. (n.d.). Buses Contenidos.
- Jimenez Montero, F. J. (2007). Sensores y Actuadores. *Publicacion En Internet*, 1–20. Retrieved from http://es.slideshare.net/kaluisitoblog/sensores-yactuadoresseat
- Bauman, Z. (2001). O D Er N Ity, (50), 163–191.

10. Agradecimientos

Dentro de un contexto más personal, en este apartado se lo dedicaré a aquellas personas que me han ayudado, no solo durante el proyecto, si no durante toda mi vida como estudiante. Como me decía un profesor que tuve en tercero de la ESO, tu padre se lo está currando, y es cierto que se lo han currado, por ello me gustaría dedicar un agradecimiento especial a mis padres, todo es gracias a ellos.

También me gustaría agradecer la ayuda prestada por mi tutor del proyecto de fin de grado, Joaquín Gracia, ya que me ha ofrecido toda la ayuda posible. Es más, al yo tener un horario bastante complicado, él me ha ayudado prestándome el material necesario para poder llevar a cabo este proyecto.

También me gustaría agradecer a Ángel Rodas, y la asignatura de Diseño e Interconexión de Periféricos, que me han sido de bastante ayuda durante mi época de aprendizaje. Además, Ángel me ayudó a compaginar la asignatura con mi horario. Esto me dio bastante libertad a la hora de realizar este proyecto, relacionándolo con la asignatura. Una pena que este sea el último año de la asignatura que me ha dado la posibilidad de aprender algo muy interesante.

ANEXO

Diccionario

Open Source: También conocido como código abierto, se trata de software de que además de ser distribuidos de forma gratuita, también se puede modificar su código fuente para cualquier uso, sin restricciones de licencia, este modelo, se basa en la colaboración de la comunidad, y en los beneficios de esta colaboración.

Shield: Se trata de una interfaz hardware, para abstraer diversas funciones en un módulo, lo cual permite llevar a cabo los proyectos con Arduino de una forma más sencilla.

Raspberry pi: Se trata de un ordenador de tamaño reducido, el cual esta compuesto de una única placa, en el cual se puede instalar un sistema operativo, y hacerlo funcionar como un ordenador.

Breadboard: Se trata de un a placa de pruebas, que tiene varios orificios conectados entre si internamente, en los cuales se pueden conectar diferentes elementos electrónicos, para el prototipado, y diseño de circuitos electrónicos.

Efecto fotoeléctrico: Se produce cuando sobre un material se incide radiación, como por ejemplo luz, y esto provoca que el material emita electrones.

TTL: Proviene de las siglas en inglés, *transistor-transistor logic*, lógica transistor a transistor, se trata de una tecnología propia de los componentes electrónicos digitales.

IDE: Son las siglas en inglés de *integrated development envoviroment*, que significa entorno de desarrollo integrado, se trata de un programa, que facilita el desarrollo software, existen diferentes entornos de desarrollo enfocados a los diferentes lenguajes de programación, o al componente software que se esta desarrollando