Entorno IoT para la ejecución de comportamientos inteligentes

IoT enviroment for the execution of intelligent behaviours

Juan Manuel Salvá Rossich

[GRADO EN DESARROLLO DE VIDEOJUEGOS]

UNIVERSIDAD COMPLUTENSE DE MADRID

UNIVERSIDAD COMPLUTENSE MADRID

[TRABAJO FIN DE GRADO]

Curso 2021-2022

Director: Juan A. Recio García

En colaboración con la cátedra Bosch-UCM en Inteligencia Artificial aplicada a Internet de las Cosas

Agradecimientos

Tras un proyecto tan largo e intenso, es lo correcto mencionar y agradecer a aquellas personas las cuales han contribuido al trabajo de la manera en la que han podido. Primero de todo agradecer el trabajo al tutor del proyecto, Juan Antonio, por el seguimiento semanal del proyecto y la ayuda a la hora de guiar y focalizar el trabajo. Afrontar un trabajo de estas magnitudes por primera vez sin ayuda de ningún tutor no es tarea fácil y se agradece tener a alguien al que acudir con cualquier duda.

También agradecer a la de la cátedra Bosch-UCM en inteligencia Artificial orientada al Internet de las Cosas por el apoyo y la integración del proyecto. A Carlos, representante de Bosch por acudir a las reuniones de seguimiento del trabajo y por conseguir todos los componentes necesarios para el desarrollo.

Gracias también a mi primo David por ayudarme inmensamente por la ayuda a la hora de enfocar el inicio del trabajo aún meses antes de empezar el desarrollo oficialmente. Sin duda, sin su ayuda este trabajo no se habría ni empezado y fue una pieza clave sobre todo al inicio del desarrollo. Básicamente ha sido un segundo tutor no oficial.

Finalmente, una mención a todos aquellos familiares, amigos y compañeros que han apoyado el proyecto. Sobre todo, a aquellos que no se han bajado del barco aún en los peores momentos en los que parecía que no se llegaría a nada.

Resumen

Desde ya hace unos años, el internet de las cosas (IOT)[1] está en constante expansión tocando ámbitos que antes eran inimaginables, pero aún hay un ámbito en el que no se está usando demasiado, en los videojuegos.

En este trabajo se pretende explorar las posibilidades de fusionar videojuegos software con réplicas robóticas físicas mediante las tecnologías del internet de las cosas. Concretamente se propone replicar el juego arcade clásico MsPacMan[2] con robots físicos que se desenvuelven en un tablero real por medio de distintos sensores. El control de estos robots recaerá sobre el videojuego software, en el que se ejecutan distintos comportamientos inteligentes. Un sistema de comunicación Bluetooth permitirá enlazar la implementación virtual con la física de modo que el videojuego se reproduzca en ambos entornos.

En este trabajo, se presentan distintas iteraciones del sistema RoboPacMan donde se evoluciona tanto la arquitectura y software de los robots, como la representación e implementación del tablero, analizando los beneficios y carencias de cada aproximación.

Palabras clave

IOT

MsPacMan físico

Inteligencia artificial

Robots

Sensores

Seguimiento de línea

Impresión 3D

Abstract

For some years now, the Internet of Things (IoT)[1] has been in constant expansion, touching areas that were previously unimaginable, but there is still one area in which it is not being used too much, video games.

This project aims to explore the possibilities of merging videogames with physical robotic replicas using the Internet of Things technology. Specifically, the proposal is to replicate the classic arcade game MsPacMan[2] with physical robots that operate on a real board by means of different sensors. The robots will be controlled by the videogame and the different sensors, in which different intelligent behaviors are executed. A Bluetooth communication system will allow linking the virtual implementation with the physical one so that the videogame is played in both environments.

In this project, different iterations of the RoboPacMan system are presented, where both the architecture and software of the robots, as well as the representation and implementation of the board are evolved, analyzing the benefits and shortcomings of each approach.

Key Words

IOT

Physical MsPacMan

Artificial Intelligence

Robots

Sensors

Line tracking

3D printing

Índice general

Capítulo 1

Punto de partida

"Te veré en la línea de meta, amigo."
-Rayo McQueen

La idea de este trabajo es replicar el juego MsPacMan en versión física con robots moviéndose de manera autónoma por el suelo sobre un tablero usando una versión modificada del simulador MsPacMan vs Robot desarrollado por el grupo GAIA[3] y una inteligencia artificial propia.

Actualmente, no hay nada similar para utilizar como punto de partida, por ello, es necesario mezclar diversos conceptos para conseguir el resultado final. Existen competiciones de robots que se dedican a seguir líneas[4] y este será el punto de partida. Desde este concepto se tendrá que extrapolar a robots inteligentes que tendrán que hacer más cosas aparte de seguir una línea por el suelo como podría ser la realización de giros de noventa grados y la comunicación entre ellos.

1.1 Motivación

Desde el nacimiento del IOT, ha sido necesario usar distintas tecnologías para lograr un resultado prometedor. Además, para que estos resultados sean inteligentes es aún más importante la cooperación de estas tecnologías.

La mayor dificultad del internet de las cosas es que no siempre se está en un entorno controlado a la perfección como puede ser un videojuego, por ello, mezclar este entorno totalmente controlado del MsPacMan con el entorno físico de la realidad en el que afectan una infinidad de factores será el mayor reto del trabajo.

Se ha elegido usar el MsPacMan como videojuego a replicar por la simpleza de las decisiones que se toman durante el juego. Tan solo se pueden realizar cuatro acciones (subir, bajar, derecha e izquierda) en cada intersección del mapa. Esto a la hora de crear los robots ayuda ya que con dotarles con la capacidad de girar noventa grados en una determinada dirección al pasar por encima de una intersección sería suficiente, siempre y cuando estos giros sean lo suficientemente precisos.

Además, a efectos prácticos, estos robots serán coches autónomos en miniatura sobre los que se pueden realizar pruebas rápidas y baratas que potencialmente se podrían aplicar a proyectos de mayor tamaño.

Por tanto, el objetivo es experimentar en este sentido, intentar seguir una partida de MsPacMan tan solo mirando el comportamiento de los robots. Que sean capaces de comunicarse entre sí realizando las acciones necesarias para no perderse y evitar colisiones durante un periodo extendido de tiempo sin input de ningún humano.

1.2 Hipótesis de partida

Las tecnologías de IoT permiten hacer una reproducción física de determinados videojuegos a través de distintos sensores y actuadores. Estos elementos permitirán al robot desenvolverse por un entorno real. Concretamente, a través de un comportamiento de seguimiento de línea se puede obtener un sistema de orientación y dirección bastante eficaz y fiable. El mapa estará representado por dos líneas paralelas en el suelo y las intersecciones por un punto negro en el medio. Los sensores del robot serán los encargados de identificar estas líneas para mantener el rumbo del robot en la dirección correcta.

1.3 Objetivos

A través de este trabajo se tratará de obtener conocimientos que se expanden desde un programa informático hasta la construcción de robots inteligentes. La idea es analizar la viabilidad y el éxito del uso de robots para la representación física de un videojuego retro.

En concreto se evaluará la efectividad de estos robots analizando los resultados del comportamiento basándose en la precisión de sus movimientos y la capacidad de completar una partida normal.

Al ser un proyecto en el que se tendrá que crear un producto físico desde cero, habrá diversas iteraciones de este, puesto que, siendo realistas, es prácticamente imposible que salga bien a la primera. Resumidamente, habrá que seguir un proceso iterativo hasta obtener el resultado esperado tal y como se indica en la Ilustración 1

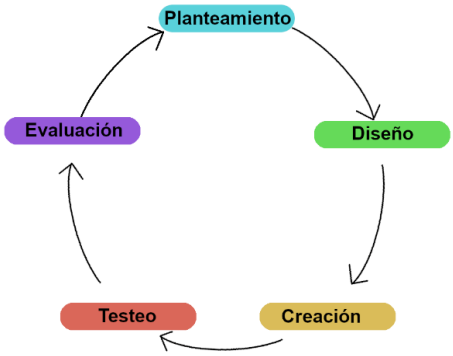


Ilustración 1, proceso iterativo a seguir durante el desarrollo

Objetivos principales que se quieren lograr:

- Integración del juego MsPacMan con un *framework* que permita la extracción de datos para así poderlos usar para la representación física del videojuego
- Entender el funcionamiento básico de robots, su construcción y la manera en la que se programan
- Desarrollar un sistema inteligente capaz de leer el entorno tomando las decisiones correctas para mantenerse en funcionamiento correctamente mientras además juega una partida de MsPacMan.
- Creación de un sistema de comunicación bidireccional entre Arduinos y Pc via bluetooth capaz de controlar un mínimo de cinco dispositivos simultáneamente.

1.4 Tareas

Para alcanzar los objetivos anteriormente descritos se definieron las siguientes tareas

- T1: Diseño en 3D de prototipos de robots móviles con sensores infrarrojos
- T2: Impresión en 3D de los prototipos y ensamblaje de los componentes necesarios
- T3: Programación del microcontrolador del robot
- T4: Programación de distintos controladores de los sensores e integración con el microcontrolador central
- T5: Adaptación del entorno MsPacMan Engine para su comunicación con los robots móviles
- T6: Desarrollo de la infraestructura Bluetooth necesaria para la intercomunicación de los destinos dispositivos

1.5 Metodologías

Al ser un trabajo individual no es necesaria la práctica de *scrum* semanalmente. Cada dos semanas se llevará a cabo una reunión con los tutores para el correcto seguimiento del trabajo.

Ser una sola persona haciendo el trabajo no significa que no sea necesaria una planificación, por ello, antes de empezar se ha hecho una planificación de todas las tareas que son necesarias completar y una estimación del tiempo que podrían llevar.

En la Ilustración 2 vemos una captura de pantalla de una parte de la planificación del proyecto. Cada tarea está relacionada a un color indicando su prioridad. Además, a cada una se le aplica un plazo estimado el cual, si varía, mueve automáticamente el resto de tareas.

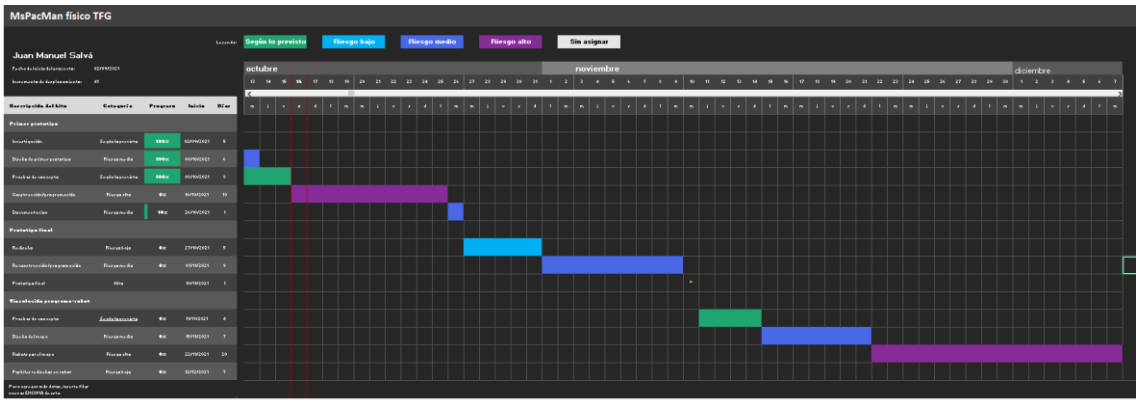


Ilustración 2, planificación de las tareas con el tiempo estimado de cada una

La idea principal es dedicarse hasta finales de diciembre en la creación y refinado del robot y a continuación seguir con la parte del *software* para conseguir el resultado esperado.

Los programas que se van a utilizar para el desarrollo son:

- Fusion360: para el diseño y modelado de todos los componentes necesarios
- Prusa Slicer: para convertir los modelos en archivos del formato *gcode* que se podrán mandar a la impresora (una Prusa Mini) para imprimir.
- Visual Studio Code y Arduino IDE: para la programación del software de la parte de los Arduinos
- Eclipse: para programar todo lo relacionado con el simulador MsPacMan
- Microsoft Teams: para tener un espacio compartido con todos los archivos accesibles tanto por el alumno como por los tutores.
- GitHub: para tener un control de versiones y poder trabajar cómodamente. El enlace al repositorio público en el que se va a desarrollas todo el proyecto es el siguiente: https://github.com/JuanmaSalva/MsPacManFisico

1.5 Estructura del resto del documento

En el segundo capítulo se estudiará como se podría afrontar el trabajo, que componentes debería tener el robot, como conectarlos entre si… Se estudiarán trabajos relacionados de los que se pueda extraer algunos datos para facilitar el desarrollo y por último se estudiarán las tecnologías necesarias.

El tercer y cuarto capítulo serán los capítulos de contribución. En estos capítulos se resume el trabajo realizado, la arquitectura y las pruebas realizadas

En el quinto capítulo se discute las ventajas y limitaciones del trabajo, así como analizar las conclusiones y el trabajo futuro que le quedaría al proyecto.

Capítulo 2

Introducción al proyecto e investigación previa

"No necesito saber a dónde voy,
sólo necesito saber dónde he estado… "
-Mate

2.1 Introducción

En este capítulo se discutirá toda la investigación necesaria para entender el problema el cual se quiere solucionar y hacerlo de la manera más eficaz posible.

Antes de todo, hay que dividir lo que se quiere que haga el robot en problemas más pequeños los cuales ya tengan solución y unificarlos al final consiguiendo el comportamiento deseado. Los tres grandes problemas para solucionar son: ¿cómo se va a mover?, ¿cómo va a saber el robot por dónde ir?, ¿cómo se va a comunicar con el resto de los robots?

La principal inspiración para el proyecto fueron las competiciones de robots siguiendo líneas[4]. En estas competiciones, hay un recorrido marcado en el suelo mediante unas líneas negras, y los robots tienen que realizar todo el recorrido lo más rápido posible sin desviarse. El problema con estos robots es que tienen un comportamiento muy sencillo el cual no es suficiente para lo que se quiere conseguir en este trabajo y normalmente suelen tener un tamaño considerable, lo cual tampoco es ideal para este proyecto.

Otra inspiración es el famoso juguete *slot*[5], dado a conocer en España bajo la marca Scalextric[6]. En los nuevos productos de esta marca, los coches se controlan mediante mandos inalámbricos, se pueden realizar cambios de carril, repostar combustible y hasta variar la fuerza de trenado. Claramente, hay una comunicación entre el mando inalámbrico y el coche. Esta comunicación no es directa, ya que los coches no tienen ningún tipo de comportamiento inteligente. El mando se conecta con la centralita (esta es la parte interesante cara este trabajo) y luego ya es la centralita la cual se comunica con el coche mediante frecuencias por las vías.

La idea es mezclar estos dos conceptos, tener un control sobre el coche parecido a los Scalextrics pero que los coches se muevan usando los conceptos de los robots de las competiciones de seguimiento de línea.

En cuanto al software, el trabajo partirá del simulador MsPacMan vs Ghosts desarrollado por el grupo GAIA[3]. En cambio, en el aspecto hardware, no existen trabajos

públicos parecidos de los cuales se pueda extraer datos ni pistas de como seguir hacia delante, por tanto, es necesario mezclar tecnologías y conceptos de distintos proyectos para conseguir hacer este.

El movimiento será un movimiento de "tanque" mediante un motor eléctrico en cada rueda. Girando los motores de cada lado en sentidos opuestos se consigue rotar sobre sí mismo, de allí el movimiento de tanque. El camino que tiene que seguir el robot estará marcado en el suelo mediante líneas negras y estas se detectarán mediante unos sensores. Finalmente, la comunicación entre los distintos robots se realizará vía bluetooth.

2.2 Simulador MsPacMan

El juego MsPacMan (inicialmente llamado Crazy Otto) empezó como una secuela no autorizada del clásico juego arcade PacMan[7]. El juego se popularizó también por las facilidades que daba a la hora de programar inteligencias artificiales permitiendo formar a estudiantes. Años más tarde, Namco[8], empresa que desarrolló el PacMan original, decidió contratar a los desarrolladores del MsPacMan y de esta manera introducir el juego dentro de sus productos oficiales.

El grupo GAIA (Grupo de Aplicaciones en Inteligencia Artificial) de la Universidad Complutense de Madrid ha desarrollado un entorno de programación basado en el MsPacMan en el que los alumnos de la universidad pueden desarrollar inteligencias artificiales para su formación. Es precisamente sobre este entorno en el que se basa todo el software de este proyecto.

Sobre este simulador, los alumnos pueden iterar fácil y rápidamente distintas implementaciones y modelos de inteligencias artificiales a medida que se van obteniendo conocimientos. Además, para motivar a los alumnos, el propio simular contiene un sistema de puntuación por el cual las inteligencias artifícialas desarrolladas por los alumnos compiten entre sí para ver cuál es la mejor. De esta manera, cada alumno valora los pros y contras de las distintas aproximaciones de IAs (inteligencias artificiales) aprendidas.

De esta manera, el proyecto se extiende hasta las competiciones digitales las cuales están en constante crecimiento desde hace años. La gran mayoría de las competiciones digitales (conocidas como *esports*[9]) son entre personas jugando a un videojuego, pero también existen las competiciones entre programas informáticos, como es este caso. Normalmente estas competiciones son organizadas por programadores para programadores. A través de este proyecto, también se tratará de dar visibilidad a este tipo de competiciones a un público mucho más casual en este ámbito.

2.3 Tecnologías

A continuación, se explicarán los componentes y tecnologías que se necesitarán para la primera implementación del robot. Al no tener PCBs (circuitos impresos) personalizadas desde un inicio, cada componente se encargará de una tecnología diferente.

La idea inicial es que cada robot esté controlado por un Arduino Uno[24], que será el componente principal. El robot tendría tracción a las cuatro ruedas mediante un motor dc en cada una de ellas, consiguiendo así aceleraciones más rápidas y un mejor control a la hora de girar mediante un movimiento de tanque (girar sobre sí mismo). Para controlar los motores es necesario un controlador de tipo puente en H, para ello se usarán los controladores l298N[25] que permiten controlar dos salidas cada uno, de esta manera, solo serán necesarios dos de ellos para controlar los cuatro motores.

El coche necesitará saber por dónde va, para ello irá siguiendo las guías que habrá por el suelo mediante cinta negra. El robot sabrá la posición de dichas líneas mediante unos sensores infrarrojos, en concreto tres sensores ky-033[26] por coche. Cada uno de estos sensores (de aquí en adelante sensores de línea), mediante infrarrojos son capaces de detectar la intensidad de color de una superficie cercana, de diez milímetros a unos treinta. Esto permite saber si el color del suelo es negro o blanco. La comunicación con el servidor será a través de bluetooth usando el módulo HC-05[27].

El robot estará alimentado mediante una batería recargable de 9V. Esto muy probablemente podría cambiar en futuras iteraciones ya que de momento el amperaje de la batería soporta las piezas, pero si se cambian muchas piezas, puede variar haciendo que esta batería no fuera suficiente. En caso de que la batería no aguantara suficiente tiempo alimentando el robot, se tendrán que usar dos baterías conectadas en serie multiplicando por dos el almacenamiento de energía. En la Ilustración 3 podemos ver los componentes principales necesarios para un robot.

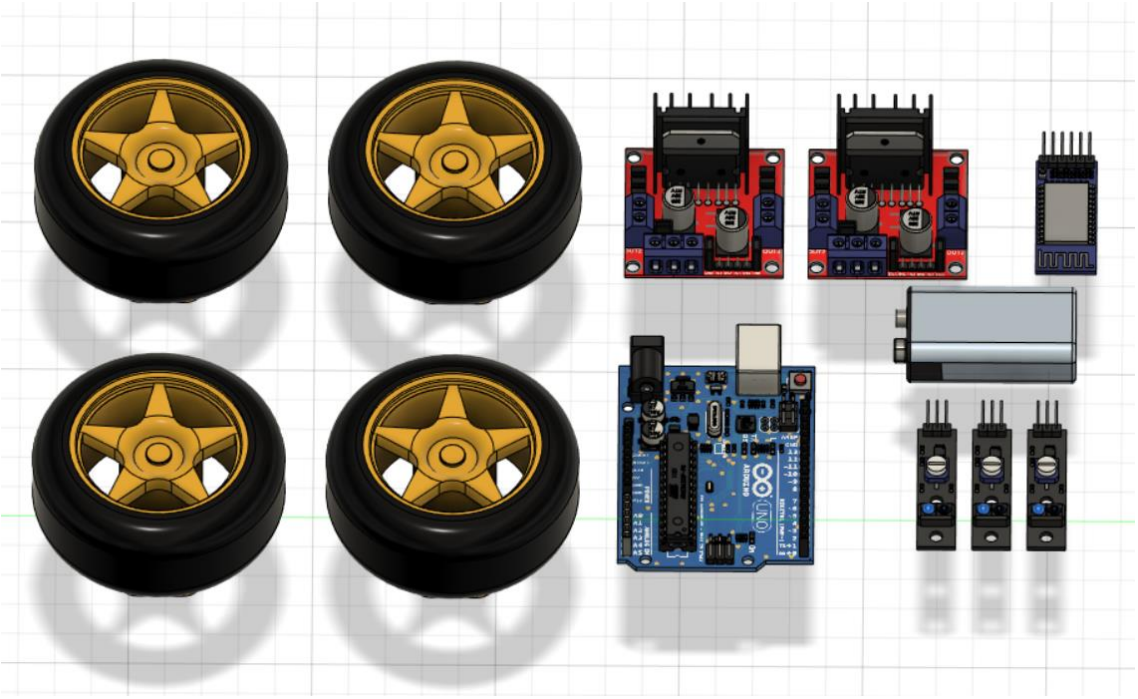


Ilustración 3, componentes necesarios para la construcción de cada uno de los robots

Para el chasis, se usará impresión 3D a través de la tecnología FDM[10], *Fused Deposition Modeling*en inglés. Básicamente es un proceso por el cual se va fundiendo material en capas sobre una base para general el producto final. Gracias a la expiración de la patente original en 2009, esta tecnología se está extendiendo rápidamente por todos los campos permitiendo un prototipado fácil, rápido y barato de componentes. Para este proyecto, el material utilizado será un filamento de termoplástico, concretamente ácido poliláctico[11] (PLA) ya que es el más accesible, más fácil de usar, más económico, biodegradable y cumple todos los requisitos del proyecto.

2.4 Pruebas con cada componente

Antes de empezar a prototipar con el robot, es necesario comprender el funcionamiento independiente de cada componente. Desarrollar un pequeño programa para cada pieza permitirá obtener conocimientos que serán de ayuda a la hora de empezar a desarrollar el robot y, además, servirán de base cuando se comience a programar los robots. Todas las pruebas se realizarán siguiendo las prácticas, consejos, y normas que fija la documentación oficial de arduino[11] y siguiendo algunos tutoriales en YouTube[12].

Conocer el comportamiento exacto de cada componente evitará errores que se podrían producir si se van incorporando nuevos componentes a destiempo ya que podría causar grandes incompatibilidades con otros componentes haciendo perder tiempo y esfuerzo. Las pruebas se irán desarrollando de las más fáciles a las más difíciles.

3.4.1 Sensor de línea

Lo más importante es averiguar el funcionamiento del sensor de línea ya que esto va a ser lo que determine la forma del robot final ya que es la única pieza que tiene que ir en un lugar fijo, cerca del suelo y en el centro de los ejes, tanto en el eje longitudinal como en el transversal (para facilitar giros con el movimiento de tanque). Para esta primera prueba, se tiene una luz led RGB de depuración para saber lo que está detectando el sensor. Podemos ver el esquema eléctrico en la Ilustración 4.

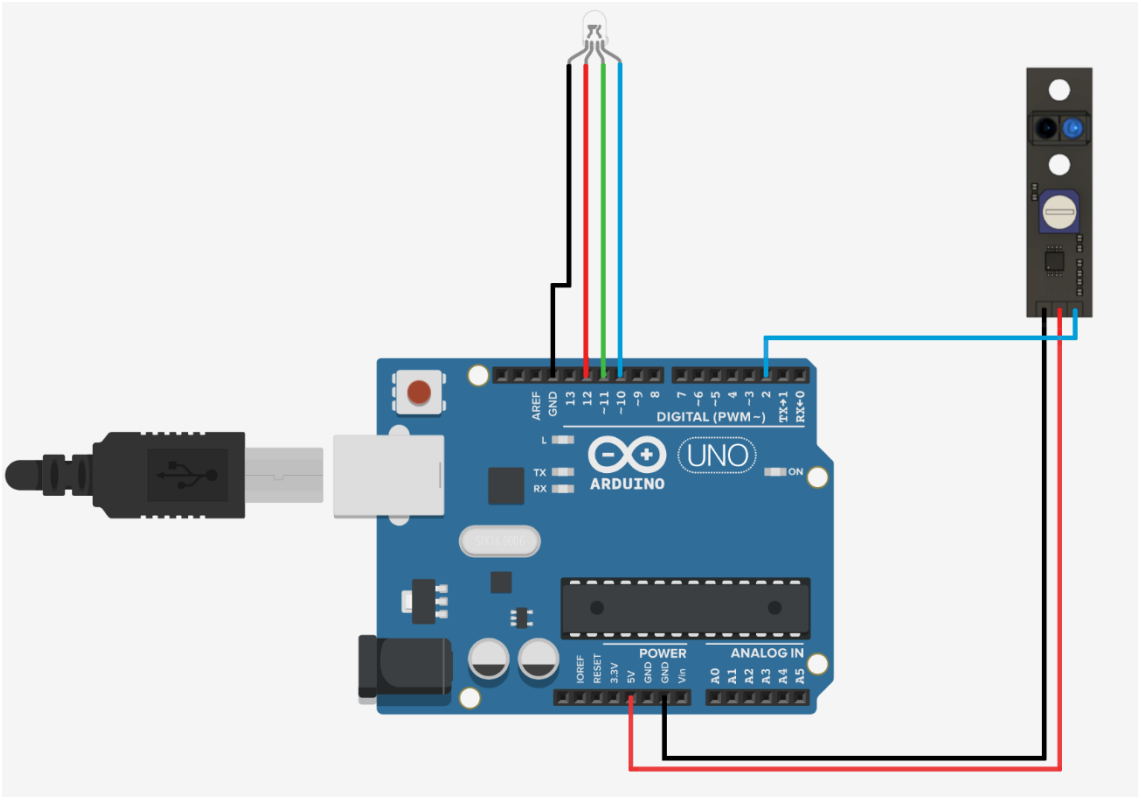


Ilustración 4, esquema eléctrico del prototipo del sensor de línea

A la hora de programar esta prueba, lo primero de todo es necesario inicializan los pines del Arduino para el sensor como datos de input y los pines del led como output.

En el bucle principal tan solo tenemos que leer el input del sensor, si el valor leído es un uno (HIGH) entonces ponemos el led verde y en el caso de leer un cero (LOW) ponemos el led rojo.

Cuando el sensor de línea saca uno de output (que el Arduino usa de input), entonces significa que estamos sobre una línea.

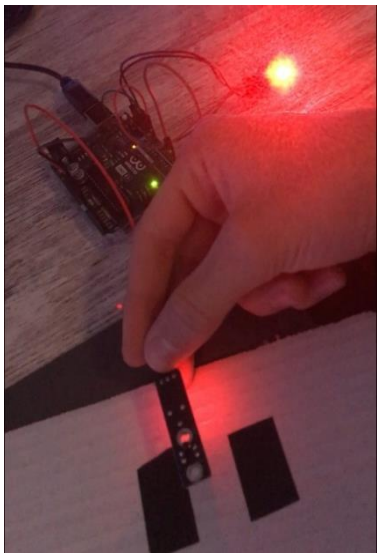


Ilustración 5, sensor de línea no detectando una línea

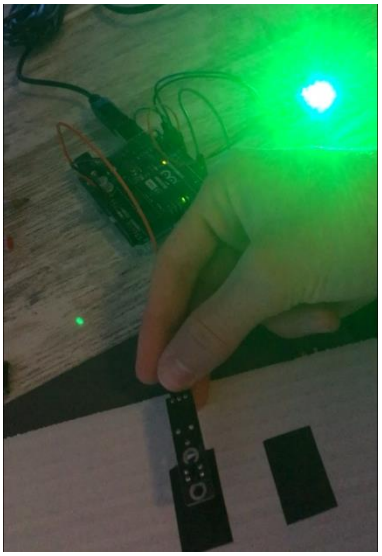


Ilustración 6, sensor de línea detectando una línea

Como se puede observar en las imágenes, el sensor funciona aun teniendo poca luz ambiente, de hecho, funciona ligeramente mejor con poca luz que con mucha, ya que en algunas pruebas en lugares con muchos focos o donde da el sol directamente, los sensores no funcionan de la mejor manera posible. Esto es debido a los reflejos que se producen sobre el papel, si se usara un material más mata, esto no pasaría. Independientemente de esto, cada sensor en la parte inferior tiene un pequeño regulador que se puede usar para variar la sensibilidad del sensor, de esta manera, no es necesario cambiar el tipo de superficie en la que se trabaja.

3.4.2 Motores

El motor requiere más potencia eléctrica que un sensor, por tanto, la alimentación del propio Arduino ya no sirve, hay que introducir una batería de 9V. También, el motor no se puede controlar mediante los pines normales de un Arduino, es necesario conectarlo al controlador l298N que será el encargado de controlar los motores dc. Es necesario este controlador entre el Arduino y los motores ya que los motores funcionan con 9V y el output máximo de un pin de un Arduino tan solo es de 5V.

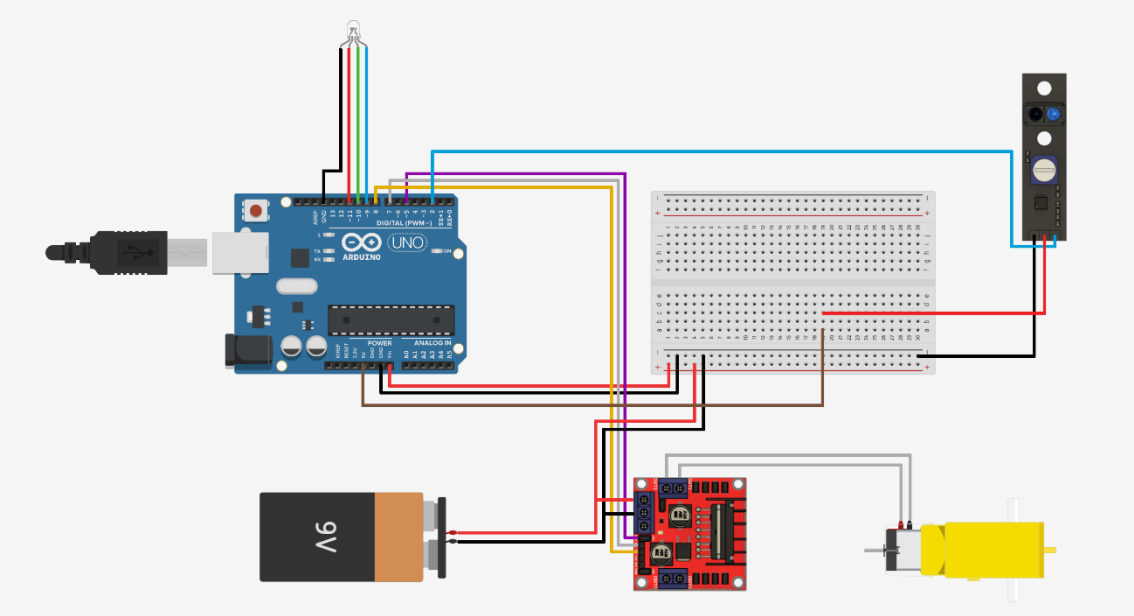


Ilustración 7, esquema eléctrico del prototipo con los motores

Al ser una prueba con bastantes elementos, usar una *protoboard* es útil para facilitar las conexiones. Vemos como la batería va conectada directamente al controlador que lleva el motor ya que, como mencionado antes, los motores necesitan bastante potencia. La batería también va conectada a las líneas de corriente de la *protoboard* para poderlo conectar con el pin VIM del Arduino (único pin que acepta 9V de corriente) y conectar ambas tomas de tierra ya que compartir la misma toma de tierra por todos los elementos es imprescindible en cualquier tipo de circuito eléctrico.

Por limpieza de cables, el led y los pines del controlador van directamente al Arduino, los únicos cables que pasan por la *protoboard* son los del sensor de línea ya que este se encuentra bastante alejado del resto de componentes.

A partir de este momento, el sistema es totalmente autónomo y no necesita estar conectado al PC, solo es necesario conectarlo para subir el programa.

El código para esta prueba se basa en el de la prueba anterior, solo que esta vez giramos el motor en diferentes direcciones según el output del sensor de línea. El pin EMA es el pin que permite el flujo de corriente, los otros dos (IN1 y IN2) marcan el sentido de la corriente que hace girar los motores hacia una determinada dirección

En la siguiente figura vemos el resultado real de esta prueba.

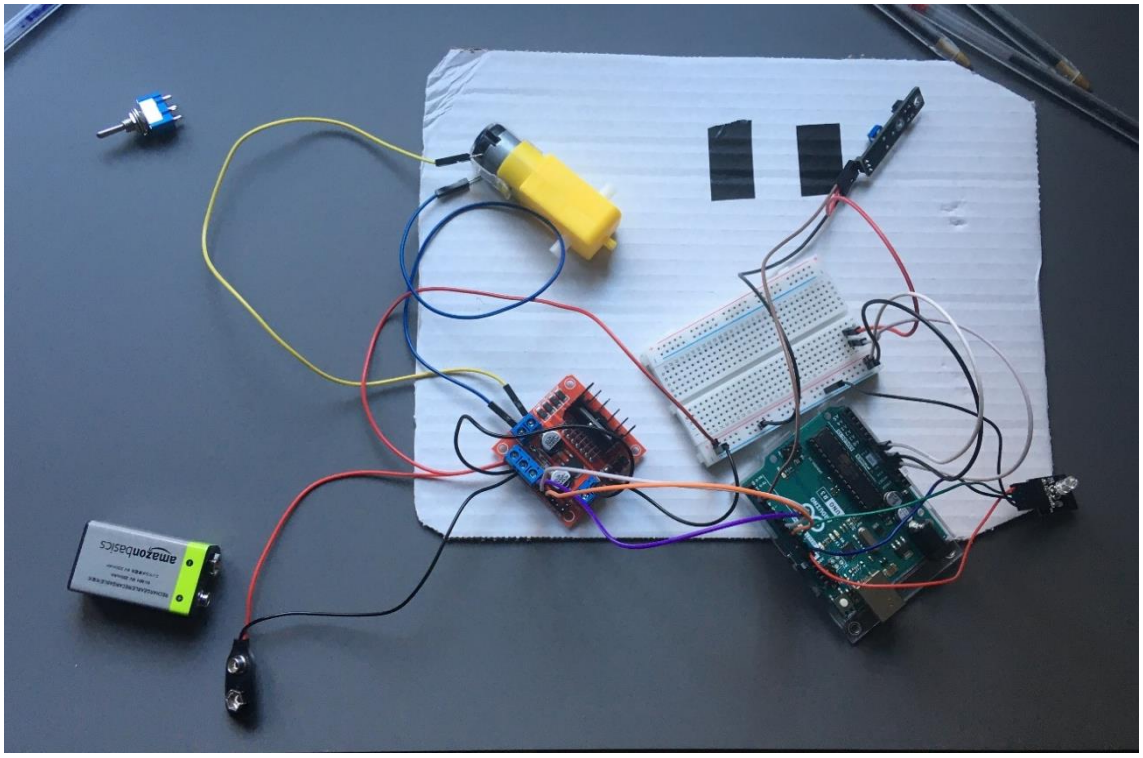


Ilustración 8, aspecto físico del prototipo del motor según los datos leídos por el sensor de línea

3.4.3 Bluetooth

La comunicación entre los robots y el servidor en el PC será mediante bluetooth, el problema viene que un ordenador con bluetooth pese tener la opción de tener diversos dispositivos bluetooth conectados simultáneamente, tan solo se puede tener un dispositivo de cada tiempo conectado. Esto se refiere a que tan solo se puede tener un dispositivo de auto conectado, por ejemplo. Por tanto, el PC (donde va a estar ejecutándose el simulador de MsPacMan) tiene que comunicarse mediante cable con un Arduino, el cual que tendrá 5 módulos bluetooth, cada uno emparejado con un robot. De aquí en adelante, este Arduino situado entre el Pc y los robots se le va a llamar el Arduino servidor.

De momento, la primera prueba de concepto se basará en dos Arduinos conectados entre sí. Uno será el maestro y el otro el esclavo. El maestro cada 200 milisegundos mandará un valor rotando entre 10, 20 y 30 constantemente. El esclavo recibirá este dato e irá cambiando el color de una luz led según el dato recibido.

A continuación, vemos el esquema eléctrico de esta prueba, primero el esquema del maestro (Ilustración 9) y después el del esclavo (Ilustración 10).

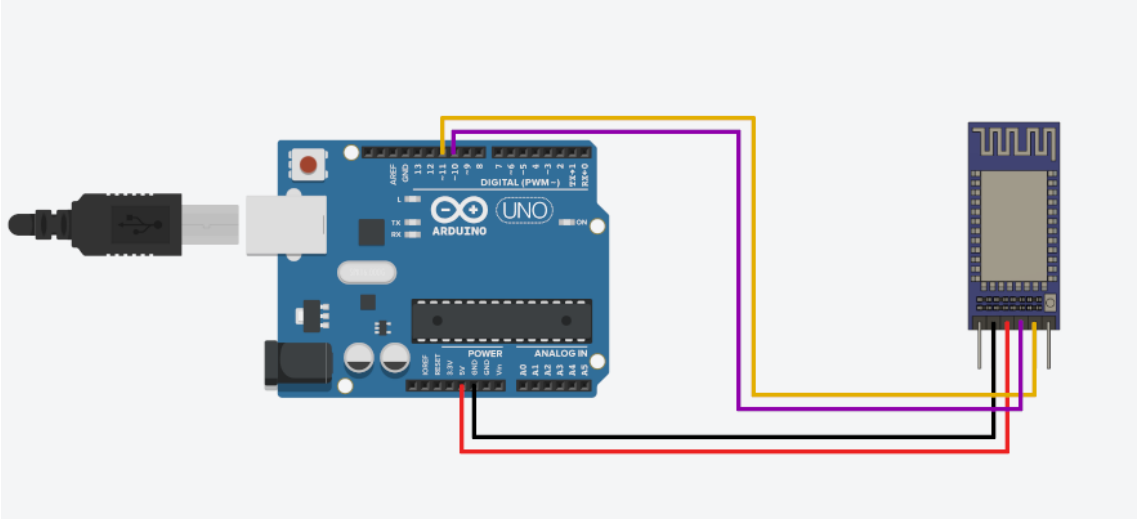


Ilustración 9, esquema eléctrico del maestro

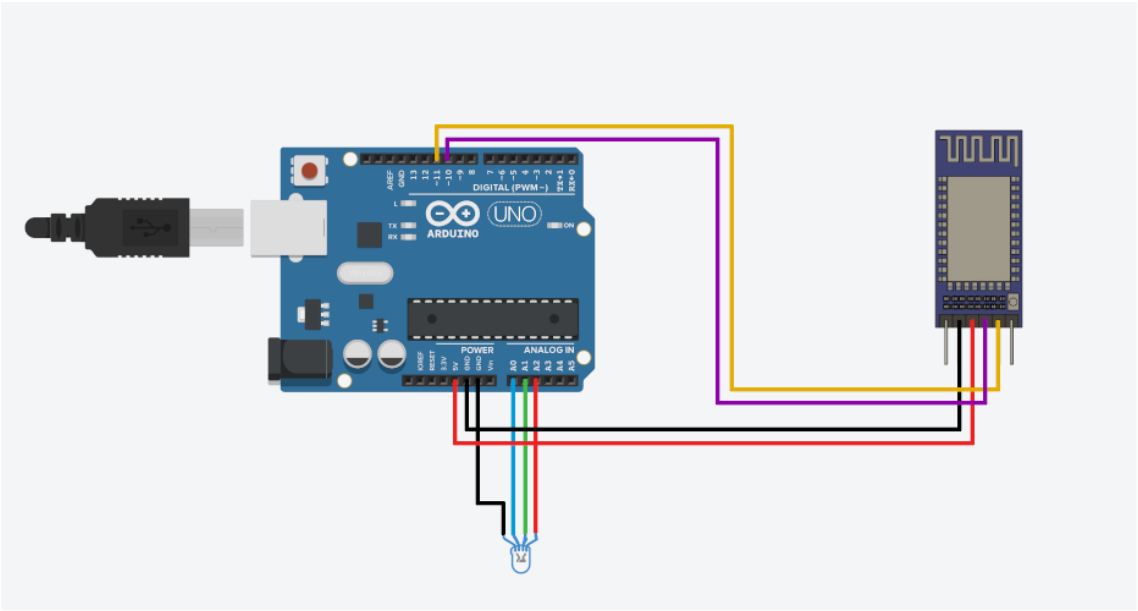


Ilustración 10, esquema eléctrico del esclavo

En este caso tenemos dos códigos, uno para cada Arduino, pero ambos son bastantes simples. Esta es la primera vez en la que se necesita una libraría, en concreto se usará *SoftwareSerial*[14], pero no es la librería que se usará en el robot final ya que se pasará a utilizar *EasyTransfer*[15] que permite pasar estructuras de datos sin tener que serializar y deserializar manualmente.

En la siguiente cadena de imágenes se ve como el led situado en la parte superior derecha de las imágenes va cambiando de color con los datos que se están mandando desde el otro Arduino.

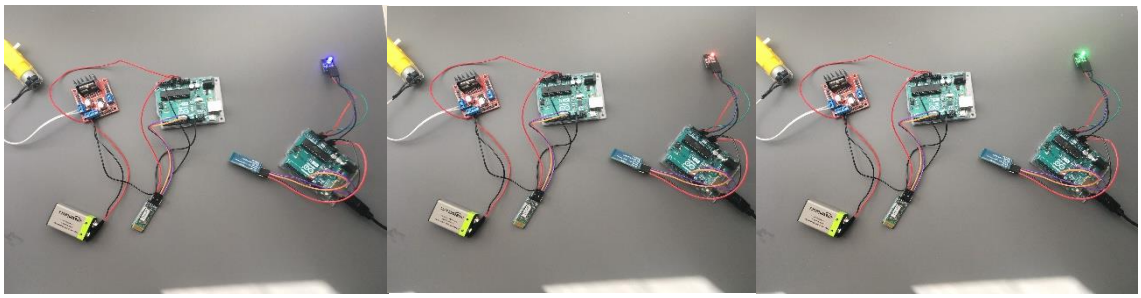


Ilustración 11, estados del led según llegan mensajes vía bluetooth

La siguiente prueba sería poder pasar estructuras de datos ya que para el robot final esta prueba no es suficiente, habrá que mandar estructuras de datos que contengan toda la información necesaria (dirección de movimiento, robot al cual le corresponde y posiblemente otros datos de sincronización).

Para ello hay tres formas de hacerlo, la primera es completamente manual, mandas una cadena larga de bytes y al recibirla, la troceas en las distintas variables que necesites. La segunda manera es mediante la creación de tramas, primero mandas un mensaje inicial indicando que se va a proceder al envió de datos, luego se mandan dichos datos, y finalmente se manda otro mensaje general indicando que el mensaje ha sido terminado (a efectos prácticos es como cuando se dice *cambio y corto* al final de un mensaje por la radio). Y finalmente, la tercera manera, y la elegida para empezar este proyecto, es usar la librería *EasyTransfer*, en concreto la versión *SoftEasyTransfer* la cual es capaz de enviar y recibir estructuras de datos.

El código es muy similar al del apartado anterior, solo que, en vez de tener un led para representar los datos, se hace por consola, así, podemos mostrar todos los campos de la estructura de datos.

En la Ilustración 12 vemos la salida. Se ve como el ciclo de datos ha empezado en 20, esto es porque aún no hay sincronización de tiempos entre los Arduinos ya que para esta prueba no hace falta, esto se incorporará a la hora de montar el primer prototipo.

```
Iniciamos
ID: 0, Data read: 20
ID: 0, Data read: 30
ID: 0, Data read: 10
```

Ilustración 12, salida por consola

3.5 Conclusiones

En este capítulo se ha presentado el entorno de simulación MsPacMan Engine sobre el que se basará toda la simulación del proyecto, además, también se estudia la lista de componentes que debería tener cada robot inicialmente y se realiza una pequeña prueba

a cada uno de ellos individualmente para entender su funcionamiento y así evitar errores más adelante cuando se empiece a prototipar con el robot definitivo.

Capítulo 3

Desarrollo del robot

"Puede ser un mal momento para mencionar esto, pero…
me debes 32.000 dólares en honorarios legales."
-Mate

3.1 Introducción

En esta sección, se repasan todas las iteraciones desde el prototipado inicial hasta el producto final analizando los pros y contras de cada iteración. El objetivo es obtener un prototipo con todas las capacidades requeridas funcionando por lo menos en un nivel básico sobre el que se podrá trabajar más adelante para perfeccionarlo con software.

3.2 Planteamiento inicial

Planteamiento

El proyecto entero estará controlado por una versión modificada del motor MsPacMan vs Ghost desarrollado por el grupo GAIA. Cada vez que se produce una acción dentro del juego (cuando un personaje llega a una intersección) enviará esta acción al servidor Arduino que ya se encargará de procesar la acción y reenviarla al robot correspondiente. Es decir, el juego como tal se corre en una máquina central que actúa como servidor, y los robots vienen siendo clientes "tontos" que van siguiendo las instrucciones motor pasando previamente por un servidor (por así llamarlo) que será el Arduino central con todos los módulos bluetooth.

Representación del mapa

El mapa del juego estará representado en el suelo mediante líneas negras, concretamente dos líneas. Cada una marcando las paredes de cada pasillo (Ilustración 13). Las intersecciones estarán representadas mediante un cuadrado negro que ocupa toda la intersección (Ilustración 14). Todo este cuadrado negro de cada intersección representa la zona de giro de los robots.

Ilustración 13, representación de un pasillo

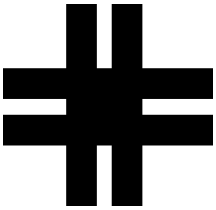


Ilustración 14, representación de una intersección

Con estas dos normas, el mapa completo quedará algo así:

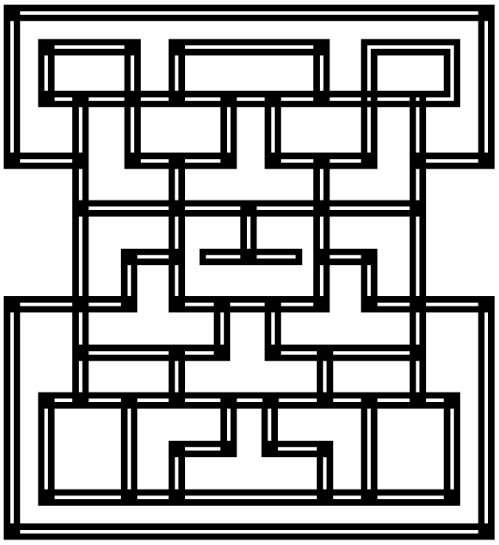


Ilustración 15, representación del mapa completo

Funcionamiento

Internamente habrá una pequeña máquina de estados, el robot puede estar en dos estados principales, recto y girando. Luego más adelante se explicarán los otros sub-estados más específicos como la salida de curva, las intersecciones rectas…

- **Recto:** estará en este estado cuando esté en un pasillo. Para asegurarse de que no se desvía, los sensores 1 y 3 tendrían que estar detectando una línea (Ilustración 16)

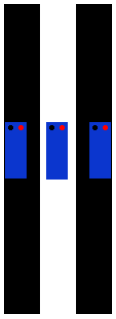


Ilustración 16, posicionamiento de los sensores en relación con las líneas

Mientras los dos sensores laterales estén viendo la línea y el central no, sabemos que estamos bien encaminados en la recta.

En cambio, si un sensor lateral ya no detecta la línea, pero el otro si, significa que el robot se ha movido de su línea, por tanto, tenemos que corregir un poco la dirección, en este caso, girando a la derecha levemente.

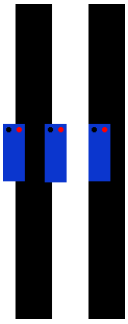


Ilustración 17, posicionamiento de los sensores en relación con las líneas

- **Giro:** estará en este estado cada vez que se llegue a una intersección. En este momento, ya el robot ya sabe (porque se lo ha comunicado el servidor) si tiene seguir en línea recto o realizar un giro de 90 grados. En el caso de tener que girar, tendrá que parar en seco y mover las ruedas de un lado en dirección apuestas a las del otro, obteniendo así un giro en forma de "tanque".

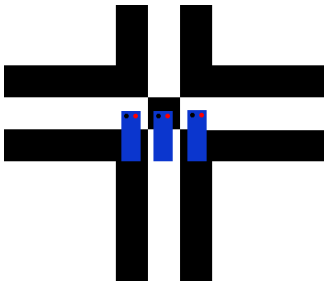


Ilustración 18, posicionamiento de los sensores en relación con las líneas

Cuando queremos girar, le mandamos la orden al robot, pero este no girará hasta que todos sus sensores detecten una línea como se ve en el ejemplo asegurándonos así que estamos girando donde queremos.

3.3 Primera iteración

Diseño del robot

La idea principal, es conseguir un robot lo más compacto posible para poder representar el mapa lo más pequeño posible. Lo primero de todo es conseguir modelos 3D de todos los componentes del proyecto para poder diseñar en 3D ya que permite una mayor precisión luego a la hora de diseñar e imprimir el chasis, ya que es mucho más fácil iterar.

Por suerte, hay miles de modelos online ya disponibles para uso libre que me puedo descargar. El Arduino, los motores, las ruedas, y los controladores estaban disponibles para descargar en la web oficial de Autodesk[16]. El resto de los elementos al no ser tan comunes, no están disponibles online, por tanto, se tuvieron que hacer los modelos 3D manualmente.

En la figura del apartado de componentes (Ilustración 3) podemos ver todas las piezas necesarias para el robot.

Una vez deseñado un chasis y todas las piezas colocadas, el robot queda con este aspecto:

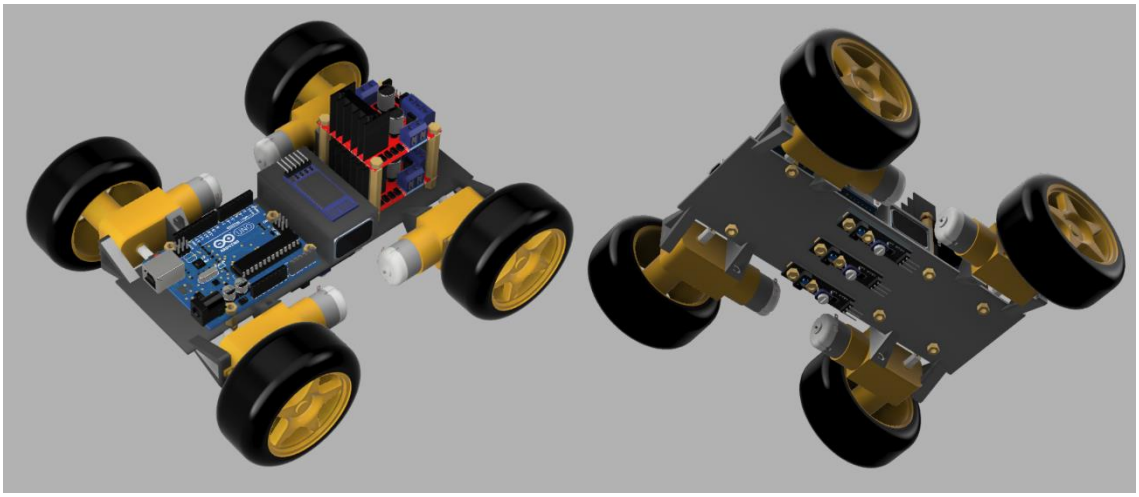


Ilustración 19, renders de la primera iteración del robot

Vemos en la parte inferior de la segunda figura que los sensores de línea están exactamente en el centro de los ejes del robot. Esto es para poder realizar giros sobre sí mismo precisamente.

El resto de los componentes están en la parte superior del chasis ya que no necesitan una posición exacta. Para ahorrar espacio, los dos controladores de los motores están montados uno encima del otro separados por unos espaciadores de nylon. Por el mismo motivo, la batería y el módulo bluetooth también están uno encima del otro, el módulo bluetooth siendo el de arriba para conseguir una mejor conectividad. Finalmente tenemos el Arduino con los puertos visibles desde el exterior para poder conectarlo al PC y realizar todos los cambios oportunos al software.

Autoevaluación

Esta primera iteración del robot sirvió como prueba de concepto sobre cómo se podrían organizar las piezas. Tras el análisis más detallado del concepto, se llegó a la conclusión de que como la distancia entre ejes de ese prototipo era de aproximadamente 13cm, la representación del mapa final acabaría siendo demasiado grande, unos cuatro metros.

Por tanto, la primera iteración no se llegó ni siquiera a imprimir en 3D. Había que intentar compactar los componentes aún más para reducir al máximo esa distancia entre ejes.

3.4 Segunda iteración

Diseño del robot

Tras volver a Fusion 360 a repensar el prototipo, se decidió colocar varias piezas de forma vertical, de esta manera, se reduce la distancia entre el eje delantero y el eje trasero, permitiéndonos reducir esta distancia de 13cm del prototipo anterior a aproximadamente 8cm (reduciendo considerablemente el tamaño final del mapa).

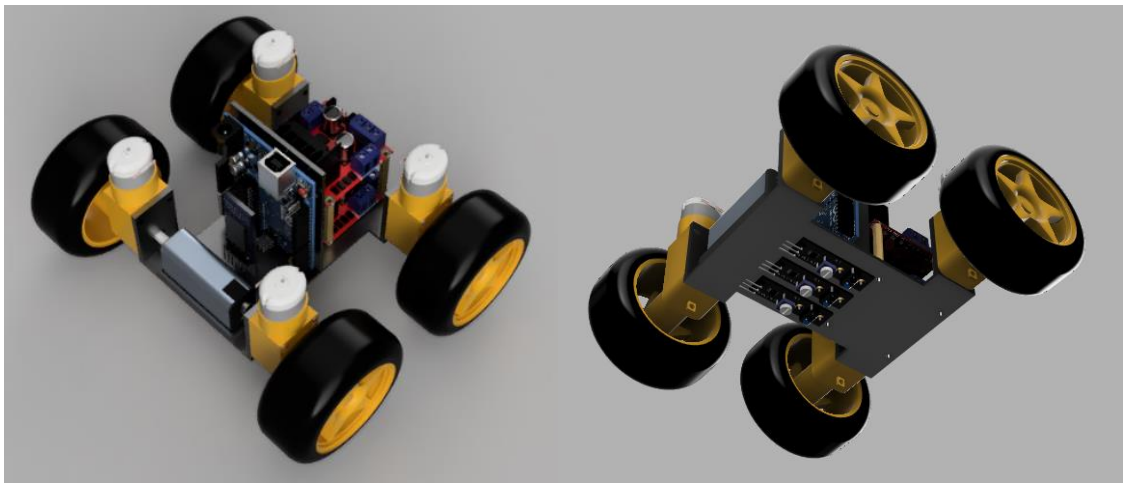


Ilustración 20, renders de la segunda iteración del robot

Como podemos ver en las figuras, el robot se ha compactado substancialmente en el eje longitudinal eje principalmente. El eje transversal a penas se ha reducido, pero esto no debería ser problema ya que el mapa está compuesto por pasillos, y estos pasillos entre si tienen bastante separación, así que si el eje transversal es más ancho que el longitudinal no importa tanto. Lo importante es que el eje longitudinal sea el corto, ya que es el que va a delimitar el tamaño del tablero.

Construcción

Tras imprimirlo en la impresora 3D, ya se podía empezar a montar los componentes dando el siguiente resultado:

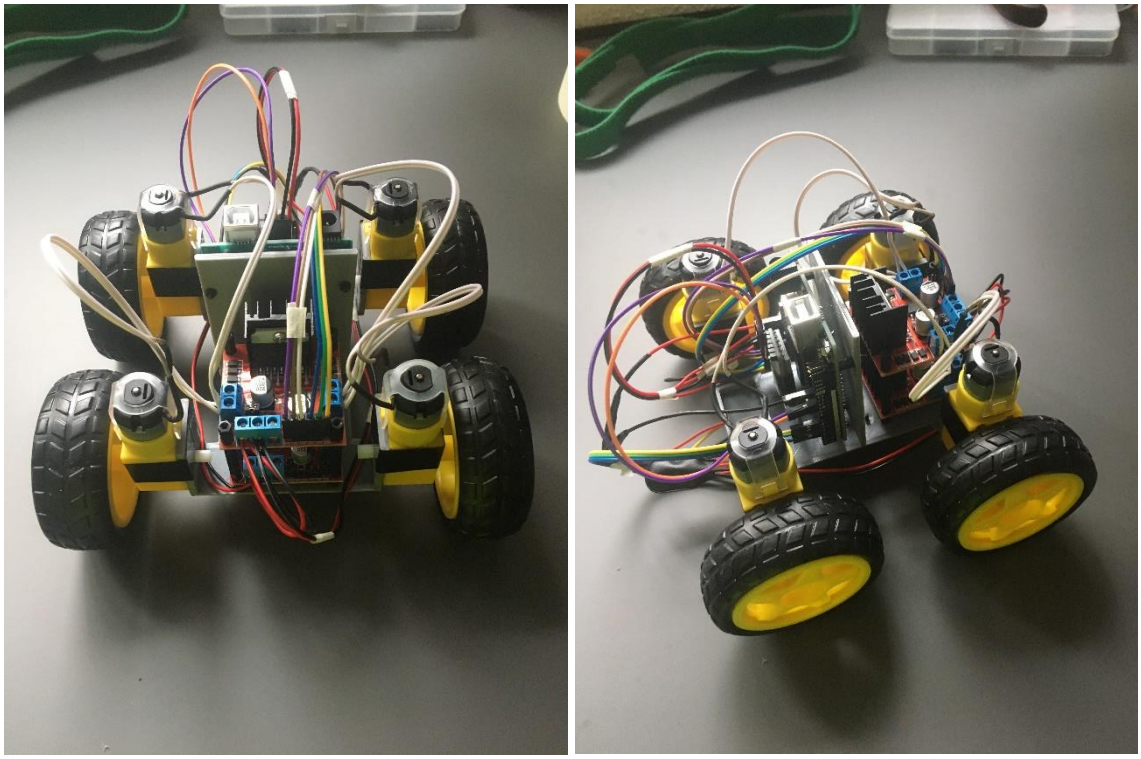


Ilustración 21, robot una vez construido y cableado

El esquema eléctrico quedaría como se muestra en la Ilustración 22.

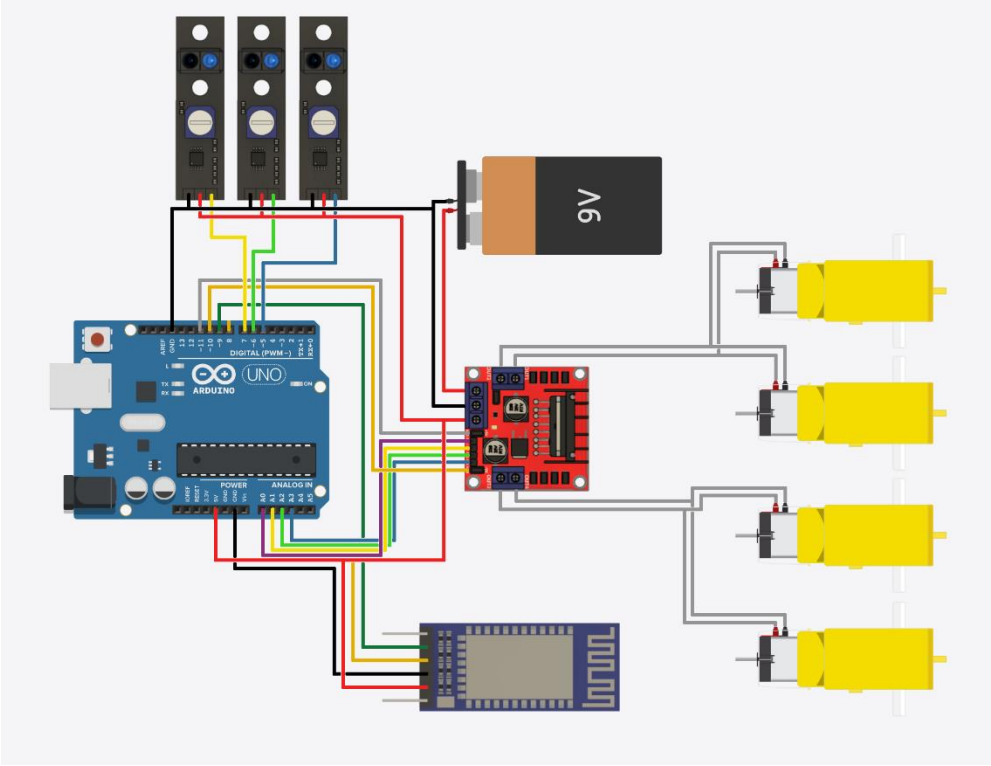


Ilustración 22, esquema eléctrico

Autoevaluación

Este prototipo llegó más lejos que el anterior, pero aun así tiene grandes fallos. Para empezar, el hueco para la batería no es lo suficientemente profundo como para sostenerla de manera estable con los movimientos bruscos del robot. Por suerte esto sería fácil de arreglar.

Otro problema es que cada controlador l298N es capaz de controlar dos salidas (que no es lo mismo que dos motores), por tanto, como se explica en el apartado de componentes del principio de la memoria, se necesitarían dos controladores para los 4 motores. Lo que no se había planteado era utilizar una sola salida para controlar los dos motores de un mismo lado ya que siempre van a tener el mismo comportamiento, o los dos van hacia delante, o los dos están parados, o los dos hacia atrás. Por tanto, solo se necesita un solo para dirigir el robot entero. Son este tipo de cosas sencillas que no se plantean hasta que se está construyendo de verdad y es precisamente por esto que esta primera parte del proyecto es un proceso altamente iterativo.

Para el siguiente prototipo, se reducirá el número de controladores l298N a tan solo uno, se hará un compartimento para la batería más estables, un chasis más rígido y ya que reducimos el número de componentes, se mirará de reducir el tamaño del robot aún más.

3.5 Tercera iteración

Diseño del robot

Para esta nueva iteración, con menos piezas, había que repensar la manera de organizar los componentes. Se decidió usar el espacio liberado por uno de los controladores I298N para colocar la batería rodeada por cuatro paredes para impedir que se moviera. De esta manera, en la parte delantera del robot quedaba un gran hueco, para recolocar el Arduino y el módulo bluetooth. La anchura del robot hasta ahora estaba delimitada por el tamaño del Arduino ya que estaba colocado de manera perpendicular al robot. Pero con el hueco que había dejado la batería, se podía colocar paralelamente permitiendo así mover los motores 30mm hacia el centro del eje compactando aún más el robot.

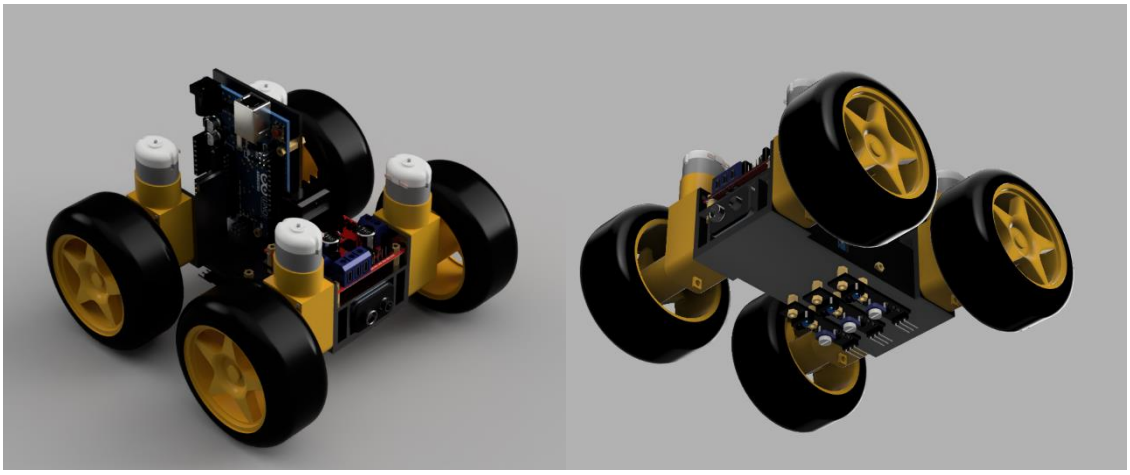


Ilustración 23, renders de la tercera iteración del robot

El eje vertical no se pudo reducir prácticamente nada, pero el eje horizontal si, en concreto esos 30mm conseguidos al mover los motores. Este eje no era el eje crítico que delimita el tamaño del tablero, pero, aun así, cuanto más parecidos de tamaño son los dos ejes, mejor girará y maniobrará además de ser más versátil.

Un problema de la iteración anterior era que el chasis no era lo suficientemente rígido y al moverse se tambaleaba bastante. Por suerte, esto tenía una solución sencilla, el grosor de la base del chasis se aumentó en un 50%, y para los soportes de los motores, se les añadió refuerzos en la parte superior ya que anteriormente solo estaban soportados por la parte inferior como podemos ver en las siguientes figuras marcado con flechas rojas:

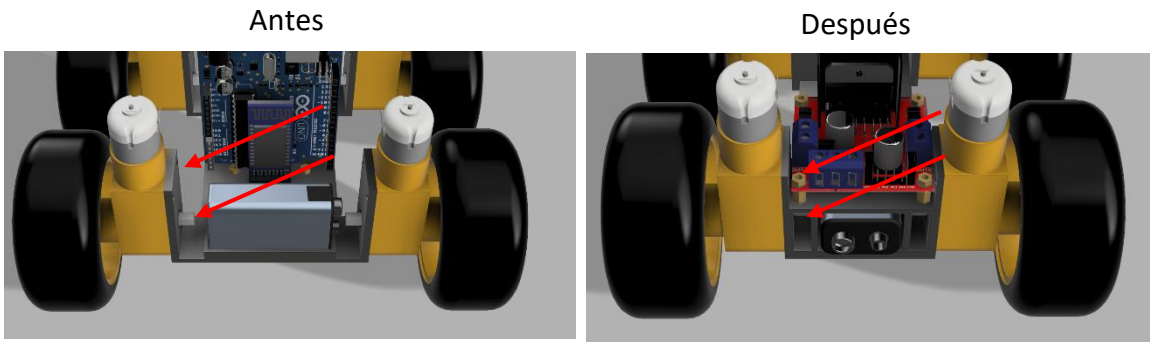


Ilustración 24, comparativa de la rigidez del chasis en distintas iteraciones

Construcción

Para este prototipo, los motores ya iban atornillados al chasis con tornillos m3 de 25mm haciéndolo mucho más estable e independiente de las cintas que había en la iteración anterior. Otra mejora fue las conexiones de los cables de los motores, hasta ahora iban conectados por contacto y termo retráctil por encima, pero a partir de ahora irían soldados con estaño.

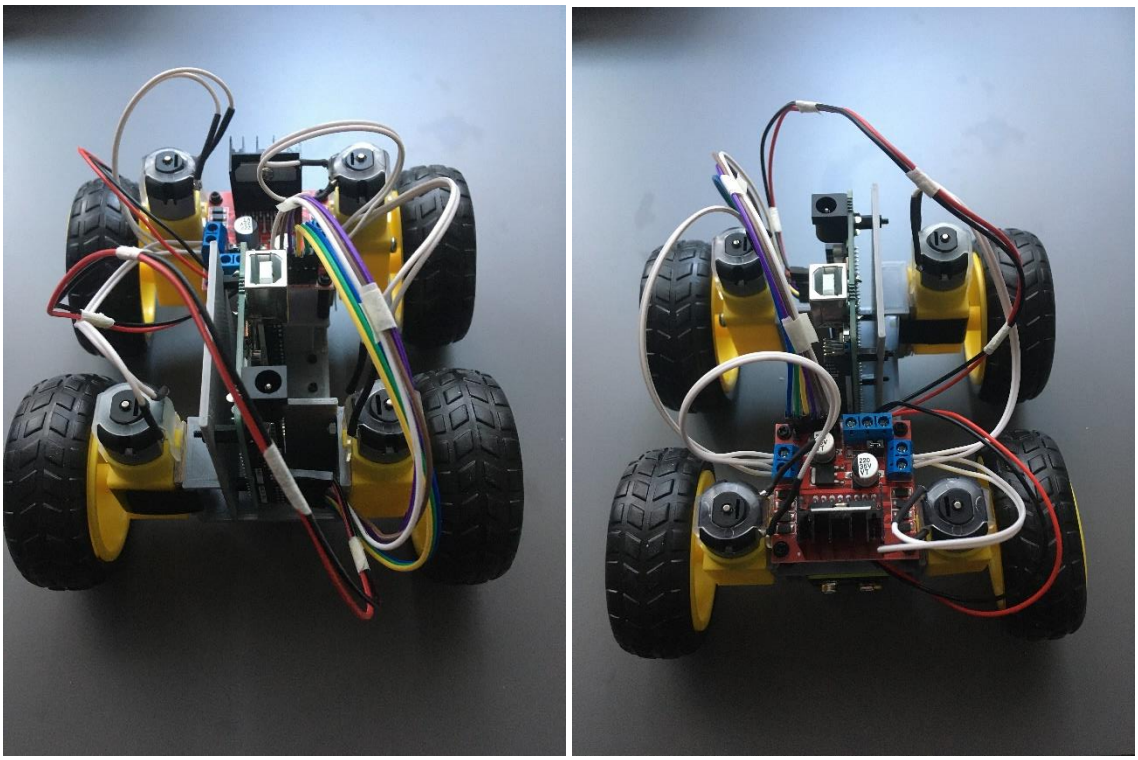


Ilustración 25, robot una vez construido y cableado

Pruebas

Este fue el primer prototipo el cual tenía algo de funcionalidad. No iba a ser el prototipo final ya que a la hora de construirlo ya surgieron ciertos errores que había que solucionar cara al futuro (tolerancias en agujeros, grosores de paredes…), pero aun así se podía empezar a hacer pruebas.

La primera prueba de todas era saber la velocidad real del robot. Este dato ya se podría haber sacado tan solo con las revoluciones del motor, pero al añadirle el peso del robot, estas se reducirían ligeramente. En línea recta, el robot iba a unos 40cm/s. Si escalásemos el mapa del juego al tamaño del robot, obtendríamos un mapa de unos 2.4m de anchura, lo cual hace que la velocidad de 40cm/s coincida casi a la perfección con la velocidad de MsPacMan en el juego original.

La segunda prueba era saber la velocidad de giro. Para ello, tan solo hay que hacer que los motores de un lado giren en dirección apuestas al del otro lado. De esta manera, un giro de 180 grados se realizaba en uno 450 milisegundos, y a consecuencia los giros de 90 grados en unos 230-235 milisegundos (no es exactamente la mitad por la aceleración inicial y final).

Ambas pruebas se realizaron en bucle, y al cabo de unos cuantos giros, el robot empezaba a desviarse, lo cual es totalmente normal. Por este motivo, están los sensores de línea, aunque para estas pruebas no se les ha dado implementación ya que había que diseñar el nuevo chasis.

Autoevaluación

El problema principal de esta iteración es que no se habían calculado bien las tolerancias. Había tornillos que no pasaban por los agujeros por los que tenían que pasar ya que en el diseño se hicieron muy justos y al imprimirlo no había suficiente margen como para pasar el tornillo. Otro problema que hubo fue la estabilidad del soporte para los motores traseros. En la segunda iteración se comenta que los soportes de los motores no eran lo suficientemente rígidos ya que tan solo se soportaban por la parte inferior. En esta iteración, los soportes traseros tenían un refuerzo en la mitad, pero aun así no era lo suficientemente estable. Por último, había lugares en los que las cabezas de los tornillos rozaban con algún componente, lo cual no afecta al rendimiento, pero no es la situación ideal.

Aun así, este chasis nos ha permitido sacar datos importantes como la velocidad en línea recta, las velocidades de giro, y el tamaño final que debería tener el mapa. De aquí en adelante ya se podría empezar a realizar pruebas unificando diversos componentes acercándonos al producto final.

3.6 Cuarta iteración

Diseño del robot

Esta fue la primera iteración de la cual realmente se pudo partir del chasis de la iteración anterior. Hasta ahora, los cambios a realizar eran tan drásticos que se tardaba menos empezando un diseño desde cero que modificar el diseño de la iteración anterior.

Para esta iteración, tan solo se aumentan distancias entre componentes para tener más hueco para los tornillos y cables. A consecuencia de esto, el chasis aumentó 7mm transversalmente (el eje longitudinal, se quedó intacto). Los márgenes añadidos fueron entre los motores delanteros y el Arduino, la distancia de ambos motores traseros con el controlador, y el hueco de la batería para que no entre tan forzadamente.

Estéticamente quedó prácticamente igual a la Ilustración 25 del apartado anterior ya que solo se modificaron pequeños márgenes.

Construcción

Esta iteración prometía ser algo más definitiva y por ese motivo ya se conectaron todos los componentes con intención de hacerlos todos funcionar dejando el robot con el siguiente aspecto:

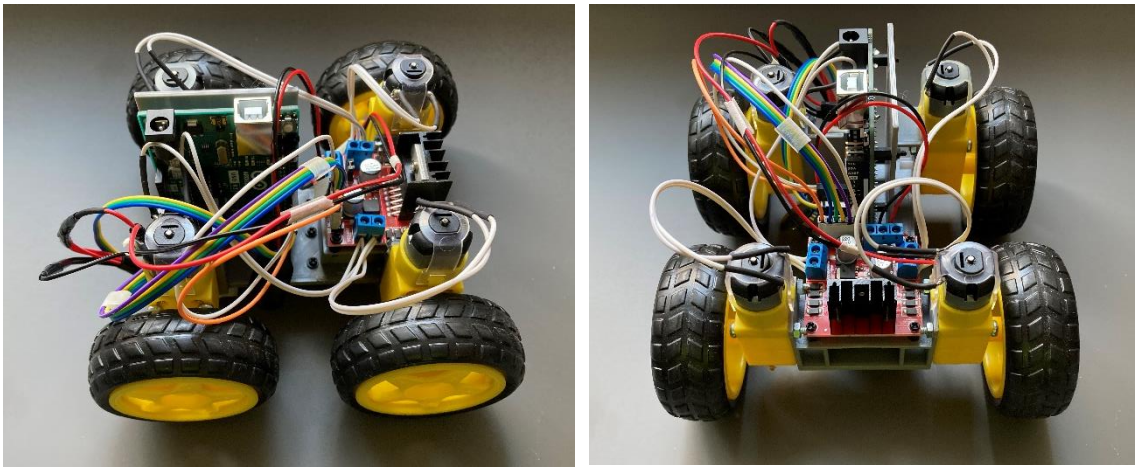


Ilustración 26, aspecto físico de la cuarta iteración del robot

Durante la construcción se detectaron más imperfecciones mínimas que se podrían corregir, pero ninguna de ellas era preocupante. El robot cumplía su función y solo se imprimiría otro chasis arreglando estas imperfecciones al final del proceso ya que no tendría sentido hacerlo en este punto sin saber aún si habría que realizar cambios más drásticos para añadir otro componente, sustituir tronillos por unos más largos…

Pruebas

Primera prueba: en la iteración anterior ya se hicieron pruebas a los motores, de esas pruebas se sacó la velocidad en línea recta y la velocidad de giro. Ahora se tenía que incorporar los sensores de línea. Para estas pruebas, se utilizó varios folios seguidos con líneas negras impreses simulando una recta del mapa.

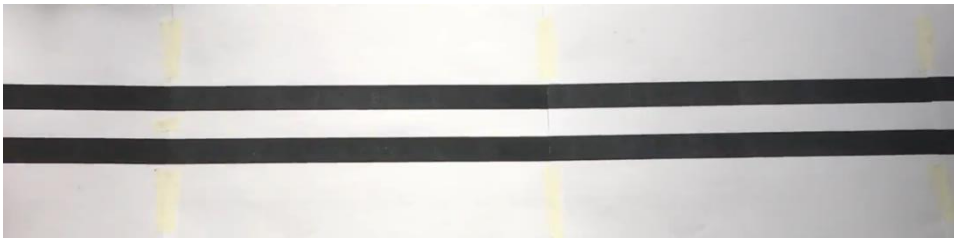


Ilustración 27, representación provisional del mapa mediante folios

El resultado obtenido fue bastante bueno, el robot reaccionaba bastante rápido al salirse de las líneas, pero, aun así, tardaba en frenar por la inercia que llevaba. Para este problema hay tres soluciones posibles, la primera es invertir el sentido de los motores para reducir la distancia de frenado, la segunda es mover los sensores ligeramente hacia la parte delantera del chasis para detectar antes el cambio de línea (dando tiempo para poder frenar situando al robot en el centro de donde se tendría que haber frenado), y la última es tener en cuenta esta distancia de frenado para empezar a realizar la acción justo antes de llegar al lugar en el que tiene que se tiene que llevar acabo.

De estas tres soluciones, la primera era la más prometedora. Tan solo habría que hacer cambios en el software y podría estar todo parametrizado. La distancia de frenado va con relación a la velocidad, por tanto, si en algún momento el robot va más lento a más rápido

que lo habitual, se puede variar este tiempo de frenado activamente sin tener que cambiar el chasis físico como habría que hacer en el caso de elegir la segunda opción.

Segunda prueba: en esta prueba había que mantener al robot en línea recta durante al menos un metro, en la prueba anterior el robot no giraba, en el momento en el que se desviaba un poco y perdía de vista a las líneas de seguimiento, se paraba en seco. Durante este metro, el robot se desviaría, pero con la información de los sensores, tendría que ser capaz de enderezar la dirección.

Esta prueba fue un poco más complicada al ya tener que incorporar comportamiento dependiente del estado cambiante de los sensores. La idea inicial era que para este comportamiento se usaran tan solo los sensores laterales. Recordemos el comportamiento de los sensores, si tanto el sensor derecho como el izquierdo estaban viendo una línea, es que íbamos en buena dirección. En cambio, si el sensor derecho dejaba de ver la línea y el izquierdo la seguía viendo, significaba que nos habíamos desviado un poco hacia la izquierda, por tanto, hay corregir la dirección ligeramente hacia la derecha.

En la Ilustración 28, vemos como quedaba una vez montado. El mapa está representado por varios folios con líneas impresas ya que las dimensiones de las líneas aun no eran definitivas y de esta manera era más fácil de prototipar.

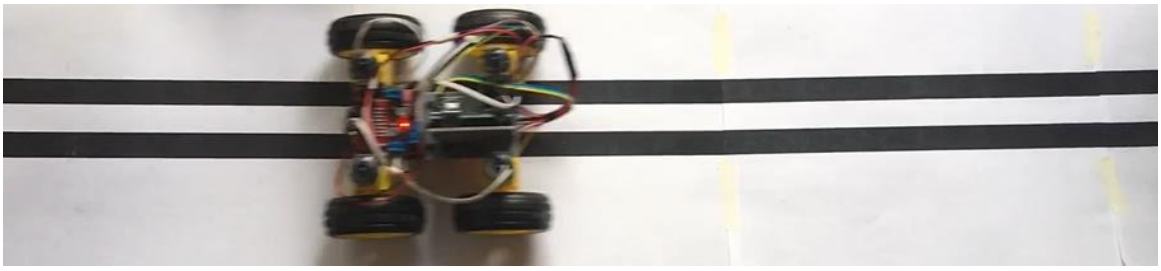


Ilustración 28, robot siguiendo líneas

El robot era capaz de recorrer cuatro folios manteniéndose por encima de las líneas negras, pero no era fiable, no todas las veces era capaz de recorrer la distancia entera. Este problema había que solucionarlo, el robot tendría que ser capaz de recorrer los metros que sean y siempre por encima de las líneas.

Tras unas pruebas más para investigar el origen del problema, se observó que las líneas negras no eran lo suficientemente gordas, había veces que dos sensores estaban sobre la franja blanca que divide a las líneas. Aumentar el grosor de las líneas negras reduciría el grosor de la franja blanca central y aumentaría el margen a cada lado.

Con este cambio y algún otro en el lado del software, el robot ya era capaz de recorrer los cuatro folios consistentemente y sin la necesidad de empezar perfectamente en paralelo con las líneas.

El método de autocorrección de la dirección es relativamente simple, una vez se detecta una anomalía (hay que corregir la dirección), se paraban los motores del lado hacia el que hay que corregir la dirección durante un instante (45 milisegundos). Durante este tiempo, el robot estaría girando. A continuación, se encienden todos los motores para

avanzar en línea recta durante otro instante (75 milisegundos) metiendo al robot de nuevo en el rango de las dos líneas.

Se utiliza esté método de girar un poco y avanzar para salir de la zona de “riesgo” ya que si tan solo girásemos el robot, como el giro realizado ha sido muy suave, los sensores seguirían detectando que estamos fuera de posición. Por tanto, se ha añadido una etapa de avance en línea recta para volver a entrar en el rango de las dos líneas.

En la Ilustración 29, se puede ver una representación simplificada del sistema de giro.

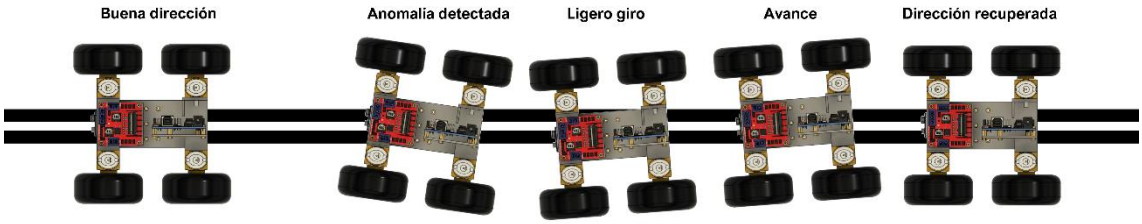


Ilustración 29, esquema del funcionamiento de la autocorrección de dirección

Este método daba resultados fiables en cuanto al seguimiento de línea, pero había un problema a la hora de hacer más pruebas. En el momento en el que el software se complicaba un poco más y se le pedía más funcionalidades, la batería no era suficiente para todo. Con los cuatro motores al 100%, los tres sensores, el Arduino y el controlador, la pila se quedaba corta.

Explorando soluciones para esto, se observó que se estaba controlando la velocidad de los motores de manera binaria, o encendidos al 100% o apagados. Este era el factor que causaba estos problemas. Se cambió el control a un control analógico para poder bajare la potencia a los motores. De esa manera, cada motor iba al 70% (aproximadamente) de su capacidad con lo que la batería ya era capaz de alimentar todo. No solo esto, sino que, gracias a este sistema, el mapa se podría hacer más pequeño ya que podemos ajustar la velocidad del robot para recorrer menos distancia. También, podemos refactorizar el código de giro para hacerlo más simple. Ahora, en vez de hacer lo mostrado en la Ilustración 29, se podría tan solo bajar la velocidad de los motores del lado hacia el que hay que corregir la dirección y queda un movimiento mucho más suave y un código mucho más limpio.

Tercera prueba: en esta prueba había que conseguir que el robot fuera capaz de realizar giros de 90 grados hacia una dirección.

Tras unas cuentas pruebas se consiguió que el robot girara más o menos al llegar a una intersección. El giro se conseguía al invertir el sentido de los motores durante un tiempo determinado para girar sobre sí mismo. Una vez el giro se había realizado, el robot avanzaría en línea recta unos 200 milisegundos para salir de la intersección, y volver al modo seguimiento de línea.

Esta prueba dejó un mal sabor de boca. Pese a que el robot era capaz de realizar giros de 90 grados, no siempre los podía hacer. Había ocasiones en las que el robot no entraba

perfectamente recto a la intersección y por tanto al salir, tampoco salía entre las dos líneas haciendo que no fuera capaz de seguir el recorrido. Otras veces, era que los motores no tenían la misma potencia a lo largo de la vida de la batería. Cuando la batería estaba recién cargada, el robot iba ligeramente más rápido, no mucho, pero lo suficiente como para girar más de 90 grados en el tiempo que debería haber girado solo 90. Al cabo de un tiempo, cuando la batería tenía poca carga, en ese mismo tiempo no le deba tiempo a llegar a 90 grados de rotación.

Estaba claro que poner el tiempo fijo de rotación no era algo fiable. Se decidió probar otras representaciones del mapa. En esta nueva representación, las intersecciones pasarían a ser más similares a las salidas de las autopistas en vida real. De este modo, la rotación no sería instantánea, sino que sería progresiva durante una distancia determinada. En la siguiente figura se puede observar cómo se intentó plantear el mapa.

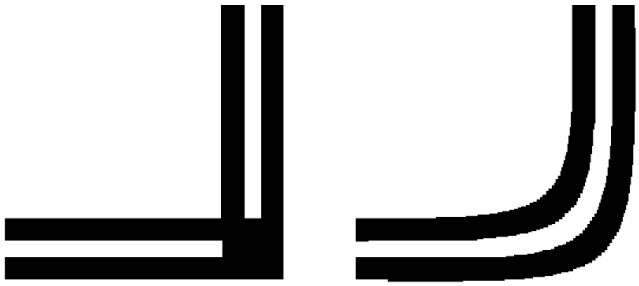


Ilustración 30, nueva representación de intersecciones

Este método al final se acabó descartando por el hecho de que seguía siendo un giro demasiado drástico. Con el planteamiento que se tenía del problema, si se quisiera usar esta representación, se tendría que multiplicar el tamaño del mapa mínimo por cuatro lo cual no era viable.

Autoevaluación

Al final , se llegó a la conclusión que tener preestablecido el tiempo de rotación no iba a ser una solución fiable. La solución más llamativa y prometedora era la incorporación de un giroscopio y utilizar los datos de este para saber cuánto tenemos que girar exactamente independientemente de la velocidad de los motores.

Sin lugar a duda, esta iteración sirvió para mucho, con ella se consiguió un prototipo capaz de seguir líneas prácticamente en cualquier ocasión y capaz de realizar giros, pero no lo suficientemente bien como para quedarse en el proyecto.

Para la siguiente iteración nos llevamos una representación de líneas definitiva, un seguimiento lineal sólido capaz de autocorregirse con movimientos suaves, y un sistema de detección de intersecciones. A continuación, se tendrá que incorporar un giroscopio y conseguir unos giros de 90 grados precisos.

3.7 Quinta iteración

Diseño del robot

Para esta iteración tan solo había que añadir un componente nuevo, por tanto, se podía partir relativamente fácilmente del chasis de la iteración anterior. El problema con el nuevo componente a añadir, el giroscopio, es que tiene que estar orientado y posicionado de una forma precisa para su correcto funcionamiento por tanto no se podía poner de forma vertical aprovechando algún hueco disponible.

Se tuvo que mover el controlador de los motores un poco hacia arriba (20mm) para dejar hueco al giroscopio entre el controlador y el chasis ya que este componente es bastante más pequeño que el controlador.

Al volver a rediseñar el chasis, se aprovechó para hacerle ligeras modificaciones como por ejemplo ensanchar el diámetro de los agujeros para los motores y añadir un nervio en el soporte de los motores traseros ya que podían llegar a flectar un poco al aplicarle peso.

En la Ilustración 31, podemos ver un plano cercano de los cambios realizados marcados con flechas rojas.

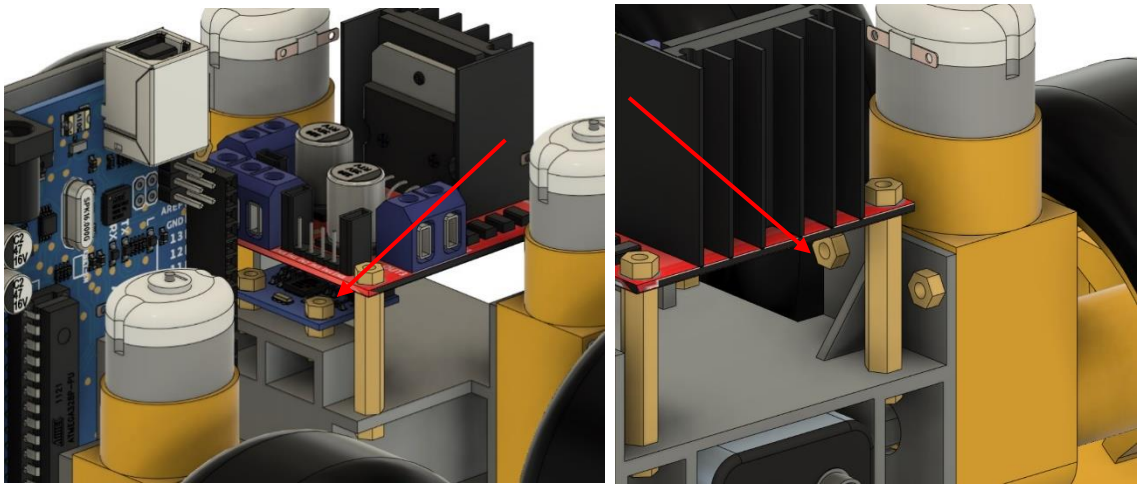


Ilustración 31, render de los nuevos cambios al chasis

Con la adición de un nuevo componente, el esquema eléctrico final queda como en la Ilustración 32.

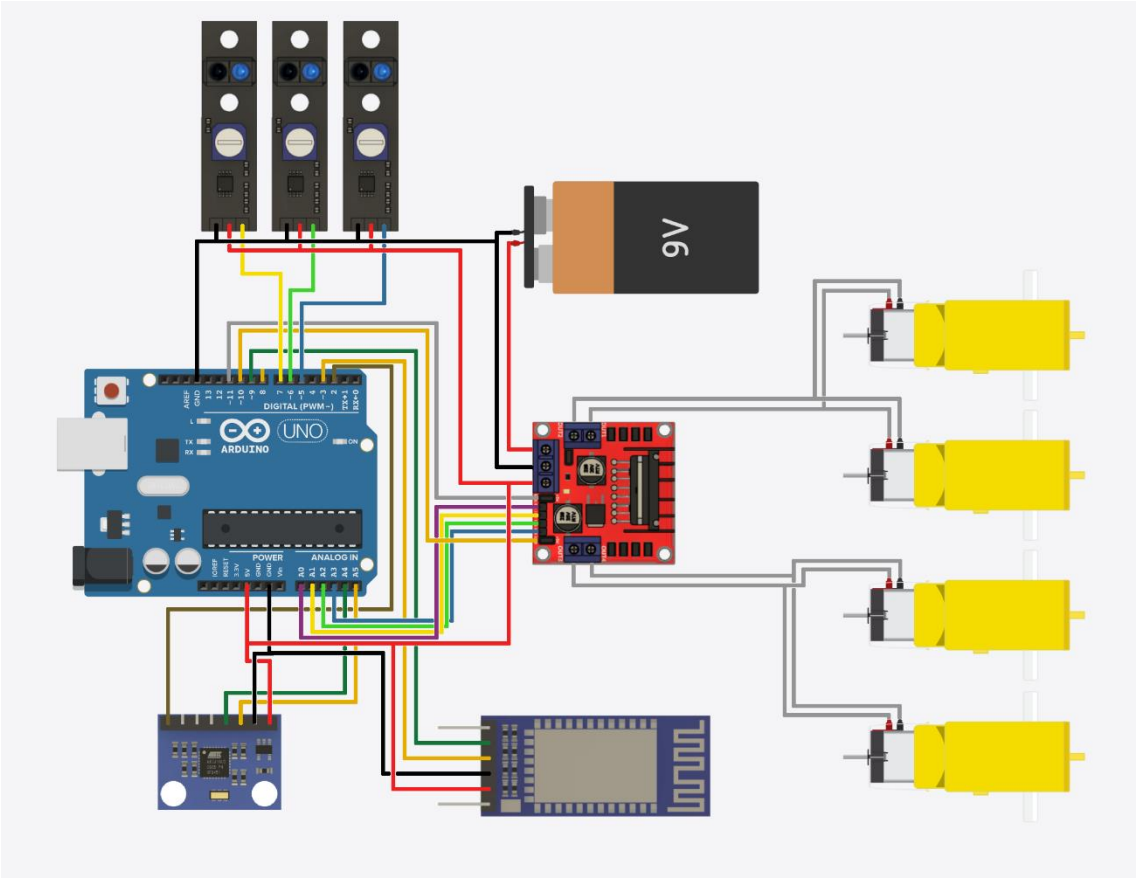


Ilustración 32, esquema eléctrico tras el giroscopio

Construcción

Por primera vez desde el inicio del proyecto, durante la construcción de esta iteración no se identificaron posibles mejoras, márgenes erróneos, lugares débiles… Dejando un robot con un aspecto similar a la iteración anterior pero mucho más pulido como podemos ver en las siguientes ilustraciones.

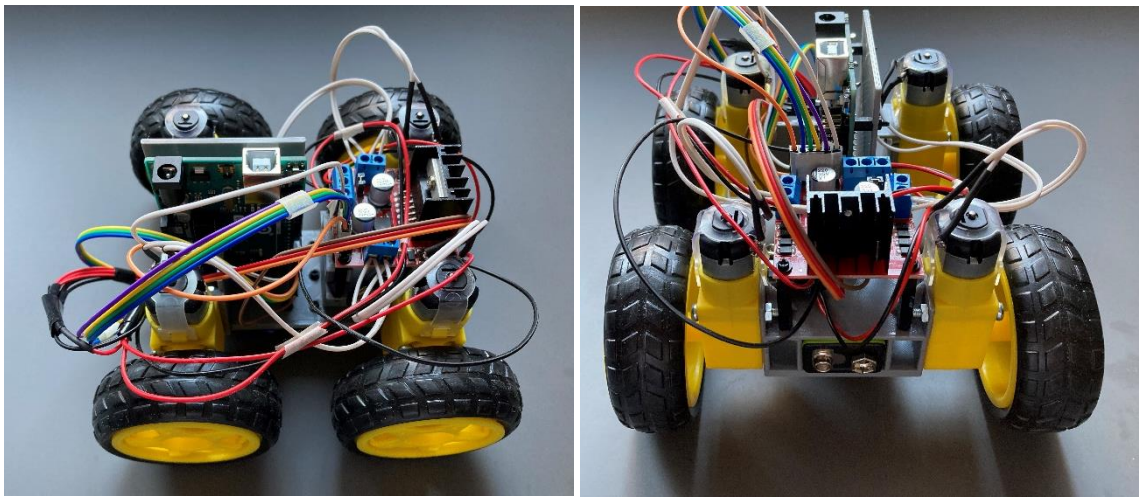


Ilustración 33, aspecto físico de la quinta iteración

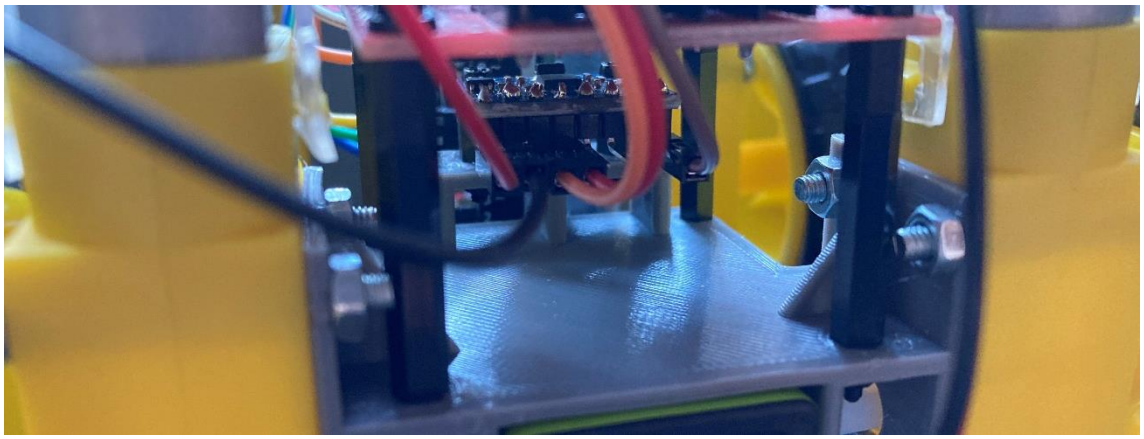


Ilustración 34, plano cercano del giroscopio recién añadido

Pruebas

Primera prueba: lo primero de todo es realizar una prueba con el giroscopio propia del apartado 3.1 de prototipado pero que no se hizo en su día ya que entonces se pensaba que este componente no era necesario.

La intención de la prueba era ser capaz de leer el *pitch*, *yaw* y *roll* del componente aun que realmente solo será necesario usar el *yaw*. Para este componente, se usarán librerías externas como *I2cdev*[17] y *Simple_MPU6050*[18] para facilitar el desarrollo ya que el giroscopio es capaz de enviar mucha información y se necesita deserializarla correctamente. Además, el uso de estas librerías abstrae al completo todos los cálculos relacionados a la conversión de ángulos de rotación por segundo (relacionado con la frecuencia del giroscopio) al *yaw*.

Esta prueba saca por el monitor serial los tres ejes del robot con bastante precisión como se puede ver en la Ilustración 35.

```
Yaw  -47.8091,   Pitch   -0.7276,   Roll    0.3607,
Yaw  -47.8090,   Pitch   -0.6935,   Roll    0.3686,
Yaw  -47.8413,   Pitch   -0.6650,   Roll    0.3638,
Yaw  -47.8576,   Pitch   -0.6642,   Roll    0.3483,
Yaw  -47.8618,   Pitch   -0.6578,   Roll    0.3455,
Yaw  -47.8737,   Pitch   -0.6862,   Roll    0.3505,
Yaw  -47.8781,   Pitch   -0.6890,   Roll    0.3441,
```

Ilustración 35, salida por el monitor serial

Segunda prueba: esta era la prueba importante ya que había que pasar del método anterior de giro (definido por un tiempo fijo) al nuevo sistema (definido por ángulos de rotación) para evitar el problema de la carga de batería (cuanto más cargada estaba, más giraba el robot).

En cada bucle da la lógica se ejecutan tres acciones principales, primero de todo se hace la lectura de los sensores de línea, a continuación, se hace la lectura de la rotación,

y por último se controlan los motores. Es importante mantener este orden ya que cada acción depende de la anterior.

Durante todo el tiempo en el que el robot está yendo en línea recta, el giroscopio está leyendo datos y calculando su ángulo de rotación medio. Cuando se habla de ángulo medio, se refiere a la dirección media que lleva el robot desde la última intersección. Esto es necesario ya que, al llegar a la próxima intersección, el robot podría estar justo corrigiendo su trayectoria y no haber entrado a la intersección perfectamente en perpendicular a ella, dando como resultado un giro incorrecto (Ilustración 36).

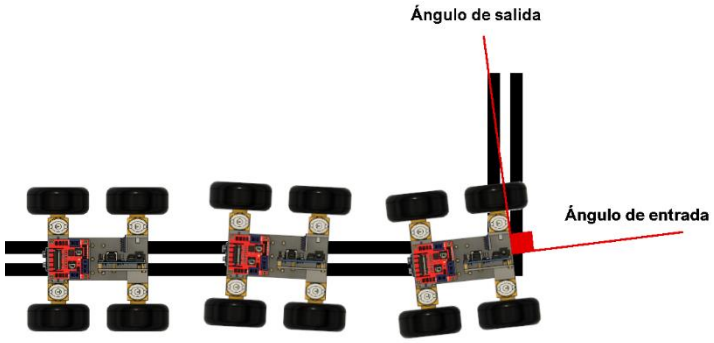


Ilustración 36, cálculo incorrecto del ángulo de salida

En cambio, si para calcular el ángulo de salida utilizamos el ángulo medio que se ha ido calculando a lo largo de la recta obtendremos un resultado como el siguiente ya que el ángulo de salida va determinado por el ángulo medio, no el de entrada:

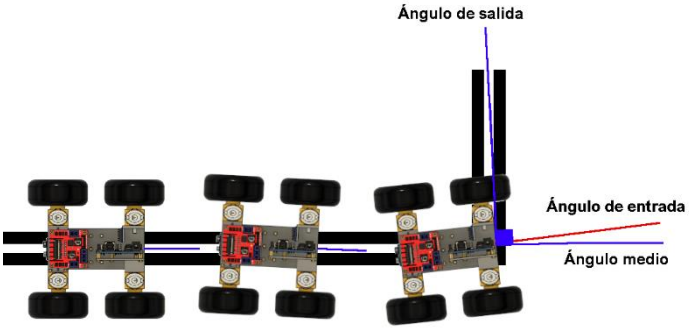


Ilustración 37, cálculo correcto del ángulo de salida

Sigue sin ser un resultado 100% preciso ya que el ángulo medio no será perfecto, pero es suficiente para que luego el propio robot con los sensores de línea sea capaz de auto corregirse en la dirección ideal dando unos resultados mucho más fiables que los que daría el método de la Ilustración 36.

Cuando el robot sale de una intersección, entra en un estado “de peligro” ya que corre el riesgo de perderse por si no sale de dicha intersección perfectamente paralelo a las líneas. Por ello, aunque el giroscopio es bastante precios, una vez se ha efectuado el giro, se vuelve a revisar si el robot está en la orientación correcta. Es necesaria esta

comprobación ya que el robot corta la corriente a los motores al llegar a 90 grados de giro (±2 grados), pero este aun lleva inercia y puede girar un poco más. Este ángulo de giro extra es extremadamente complicado de prever ya que no solo depende de los motores, puede depender también de si hay viento, de si justo esa parte del mapa tiene más fricción o menos, si ha derrapado una rueda… Por este motivo, justo al terminar un giro, durante los siguientes 250 milisegundos (como máximo), el robot se dirige con el giroscopio hasta alcanzar el rumbo óptimo, en ese momento, se vuelve a guiar por los sensores de línea.

Con este nuevo método, el robot ya era capaz de enlazar cuatro a cinco intersecciones seguidas. Es un gran avance, pero no es suficiente, el robot debería poder enlazar mínimo cincuenta intersecciones seguidas para poder jugar una parida de MsPacMan.

Autoevaluación

Hasta este momento, todo el trabajo se ha centrado en construir una buena base sobre la cual se irá trabajando en los próximos meses. En teoría la parte hardware ya estaría prácticamente terminada y ahora tan solo hay que centrase en la parte del software.

El robot ya es capaz de ir en línea recta y realizar unos pocos giros. Con toda la información de los sensores, cambiando la lógica del programa se debería poder conseguir un movimiento casi perfecto.

3.8 Conclusiones

Tras cinco iteraciones, se ha logrado un robot con la capacidad de guiarse en línea recta y realizar diversos giros seguidos en intersecciones. Durante estas iteraciones, se han explorado diversas soluciones al problema de los giros, llegando a la conclusión de que la mejor manera de realizarlos sería con la introducción de un giroscopio. Pese a aún no ser un prototipo lo suficientemente fiable como para ser el producto definitivo, ya era suficiente para empezar con el desarrollo de todos los componentes software del proyecto

Capítulo 4

Software y comunicación

"Esto es Rayo McQueen. Puedo manejar cualquier cosa."
-Rayo McQueen

4.1 Introducción

En el siguiente capitulo se analizará y desarrollará un sistema de comunicación bidireccional robusto que permita enlazar el robot con el MsPacMan Engine pasando por el servidor arduino. Este sistema tendrá que ser robusto permitiendo encender cada componente del sistema (PC, servidor y robot) en cualquier orden y seguir funcionando.

4.2 Comunicación Java-Arduino

Lo primero a tener en cuenta a la hora de comunicar los robots con el simulador es sacar los datos pertinentes del simulador, el cual está escrito en Java. Para la comunicación entre un proyecto java y un Arduino se usará una librería llamada jSerialComm[21] que permite mandar y recibir datos por el puerto serial del Arduino.

Tras haber inicializado la librería y haberla enlazado al Arduino que estuviera en ese momento conectado había que sincronizar ambos programas. Los programas escritos en java son bastante más rápidos en tiempo de ejecución que los escritos en Arduino, y pues el primer programa (escrito en java) tiene que esperar a recibir un mensaje del Arduino avisándole que ya está inicializado y listo para ejecutar instrucciones. A continuación, para la primera prueba, desde java se manda un mensaje para cambiar el color de una luz led conectada al Arduino. Finalmente, es importante cerrar el puerto serie para poderlo abrir cuando se vuelva a ejecutar el programa.

Con estos pasos, ya se tiene una comunicación bidireccional entre ambos programas. Cabe destacar, que cada vez que se manda un mensaje desde cualquier programa, es necesario hacer un *flush* para asegurarnos de que se han mandado todos los datos del monitor serial.

4.3 Comunicación Java-Arduino-Arduino

El siguiente paso es conectar vía bluetooth al Arduino servidor con el Arduino cliente (los robots).

Recordemos que es necesario tener un Arduino servidor entre los robots y el simulador ya que un ordenador normal tan solo es capaz de controlar un solo dispositivo bluetooth de este tipo (se pueden tener más de un dispositivo conectado al bluetooth, pero, por ejemplo, tan solo se puede tener conectado un dispositivo de audio como podría ser un altavoz). De este modo, tendremos conectado al PC un Arduino con cinco módulos bluetooth para tener cada uno de estos módulos conectados a un robot en particular.

Se empezó por la comunicación entre ambos Arduinos. Como pudimos ver al principio del proyecto, ya se realizaron pruebas de la comunicación entre dos Arduinos. En estas pruebas ya eran capaces de mandar estructuras de datos, pero tan solo en una dirección. De este modo, la comunicación era simulador → Arduino servidor→ Arduino cliente. Este método no era la suficientemente viable ya que para el proyecto final se necesitará una comunicación bidireccional.

Tras muchas pruebas, se averiguó que la librería que estaba utilizando (SoftEasyTransfer) no servía para comunicación bidireccional, por tanto, se creó un sistema de comunicación propio simplificado.

Este nuevo sistema propio tan solo es capaz de mandar *enums* entre dispositivos. Pero tan solo con esto, se podían mandar 256 tipos de mensajes distintos, los cuales eran más que de sobra para este proyecto. Por organización de código se dividieron los mensajes en dos *enums*, tenemos el *enum* con los mensajes de comunicación entre Arduinos y otro *enum* con los mensajes de comunicación entre arduino y java.

El siguiente paso era sincronizar los tres dispositivos. Esta tarea no fue sencilla ya que tenía que ser lo suficientemente robusta como para funcionar independientemente del orden de encendido de los dispositivos. Durante esta fase de sincronización, el arduino servidor se queda en espera hasta recibir el mensaje *SYNC_ATTEMP* del PC, una vez lo recibe, devuelve el mensaje *SYNC* indicando al PC que ambos dispositivos tienen una comunicación establecida y están sincronizados.

Una vez el arduino servidor y el PC están sincronizados, es hora de sincronizar el robot como tal. Siguiendo el mismo modelo que antes, el robot está esperando recibir el mensaje *SYNC_ATTEMP* del arduino servidor y una vez recibido, devuelve el mensaje *SYNC* al arduino servidor. El siguiente paso, es inicializar los componentes del robot. Como ahora se consta de un giroscopio, hay que esperar a que este se calibre, y no siempre tarda lo mismo. Por tanto, cada vez que el robot ha inicializado un componente, manda un mensaje al servidor avisando de qué componente se ha inicializad. A continuación, el robot no pasará a inicializar el siguiente componente hasta no recibir el mensaje *OK* del servidor indicando que se ha recibido el mensaje correctamente.

Para facilitar la depuración, el arduino servidor tiene una luz led rgb la cual indica el estado en el que se encuentra el sistema en ese instante.

- **Luz roja:** ningún dispositivo está sincronizado
- **Luz azul:** el arduino servidor y el PC están sincronizados y pueden intercambiar mensajes
- **Luz verde:** el robot y el arduino servidor están sincronizados. Solo es posible llegar a este estado habiendo pasado por el estado azul previamente, por tanto, si la luz es verde, se pueden mandar mensajes desde el PC al robot pasando por el servidor y viceversa.

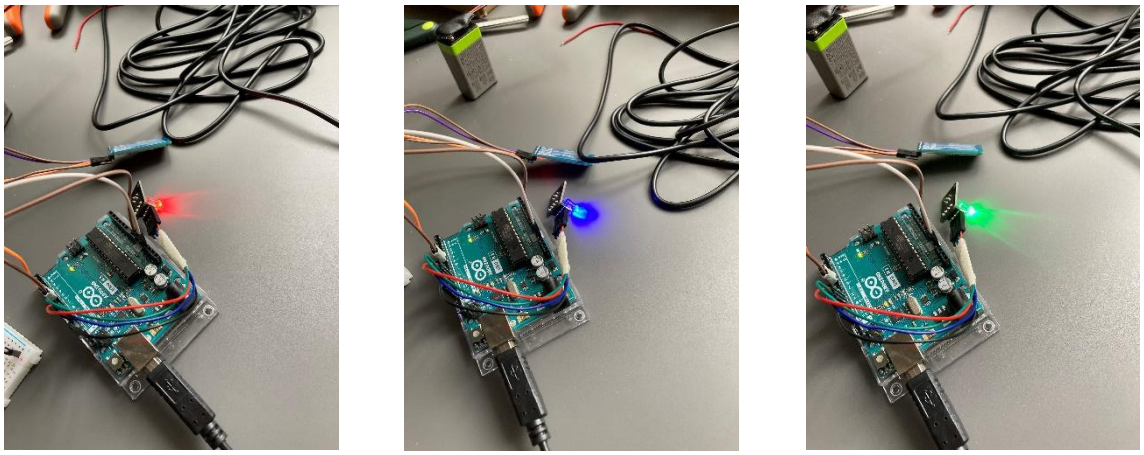


Ilustración 37, estados del arduino servidor según los dispositivos sincronizados

4.4 Integración del simulador MsPacMan

Recordemos que las partidas van a estar simuladas en un motor de MsPacMan vs Ghosts. Lo primero de todo y más importante, es extraer la información de dicho motor. La información que se necesita tan solo son las decisiones que ha tomado cada entidad (MsPacMan y lo fantasmas) en cada intersección.

En la Ilustración 38 vemos una representación de la organización general del software del proyecto. Vemos como el motor del MsPacMan es completamente independiente del resto del proyecto, de hecho, se puede seguir utilizando para cualquier otra utilidad. El otro proyecto “Simulador físico” es el encargado de la extracción de los datos, procesarlos y mandárselos al servidor Arduino el cual podrá reenviar a cada robot correspondiente, es el encargado de controlar la simulación de los robots físicos, de allí el nombre.

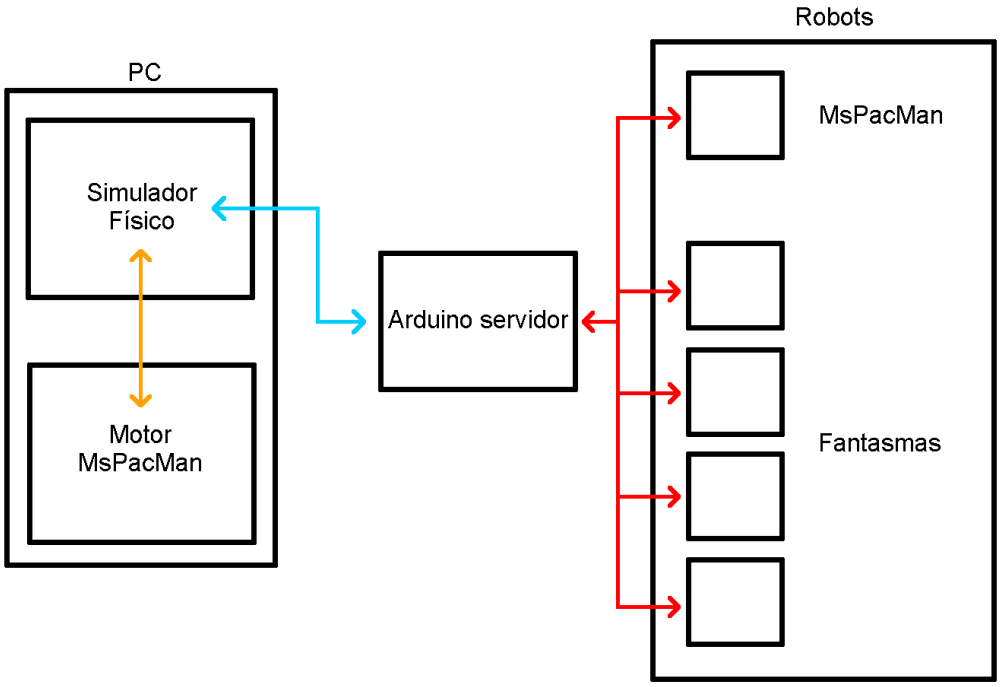


Ilustración 38, esquema de la organización de programas

Los únicos cambias que se han tenido que realizar al motor del MsPacMan es la opción de poder tomar como argumento un *observer* el cual se implementa desde fuera y es el encargado principal de la extracción de los datos.

Cada vez que se llega a una intersección, el *observer* registra la acción a la reenvía mediante el mensaje correspondiente al arduino servidor. Es necesario tener diversos *observers* ya que hay que diferenciar los mensajes mandados al arduino servidor según la entidad a la que va relacionada esa acción (MsPacMan o Ghosts).

4.5 Conclusiones

El sistema de comunicación resultante es francamente robusto, además, cumple todos los requisitos. Se ha conseguido que el sistema funciona independientemente del orden de encendido de los componentes y se ha creado con la intención y capacidad de soportar hasta cinco robots conectados simultáneamente vía Bluetooth.

Capítulo 5

Evaluación funcional

"Cometí un error. Pero puedo asegurarte que no volverá a suceder."
-Rayo McQueen

5.1 Introducción

En los siguientes apartados se perfeccionará el funcionamiento del robot para obtener el comportamiento deseado y conseguir poder completar recorridos complejos sin perderse, se perfeccionará la representación del mapa, y se analizarán y resolverán diversos factores externos que afectan al comportamiento del robot.

5.2 Sistema de control de giro

Hasta este punto, todo el desarrollo se ha estado haciendo sobre unos folios con líneas impresas creando un cuadrado. Estos folios daban mucha flexibilidad a la hora de crear algún recorrido concreto, pero tras unas pruebas se observó que teniendo las líneas negras de 25mm de ancho a veces no daban suficiente margen al robot para enderezar su dirección. Por este motivo, se tuvo que modificar el mapa final aumentando el grosor de las líneas en 5mm lo cual ya debería ser suficiente. Al tener todas las medidas del mapa ya establecidas, se podía mandar a imprimir para poder seguir desarrollando el robot sobre el mapa final que siempre dará resultados más fieles a lo que será el resultado final. Tras unos problemas técnicos a la hora de imprimir, el mapa se acabó retrasando unas semanas afectando así ligeramente los plazos siguientes.

El mapa al ser de grandes dimensiones se dividió en seis trozos para imprimir en folios A0. Como para el desarrollo inicial no es necesario tener el mapa entero, todas las pruebas de aquí en adelante se realizarán sobre una pequeña parte del mapa, concretamente la parte que representa la esquina inferior derecha.

En la Ilustración 39 podemos ver el aspecto que tiene cada parte del mapa. Vemos como hay un carril libre de intersecciones por la parte derecha e inferior que no existe en el mapa del juego. Este camino, será el encargado de conectar uno de los túneles que hay en el juego a los laterales del mapa. En el juego cuando un personaje pasa por uno de los túneles laterales, reaparece en el otro lado del mapa. Como en la vida real esto no es posible, obviamente, hay que conectar ambos extremos de los túneles mediante un camino (mientras un robot está en el camino, el resto de los robots se pararán en seco hasta que el robot en el camino llegue al otro extremo).

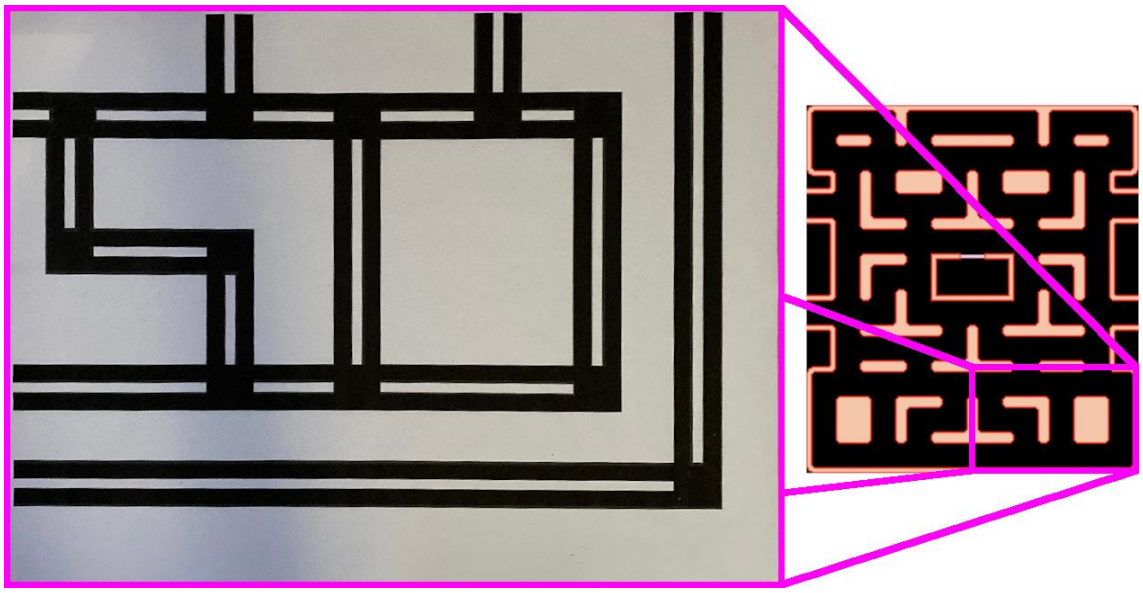


Ilustración 39, representación del mapa

Una vez se tiene el mapa, se desarrolló un sistema de dirección. Este sistema era el encargado de leer los mensajes direccionales provenientes del servidor Arduino, guardarlo en una cola y ponerlo a la disposición del resto del robot para que vaya extrayendo órdenes a medida que va llegando a intersecciones. Aunque esta lógica estaba implementada, faltaba la parte del servidor, por tanto, se dejó a un lado y se decidió establecer una ruta fija por el mapa para poder perfeccionar el movimiento y no tener problemas más adelante.

El primer problema que surgió fue que, hasta el momento, todas las pruebas se habían realizado sobre un rectángulo con rectas de aproximadamente un metro, ahora, había lugares en los que había dos intersecciones en tan solo 25cm. Esto causó un problema ya que el robot no era lo suficientemente preciso como para auto corregir su dirección en apenas estos 25cm.

Por tanto, durante las próximas semanas había que perfeccionar el giro lo suficiente como para que el robot sea capaz de realizar giros lo suficientemente precisos como para poder estar perfectamente colocados para poder realizar otro giro en apenas 20cm.

Esta tarea seguramente fuese una de las más complicadas hasta el momento, no solo porque hay infinidad de factores físicos externos que afectan (que un engranaje se quede ligeramente pillado, una pérdida de tracción en cualquier rueda…), sino que también no es posible depurar el programa corriendo en el Arduino de manera sencilla. Ni siquiera un Arduino conectado directamente al ordenador es sencillo de depurar. Durante todo el

desarrollo, mi método de depuración era imprimir mensajes por el monitor serial del propio Arduino el cual se puede ver desde el ordenador. Pero el robot, al no estar conectado a ningún ordenador, no se podía depurar de esta manera, así que se decidió crear un sistema de depuración propio vía bluetooth. Suena complicado, pero realmente tan solo es una luz led rgb en el Arduino servidor controlada por el robot. De este modo, desde el robot se puede cambiar el color de dicho led indicando por qué parte del programa está el robot en cada momento.

Con este sistema de "depuración", se consiguió arreglar algunos errores que había en el controlador de direcciones en el que había veces que no daba las indicaciones correctamente. Pero una vez más, aunque cada vez los giros eran más precisos, no era suficiente. El robot seguía acumulando error con cada intersección y al final se acababa perdiendo.

Tras analizar varias pruebas, se sacó la conclusión de que uno de los problemas era que en cada intersección se entraba con una velocidad ligeramente distinta. La velocidad

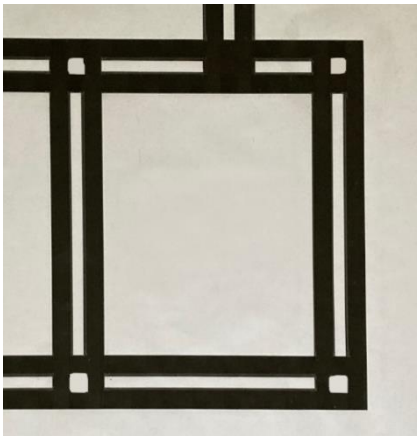


Ilustración 40, ligero cambio del mapa introduciendo una marca blanca en cada intersección

afecta porque cuando se detecta una intersección, el robot tiene que frenar, y según su velocidad podía pararse antes o después. Cuando había suficiente velocidad para acelerar y llegar a la velocidad de "crucero", la siguiente intersección no suponía ningún problema ya que, al frenar, el centro de los ejes quedaba exactamente encima de la intersección. El problema surgía cuando no había suficiente distancia entre giros y el robot aún no había acelerado suficiente. Lo primero que se introdujo fue una aceleración mayor los primeros instantes después de una intersección. De esta manera, al salir de un giro se recuperaría la velocidad de "crucero" más rápido. Esto una vez más, ayudó a la precisión de los giros, pero no fue suficiente. Una ayuda más era hacer un pequeño cambio al mapa para indicar a los sensores de línea mediante un punto blanco donde tendría que estar colocado el robot para realizar el giro.

Pese no haberla comentado antes, esta idea ya fue considerada hace meses en el inicio del proyecto, pero fue descartada. Tras unas cuantas pruebas, la idea se tuvo que descartar de nuevo ya que los sensores no eran capaces de detectar la intersección e instantes más tarde detectar el centro del giro. Cuando el robot iba a máxima velocidad, pasaba tan rápido por la zona de detección de la intersección que los sensores ni la detectaban.

Con pocas soluciones restantes, se decidió aceptar que los giros no siempre se iban a realizar exactamente en el centro de la intersección y se intentaría arreglar esa imprecisión en los instantes posteriores al giro mediante el refinado del seguimiento de línea.

5.3 Refinado del seguimiento de línea

Tras los problemas del giro del apartado anterior, surgieron otros problemas relacionados con el seguimiento de las líneas. Tras varias intersecciones, se iba acumulando un error el cual no afectaba directamente al giro ya que este está controlado por el giroscopio y es indiferente el ángulo de entrada al. Pero este error si afectaba, y bastante, al seguimiento de línea, ya que si el robot empezaba a seguir la línea y por algún motivo llevaba una dirección sub-óptima (en vez de girar 90 grados, giraba 95) en determinados casos, no tenía la suficiente distancia ni formación como para autocorregirse.

Por este motivo, era necesaria la coexistencia del giroscopio también cuando se va en línea recta. De normal el control principal recae sobre los sensores de línea, pero el giroscopio es capaz de quitarles este control a los sensores y tomarlo el mismo en los casos en los que se detecta que el robot se está desviando demasiado. De norma general, cuando el giroscopio detecta que se está desviando demasiado hacia la derecha, por ejemplo, los sensores de línea ya están intentando corregir la dirección girando ligeramente a la izquierda. El problema es que en determinados casos los sensores de línea no giran lo suficiente como para recuperar el rumbo lo suficientemente rápido, por tanto, lo que hace el giroscopio en estos casos es realizar un giro bastante más brusco. En la Ilustración 41 podemos ver este comportamiento. La línea roja representa el radio de giro que tiene el robot tan solo con los datos de los sensores de línea. Podemos ver como aun que se esté corrigiendo la dirección, no es suficiente y el robot acabará saliéndose completamente del recorrido. La línea verde representa el ángulo de giro cuando el giroscopio detecta un rumbo demasiado impreciso. Vemos como si es capaz de realizar el giro sin salirse completamente del recorrido.

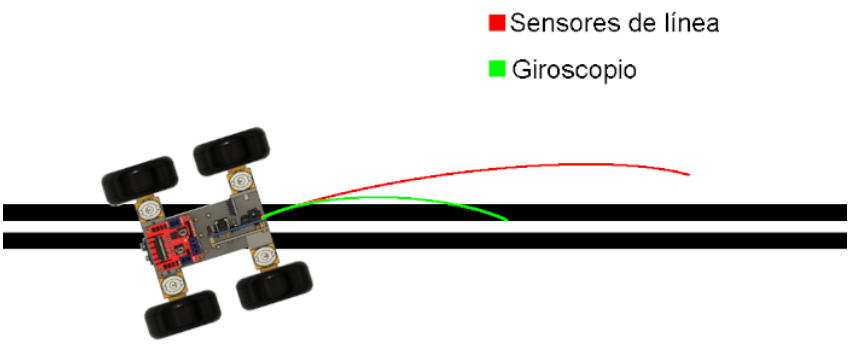


Ilustración 41, esquema comparativo del giro con los sensores de línea y el giroscopio

Lo siguiente era dotar al robot la capacidad de seguir en línea recta al llegar a una intersección en la cual no se requería girar. Hasta el momento, cada vez que se llegaba a una intersección, se giraba necesariamente, pero hay en ciertas intersecciones que el robot seguirá recto. En la Ilustración 40 vemos como en la parte superior hay una intersección con dos opciones, si el robot viene por alguno de los laterales, puede subir o bien seguir recto. Esto podría parecer una tarea sencilla a priori, pero recordemos que cuando se va en línea recta, el robot se guía por las líneas en el suelo, y en una intersección no hay líneas, todo es negro. Por tanto, desde que se detecta que se ha entrado en una intersección en la que no se requiere girar hasta que se detecta que se ha salido de dicha intersección (o ha pasado un mínimo de tiempo), los motores irán dirigidos tan solo por los datos leídos del giroscopio.

5.4 Sensibilidad a la iluminación

La primera vez que se sacó el proyecto del entorno de desarrollo se detectó un gran problema. El robot en otros entornos que no sea en el que se desarrollaron las pruebas no funcionaba para nada, no era capaz ni de seguir una línea recta. Al final se llegó a la conclusión de que la iluminación tenía un gran afecto en los sensores. En el entorno controlado había poca iluminación, y al ponerlo en un lugar repleto de focos, los sensores eran mucho más sensibles haciendo que el resto del programa no funcionara.

Que los sensores sean sensibles en teoría es buena señal, serían capaces de detectar mejor las líneas, pero toda la lógica estaba desarrollada sin tener toda esta precisión en cuanta. Por suerte, desde el principio del desarrollo, todos los valores relacionados con la calibración se fueron parametrizando y recalibrar el robot era cuestión de ir tocando estos valores. El aspecto negativo es que costaba mucho recalibrar estos valores y nunca se llegaba al nivel de precisión que se tenía en el entorno de desarrollo.

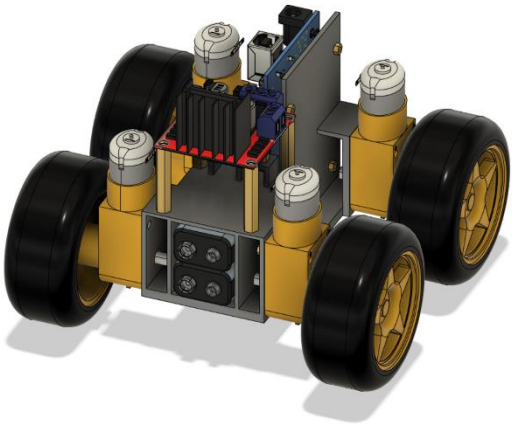
Una solución fue reducir la velocidad de los motores ligeramente. Esto ayudó en las rectas, pero no mucho, de hecho, trajo más problemas de los que arreglaba. Al tener menos velocidad, también tenían menos potencia lo cual en los momentos en los que se tenía que acelerar causaba que algunas veces los motores de un lado se quedaban quietos ya que no tenían suficiente potencia como para empezar el movimiento. La única solución que hubo a este problema fue volver a aumentar la velocidad y arreglar los otros problemas mediante la calibración del resto de parámetros incluyendo la sensibilidad de los sensores.

5.5 Niveles de la batería

Una vez más, el nivel de carga de las baterías volvió a ser un problema. Al principio era por que la batería no era suficiente como para alimentar los motores. Ahora que los motores iban más despacio sí que era suficiente para alimentar todo, pero surgía un problema que ya se tuvo cuando los giros estaban limitados por tiempo y no por ángulo de rotación. Cuando la batería tenía poca carga, había ocasiones en las que los sensores de línea dejaban de funcionar de manera correcta. Detectaban intersecciones en una recta, había intersecciones que directamente no las detectaba…

La solución a este problema ya se planteó al principio del desarrollo, pero no se ejecutó ya que hasta el momento no era necesario. La solución era añadir una segunda batería y conectar ambas en serie aumentando por dos la capacidad de energía y ayudando al *output* de estas.

Esto dio lugar a una nueva iteración en la que el único cambio que se hizo fue a la parte trasera del chasis para hacer hueco a la segunda batería.



En general fueron cambios bastante simples. Se tuvo que mover el giroscopio y el controlador de los motores hacia arriba en 17mm y cambiaron ligeramente los enganches de los dos motores traseros. Respecto al resto del prototipo, se quedó intacto.

Ilustración 42, render del robot con dos baterías

De esta manera, ya no surgía el problema de los sensores fallando por temas de energía lo cual permitió centrarse en otros aspectos del robot.

5.6 Conclusiones

En este capítulo se consiguió perfeccionar las intersecciones pasando por varios planteamientos hasta llegar a una representación la cual serviría para usar como producto final. También se introdujo la coexistencia de los sensores de línea con el giroscopio en las rectas para los casos en los que el robot se empezaba a desviar demasiado y el comportamiento básico no era suficiente para rectificar el rumbo. Además, se ha estudiado también la importancia de la iluminación para el correcto funcionamiento de los sensores. Por último, se realizaron pruebas con más baterías para comprobar si

afectaba positivamente sobre el comportamiento final del robot ya que de esta manera se dispondría del doble de capacidad eléctrica, aumentando el tiempo óptimo de funcionamiento.

Capítulo 6

Evolución del robot

"Mira, si no gano, venderé todas las chapuzas que tengas.
Pero si gano, yo decido cuando termine."
-Rayo McQueen

6.1 Introducción

Tras el apartado anterior, podría parecer que cada problema independientemente se había solucionado, y realmente si era así, pero con que fallara una mínima cosa ya estropeaba todo y el robot se perdía. Se invirtieron muchas horas haciendo pequeños cambios a los valores de calibración de cada sistema inteligente que había en el robot, pero no se llegó a ningún estado en el que se solucionaba todo.

En este punto del desarrollo ya no quedaban más ideas para solucionar problemas y la única solución era cambiar el planteamiento del problema. Por tanto, en vez de dirigir el robot íntegramente por software, se decidió introducir algún sistema de guías físicas que ayudarían a enderezar al robot.

6.2 Posibles soluciones

La primera nueva implementación suponía unos cambios muy drásticos, pasaríamos de tener cuatro ruedas con un motor en cada una de ellas a tres ruedas y dos motores. En resumidas cuentas, esta iteración pasaría de ser dirigida por sensores y líneas en el suelo a ser dirigida por unos carriles tipo *Scalextric* y la dirección pasaría de ser de tipo "tanque" a ser más parecida a la de un coche normal con una rueda delantera que rota. En la Ilustración 43 vemos una comparativa del sistema antiguo con el nuevo y un ejemplo de su funcionamiento.

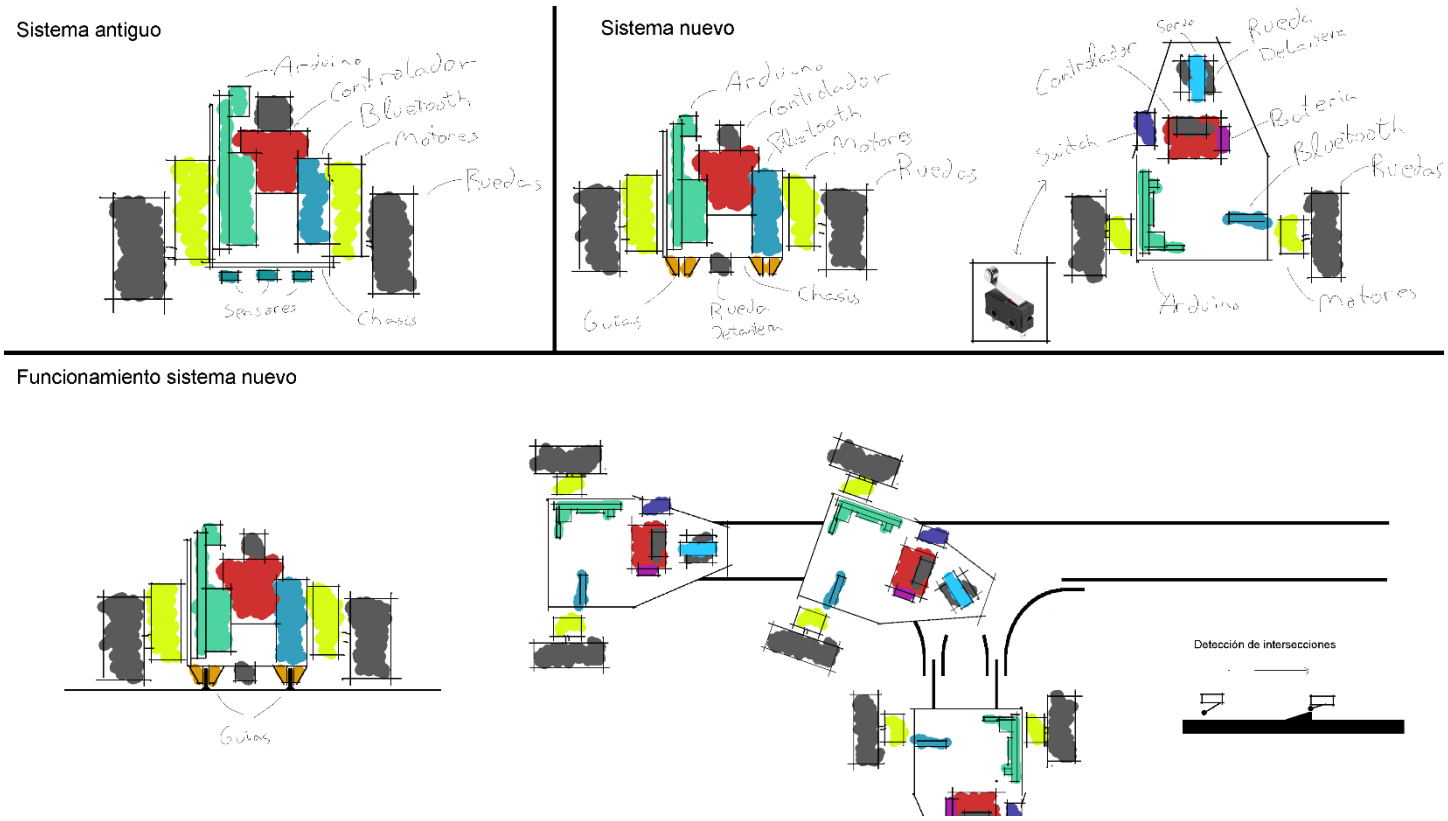


Ilustración 43, primera iteración del nuevo planteamiento

De frente vemos como lo único que cambia es la parte inferior del robot en la que pasaríamos de los tener tres sensores de línea a dos guías que se encargarían de dirigir el robot. Mirando el robot desde encima, ya sí que cambia bastante, el chasis tiene una forma distinta ya que tan solo se necesita una rueda delantera. Esta rueda estará dirigida por un servo en la parte superior que se encargaría de rotarla. El último gran cambio al robot es el interruptor situado en la parte izquierda del mismo. Este interruptor sería el que detectaría las intersecciones las cuales vienen marcadas con una especie de rampa en las propias vías (esquina inferior izquierda de la Ilustración 43).

Los cambios al mapa también serían bastante drásticos ya que ahora necesitaríamos dos carriles físicos los cuales irían encajados en las guías del robot. En la parte inferior de la Ilustración 43 vemos como funcionaria. Las intersecciones sería lo más complicado, una vez más, ya que deberían de ser abiertas para permitir todo tipo de movimientos. Vemos que, al ser abiertas, una vez se termina la intersección, habría que tener una

especie de embudo para hacer que las guías del robot encajaran directamente a los raíles de las rectas.

Como esta implementación supondría unos cambios demasiados drásticos a este punto del desarrollo, se decidió explorar otras soluciones más fáciles de adaptar al sistema que ya se tenía. La única cosa clara era que se necesitaba un sistema de guías físicas.

La otra opción era intentar adaptar el mapa ya existente. En las rectas, se podrían añadir unas guías a cada lado, pero estas en vez de ser lo que dirige al robot, serían unos límites que impedirían al robot desviarse demasiado. La lógica interna del robot se mantendría exactamente igual y seguiría estando guiado principalmente por los sensores y por el giroscopio solo que ahora además existirían unas barreras físicas para cuando se desvía demasiado. En la Ilustración 44 podemos ver los cambios principales al robot.

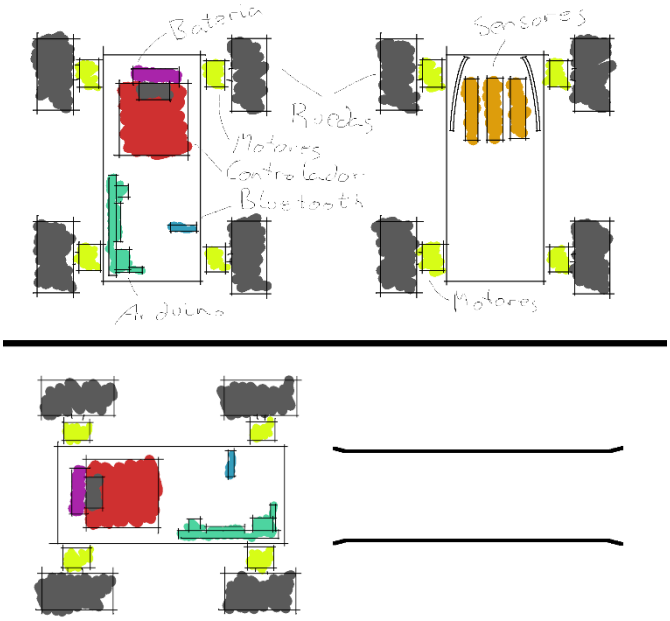


Ilustración 44, segunda iteración del nuevo planteamiento

Vemos que en la parte inferior del robot (esquina superior izquierda de la Ilustración 44) se han añadido unas “cuñas” que serán las que harán tope con las guías colocadas en el suelo. Además, los sensores se han movido del centro del chasis a la parte delantera. Este cambio ayudaría a la dirección, pero también habría que tenerlo en cuenta al ahora de detectar una intersección ya que ahora se detectaría antes de tener que realizar el giro.

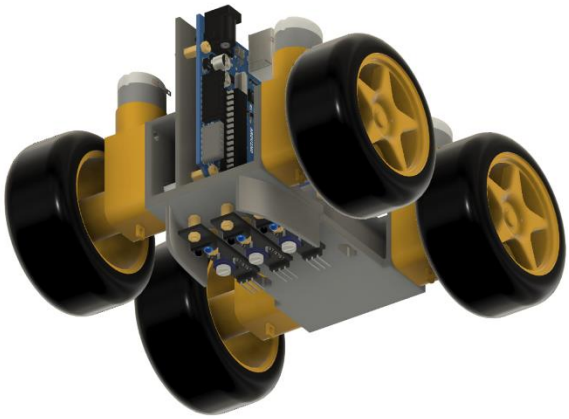


Ilustración 45, render del robot tras añadir las cuñas

6.3 Solución implementada

Al final se decidió ir con esta segunda opción. Se usarán perfiles de PVC de 1x1cm pegados al mapa para representar las guías.

Ilustración 46, perfiles PVC usados como guías

En la Ilustración, 44 podemos observar cómo sería la representación final. La parte plana de los perfiles iría orientada hacia el exterior para no impedir la visibilidad de las líneas a los sensores.

También vemos como los perfiles no llegan hasta la zona de detección de la intersección, esto es para el área libre alrededor de cada intersección que permitir realizar giros. En concreto, se necesita unos 8cm de área libra desde la zona de detección.

Este sistema tuvo efecto al instante. Con apenas unos pocos cambios, el robot ya era capaz de encadenar más intersecciones seguidas de las que se habían conseguido hasta el momento. Aun teniendo resultados favorables, no eran perfectos. El inicio de las zonas de guías eran abruptas, si cuando llegaba el robot a estas zonas, sus cuñas no entraban en el rango de tolerancia de las guías, se chocaría. Por este motivo, se imprimieron otra especie de cuña, pero esta vez para las guías (Ilustración 47).

De este modo, había muchas más tolerancias a la hora de entrar a las intersecciones y se reducían el número de choques que había al enterar a estas zonas

Ilustración 47, cuñas en el inicio de las guías

Las rectas largas ya estaban solucionadas en teoría, ahora había otro problema, había tramos de rectas que median menos de 16cm entre intersección e intersección. Recordemos que necesitamos 8cm de margen a cada lado de las intersecciones para colocar las guías. Por tanto, para estas zonas en las que no podía haber guías, había que confiar completamente en el movimiento habitual con los sensores. Una solución era hacer el mapa más grande para poder poner guías en todos lados, pero tener algunas zonas cortas sin guías tampoco debería ser un problema demasiado grande.

También, había zonas en las que tan solo se podría poner 1cm de guía, lo cual tampoco era de mucha ayuda ya que causaba más accidentes. Para estas zonas, se diseñó una guía redonda cuyo objetivo no era guiar en línea recta al robot, si no que se usaba para enderezar ligeramente su dirección si pasaba por esa zona con una trayectoria desviada.

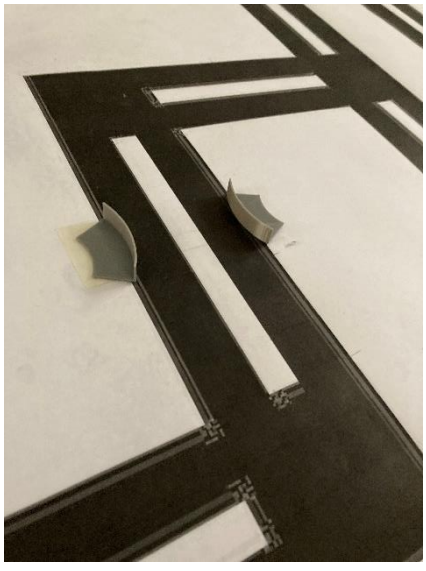


Ilustración 48, guías redondas entre intersecciones muy cerca

Una vez puestas en el mapa quedaban como se ve en la Ilustración 48. Había veces en las que estás guías especiales si servían, pero la mayoría de las veces eran contraproducentes. Muchas veces el robot se chocaba con ellas y acababa saliendo más desviado de lo que había entrado a esta pequeña zona de guías.

Al final, se acabaron descartando por este motivo. Igualmente, estas intersecciones estaban tan juntas que en la realidad al robot no le daba tiempo a desviarse tanto como para no llegar a la siguiente intersección.

Con el robot siendo capaz de enlazar más intersecciones de las que había sido capaz hasta el momento, salieron nuevos problemas. El giroscopio iba acumulando un ligero error. Este error era tanto que al girar 360 grados acumulaba unos cinco grados de error. Teniendo en cuenta que los giros se realizan con un margen de error de tan solo dos grados, podemos ver como este error acumulado tenía un gran efecto sobre el comportamiento final. Básicamente, el robot podía realizar un recorrido casi al completo hasta el momento en el que se completaba un giro completo, en ese momento se perdía prácticamente al instante por culpa de este error.

Se intentó cambiar el giroscopio actual, el MPU6050 por uno de mejor calidad, en concreto el BMI160. Este componente era más preciso y tenía detección de movimiento en nueve ejes en vez de tan solo en seis. El problema con este componente es que no existen librerías publicas que encapsulen todo su comportamiento facilitando el uso del mismo. Tampoco existen tutoriales en internet ya que este componente está orientado a un ámbito industrial y no a proyectos de pequeña escala como este. Independientemente de esto, se intentó utilizar la API oficial del componente[19] para obtener los datos, y a través de la librería *Curie IMU*[20], convertir estos datos a *pitch, roll* y *yaw* para poderlo usar en el proyecto. Los resultados obtenidos no eran los esperados, y no se consiguió un funcionamiento decente para introducirlo al proyecto.

Haciendo pruebas independientes al giroscopio se sacó el error que acumulaba en cada giro completo (360 grados) y se cambió la implementación del giroscopio para que, con cada lectura de datos, se tuviera en cuenta el error proporcional que había acumulado desde la lectura anterior para poderlo sumar y por tanto neutralizar este error. Para ayudar, se recalibró desde cero todo el giroscopio mediante un proceso en el que se deja el giroscopio quito en un sitio, y mediante unas lecturas durante dos o tres minutos, se calcula la desviación y se establece como valores por defecto. De esta manera, teóricamente, la próxima vez que se enciende el componente, usará dichos valores para no acumular desviaciones. Pero pese a usar los métodos de las librerías que se estaban usando para calibrar el giroscopio, no sirvió para mucho ya que se seguía con el mismo error.

Se encargaron nuevas baterías de mejor calidad que aumentaban la capacidad de 200mah a 850mah. Esto proporcionaría más tiempo de funcionamiento y aumentar el intervalo de tiempo en el que los motores tienen suficiente fuerza como para maniobrar el robot. Estas baterías funcionaban bien, y efectivamente duraban más tiempo, pero aún se tenía un problema con los giros y la falta de potencia.

Resumidamente, cuando las ruedas de un lado giran en dirección contrario a las del otro lado, hay dos fuerzas las cuales hay que sobrepasar. La primera es la fuerza paralela al robot (hacia delante y hacia atrás). Esta fuerza es la más leve ya que las ruedas están orientadas correctamente y apenas hay fricción. La segunda fuerza a tener en cuenta durante los giros es la perpendicular. Cuando estamos rotando el robot, las ruedas tienen que deslizarse por el suelo para rotar su orientación. Por tanto, al rotar sí que hay bastante rozamiento que tenemos que superar y esto causaba que el robot pegara pequeños saltos cuando realizaba giros. Esto es debido a que las ruedas cogían tracción en el suelo, pero

como la fuerza aplicada sobre ellas era de 45 grados, no podían rotar libremente, por tanto, cuando la fuerza sobre las ruedas pasaba de un cierto umbral, estrás realizaban un pequeño bote, rotaban en el aire y volvían a coger tracción.

El motivo por el que habría que tener esto en cuanta es porque estos pequeños botes mueven todos los componentes, incluyendo el giroscopio, afectando potencialmente a los valores que lee y por tanto afectando al comportamiento del robot.

Otra consecuencia relacionada con este problema es que como en los giros hay que superar una fuerza de rozamiento, los motores debían tener suficiente potencia como para superar esta fuerza. El problema es que, con el aumento de fuerza de los motores, también viene el aumento de velocidad en línea recta, y con el aumento de velocidad los sensores de línea no eran capaces de rectificar el rumbo creando así una espiral sin fin de problemas.

Una posible solución a estos dos problemas es sustituir las ruedas por orugas que conectaran ambos motores de cada lado. Las orugas al rotar seguirían teniendo que deslizar por el suelo, pero al ser de un material con un coeficiente de rozamiento más bajo permitiría deslizarse más fácilmente. Otro beneficio que añadirían las orugas sería conectar ambos motores multiplicando por dos la potencia que se dispone para rotar un lado del robot permitiendo así reducir la velocidad total del mismo sin tener el problema de la falta de fuerza a la hora de realizar giros.

En la Ilustración 49 podemos ver un render de los diseños de los eslabones y la rueda que iría conectada a los motores para hacer girar la oruga entera.

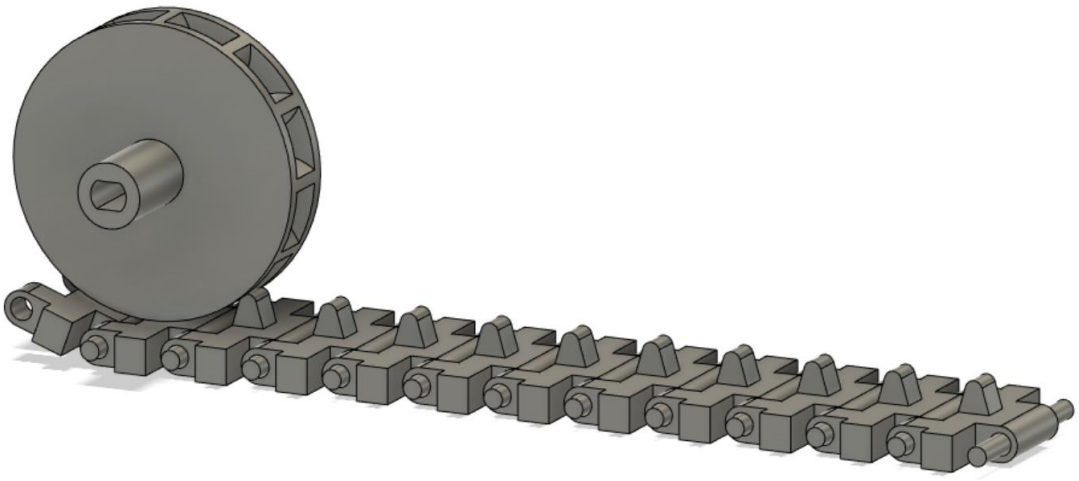


Ilustración 49, Render de las orugas

Se usaron palillos de madera como pasadores entre cada eslabón, dejando el robot con el aspecto siguiente:

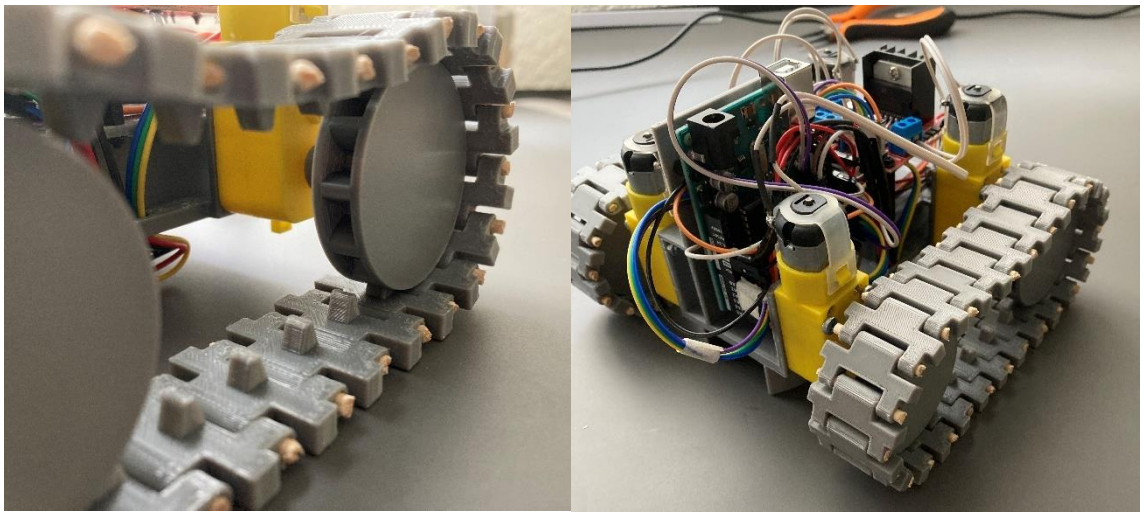


Ilustración 50, robot final con las orugas

La implementación de las orugas tuvo su parte buena y su parte mala. La mala era que las guías físicas que había previamente en el mapa ya no servían ya que ahora se necesitaba más espacio para realizar giros. La buena parte era que el robot era más fiable, casi lo suficiente como para eliminar por completo las guías. Se llegaron a pasar más de 20 intersecciones seguidas sin perderse lo cual era un récord hasta el momento.

La adaptación de las orugas al mapa no fue tarea sencilla, se tuvo que cambiar por completo el método de frenada ya que ahora con las orugas se frenaba mucho más rápido ya que había más superficie de contacto con el suelo. Como los motores ahora estaban conectados, se pudo bajar la velocidad total del robot. Y finalmente, también se tuvo que cambiar la representación del mapa puesto que, con los sensores de línea ahora posicionados en la parte delantera del robot, no era necesario una zona de detección tan grande. En la Ilustración 51 vemos el antes y el después de la representación de las intersecciones.

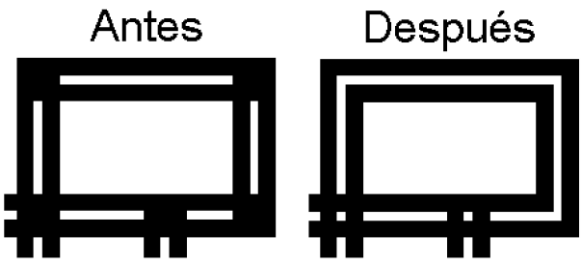


Ilustración 51, nueva representación de intersecciones

En las intersecciones en las que tan solo había una opción de giro, ahora se podía utilizar la propia línea de guía como detección (explicado más adelante) ya que se podía frenar prácticamente en seco dejando el robot casi en el centro de la intersección.

Las intersecciones con más de una sola opción suponían un nuevo problema ya que no se podían eliminar del todo por el hecho de que no hay una línea perpendicular que delimite la intersección.

Para entender mejor el por qué se pueden eliminar algunas zonas de detección, tomemos de ejemplo la esquina superior izquierda de la Ilustración 51. La zona de detección se ha eliminado por completo, pero aun así en algún momento, el robot se encontrará con una nueva línea en su camino, que es la guía de la otra recta que llega a esa misma intersección.

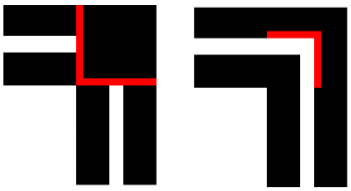


Ilustración 52, comparativa de zonas de detección

En la Ilustración 52, vemos marcado en rojo la antigua zona de detección frente la nueva. Vemos como efectivamente se ha eliminado el cuadrado negro grande en la intersección, pero la línea de detección tan solo se ha movido, no se ha eliminado.

Lo difícil ahora pasa a ser la detección de las intersecciones con más de dos caminos posibles. En la parte derecha de la Ilustración 51 vemos como las dos intersecciones de la parte inferior izquierda tan solo tienen un punto blanco en el medio. Esto es debido a que no se puede eliminar por completo ya que no hay zonas de detección "naturales" como si hay en la Ilustración 51.

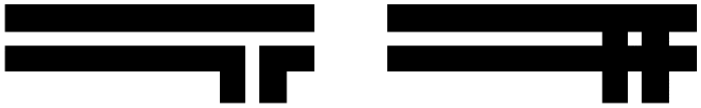


Ilustración 53, comparativa de zonas de detección en intersecciones complejas

Imaginemos que quitásemos por completo la zona de detección como se puede observar en la parte izquierda de la Ilustración 53. En el caso de que el robot viniese de izquierda a derecha y quisiera realizar un giro a derechas al llegar a la intersección (ir hacia abajo), no se podría ya que no hay zona de detección y no hay manera de saber si estamos sobre la intersección o no. Se podría argumentar que justo al pisar la intersección el sensor izquierdo estaría detectando una línea y los otros dos no, pero esto no es válido ya que cuando el robot se desvía hacia la izquierda, los sensores detectan este mismo caso. Por tanto, la única solución es dejar la intersección donde estaba y añadir un punto blanco en medio.

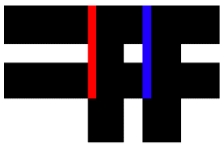


Ilustración 54, nuevas zonas de detección

La nueva manera de detección pasaría a estar compuesta por dos fases, la primera, marca con una línea roja en la Ilustración 54, es la fase de entrada en una intersección, a continuación, cuando se vuelve a detectar blanco, se sabe que la próxima vez que se detecte negro (línea azul en la ilustración), es que estamos sobre la intersección y tenemos que realizar el giro.

Este nuevo método mostró una gran mejoría, cada vez se avanzaba más. Ahora el robot era capaz de realizar dos veces un recorrido de unas doce intersecciones realizando un giro en la mayoría de ellas. Pese ver mejorías en los resultados, no se estaba ni por asomo cerca de la precisión que se debía tener para poder realizar este trabajo. Los errores que quedaban parecían estar relacionados todos con el giroscopio, el cual es un componente muy sensible a vibraciones, campos electromagnéticos, temperatura… Se trató de

introducir un cuarto sensor de línea en la parte posterior del robot, pero no fue de ayuda, el problema no estaba en las rectas, los tres sensores que ya había en el robot funcionaban muy bien, el problema eran las intersecciones.

Para acotar más el problema, se diseñó sobre el mapa actual, un recorrido rectangular, simple y grande. Se taparon la línea central de todas las intersecciones en las cuales no se iba a realizar un giro para eliminar al máximo los factores que podrían afectar. Esta prueba nos proporcionaría de varias conclusiones. La primera y más importante, comprobar si el robot tiene suficiente espacio entre intersección e intersección como para corregir su rumbo (indicándonos que el problema no está en los sensores), y la segunda conclusión nos proporcionaría información de si con cada giro realizado, el giroscopio acumulaba aún más error del que ya se tenía en cuenta.

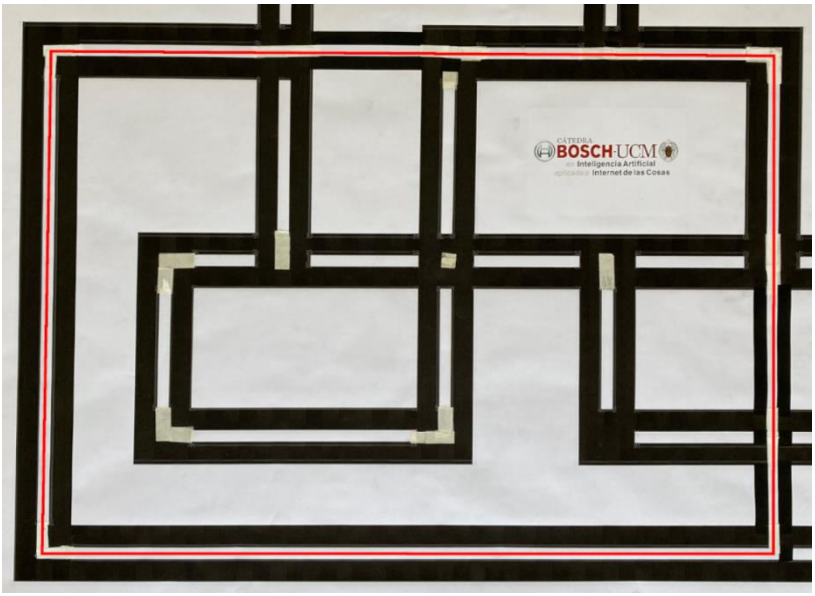


Ilustración 55, recorrido simple y grande

Los resultados fueron un poco mixtos. Si que parecía que el mayor culpable de los problemas era el giroscopio y por ello se decidió explorar más librarías por si el error estuviera allí, pero los resultados eran los mismo. Se recalibró el giroscopio con la ayuda de otra libraría, pero tampoco, se intentó hasta rotar el giroscopio para tenerlo verticalmente y así usar otro eje de guía por si el problema estuviera en que el eje z de este giroscopio no funcionara correctamente, pero esto fue aún peor.

Los robots de competición de seguimiento de línea no tienen giroscopios, de hecho, tan solo tienen dos sensores de línea. Viendo videos de estas competiciones surgieron varias preguntas, ¿Por qué necesitamos nosotros tres sensores en vez de dos? ¿Por qué necesitamos un giroscopio? ¿Por qué necesitamos cuatro ruedas motrices y no tan solo dos?

Estas preguntas llevaron a la idea de retomar el planteamiento de la Ilustración 43 (página 55) pero con algunos cambios. Los cambios serían los siguientes:

- Pasar de un movimiento dirigido por guías a uno dirigido por dos sensores de línea y por tanto eliminar también el interruptor para la detección de intersecciones.
- Eliminar el uso del servo para dirigir la rueda y hacer que sea una rueda libre, como la de los carritos de la compra.

- Usar las ruedas traseras como control de dirección, según la velocidad de estas, el robot giraría para un lado o para otro.

Realmente del planteamiento de la Ilustración 43 tan solo se mantuvo el concepto de la rueda delantera, el resto es distinto pese a que, aun así, se realizó un diseño para mantener el servo en la rueda delantera. Pero como podemos ver en la Ilustración 56, el diseño se complicaba demasiado y tener un servo significa tener en cuenta muchos otros factores que al final del día son más cosas que pueden fallar. No solo eso, sino que también habría que tener en cuenta que para corregir la dirección en una recta los movimientos del servo tendrían que ser mucho más sutiles que al llegar a una intersección

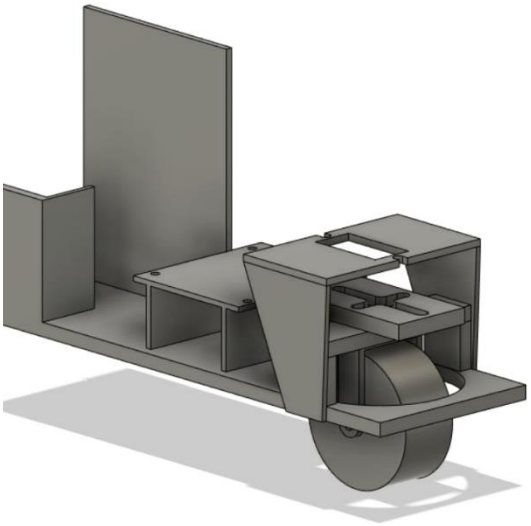


Ilustración 56, chasis teniendo en cuenta el servo

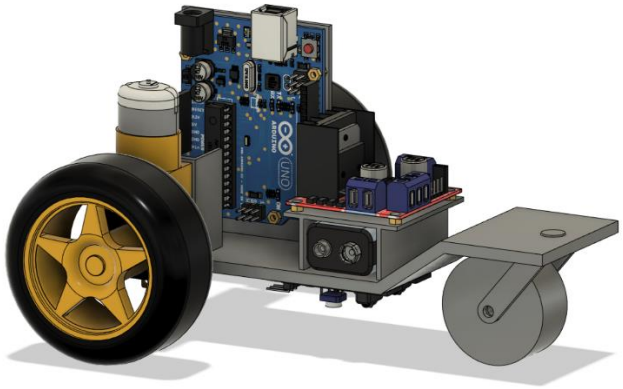


Ilustración 57, chasis sin servo

Para la simplificación de la rueda delantera se cogió como inspiración las ruedas de los carros de la compra. Este tipo de rueda rota según la fuerza que hace la persona que lleva el carro, en este caso, la persona serían los motores. De esta manera eliminábamos la necesidad de tener un servo.

El problema con esta implementación es que la fricción lateral sobre la rueda era mayor a la fuerza que era capaz de hacer el propio robot por tanto no giraba bien. La solución era cambiar el mecanismo de la rueda delantera por una en la que la fuerza sea completamente paralela al suelo y no en diagonal como estaba previamente en la Ilustración 57.

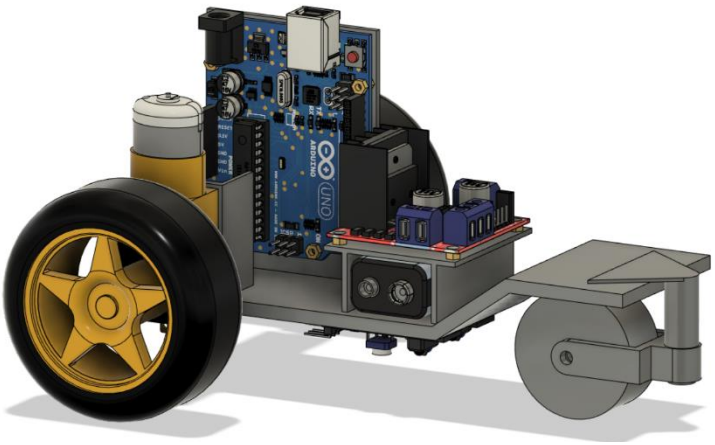


Ilustración 58, cambio en el sistema de rotación de la rueda delantera

Esta nueva implementación ya si era capaz de realizar giros (giros en el suelo sin ningún tipo de lógica ni seguimiento de línea). Tras implementar una pequeña lógica, era capaz de seguir una línea recta, pero no siempre. Ahora, como la dirección la controlaba únicamente la rueda delantera, los sensores no podían estar por detrás de la misma. En la prueba si lo estaban, y la mayoría de las veces, cuando se detectaba que era necesario corregir el rumbo, ya era demasiado tarde como para corregirlo. La solución era mover los sensores por delante de la rueda, pero en caso de hacer esto, se nos quedaría un robot extremadamente largo. Se tendría que cambiar a otro tipo de sensores que en vez de tener que estar paralelos al suelo, tienen que estar perpendiculares, de esta manera, tampoco habría que alargar tanto el robot.

6.4 Conclusiones

En este capítulo se han descrito las distintas evoluciones realizadas al robot para solucionar los distintos problemas encontrados durante su evaluación funcional. Después de realizar un análisis de las distintas soluciones aplicables, se describe cómo se han implementado algunas de ellas y cómo debería continuarse con la evolución del robot para conseguir una funcionalidad totalmente correcta.

Capítulo 7

Conclusión y trabajo futuro

"No vine hasta aquí sólo para verte renunciar."
– Doc Hudson

7.1 Conclusiones

El prototipo final es capaz de navegar un mapa durante unos treinta segundos pasando por veinte intersecciones sin perderse, lo cual, pese a no ser suficiente para una partida entera, ya es un gran avance el cual servirá como punto de partida para el trabajo futuro. Además, durante el tiempo de juego, es capaz de mantener una comunicación bidireccional con el servidor intercambiando mensajes de direcciones.

Mientras está en movimiento, el robot está constantemente autocorrigiendo su rumbo guiándose por los datos leídos por los sensores, aumentando y reduciendo la entrega de potencia a cada motor para obtener un movimiento suave al mismo tiempo que corrige su rumbo. Paralelamente a esto, está leyendo los datos del giroscopio que le servirán para calcular la dirección media en la que se está dirigiendo, y más adelante, saber exactamente cuantos grados girar a la hora de llegar a una intersección independientemente del ángulo de entrada.

A la hora de analizar el prototipo final, también consideramos el servidor arduino como parte del prototipo. Al fin y al cabo, es el que controla el rumbo del robot y es el responsable de su funcionamiento. Hasta que el servidor arduino no se sincroniza con el motor MsPacMan y con el propio robot, este no empezará a moverse. El servidor arduino está constantemente pendiente de si recibe mensajes tanto del motor MsPacMan como del robot. En el último prototipo del trabajo, los únicos mensajes recibidos por parte del robot son los correspondiente a la parte de depuración mediante la luz led, y los mensajes recibidos del motor tan solo corresponden a las direcciones del MsPacMan. En esta última parte se han omitido las instrucciones relacionadas con los fantasmas ya que estos aún no están representados en el mundo real.

La representación del mapa está bastante refinada al modelo de robots que se tiene actualmente. Los márgenes a cada lado del robot en las rectas y en las intersecciones están bien calculadas. La única cosa que podría afectar es la distancia entre intersecciones, no la representación como tal. En caso de querer cambiar esta distancia, tan solo habría que aumentar el tamaño del mapa, pero manteniendo el grosor de las líneas negras.

Durante todo el desarrollo se han implementado numerosas iteraciones evolucionando el prototipo hasta el estado actual. Las primeras cinco iteraciones son las principales del proyecto, ya que son las que han focalizado el trabajo. En estas iteraciones se perfeccionó la rigidez del chasis, recolocación e introducción de componentes, tolerancias… Pero estas iteraciones no fueron las únicas más adelante fueron surgiendo más. En la sexta iteración se introdujo una segunda batería, en la séptima se introdujeron las orugas, y finalmente en la octava y novena iteración, se exploró la posibilidad de utilizar robots con tan solo tres ruedas.

Constantemente, en cada iteración se realizaban numerosas pruebas buscando perfeccionar el prototipo lo máximo posible permitiendo así evolucionar tanto el hardware como el software.

Gracias a este proyecto se han obtenido gran cantidad de conocimientos abarcando campos como es el diseño e impresión 3D, el mundo del arduino y sus componentes, los controladores, sistemas de comunicación tanto por cable como vía Bluetooth, software de simuladores, IoT, funcionamiento de sensores…

Al ser un trabajo muy llamativo y elaborado, ha sido elegido para un spot publicitario de la propia facultad[22]. Lograr esto, no se consigue sin un buen proyecto, la cual es una buena señal. Además, el proyecto ha sido escogido para su exposición en el Aula 2023[23], el salón internacional del estudiante y oferta educativa que se celebrará el próximo año en IFEMA, Madrid para representar a la facultad.

Ilustración 59, día de grabación para el spot publicitario de la facultad de informática

7.2 Limitaciones

La primera limitación es la precisión del robot. No se ha conseguido de momento una comunicación entre tecnologías lo suficientemente robusto como para reproducir una partida entera. Esta limitación está condicionada por otras como es el error acumulado del giroscopio, la potencia de las baterías y la fuerza de los motores en ciertas condiciones. Como el robot está compuesto por muchos componentes distintos y al no disponer de ninguna PCB customizada que une varios de estos componentes en uno, el peso del robot es bastante elevado, influyendo así sobre el comportamiento final del robot.

Otra limitación es el tamaño del mapa. La idea principal era hacer un mapa de unos dos metros de ancho, pero con el robot final siendo de unos veinte centímetros, el mapa tendría que ser bastante más grande para obtener el resultado esperado.

El resultado final que se planteaba al principio no se ha conseguido por completo, pero aun así se ha conseguido una funcionalidad muy alta. La parte del software se podría mantener tal cual cara al futuro ya que cumple su función completamente y es lo suficientemente robusta como para aguantar proyecto de esta escala. Esta parte, sirve como ejemplo de cómo unir los videojuegos al internet de las cosas y como realmente si es factible hacer un videojuego en físico.

Teniendo en cuanta la poca información disponible en internet, siendo uno de los pocos trabajos del estilo, y haberlo desarrollado todo por un solo estudiante del grado de desarrollo de video juegos, lo considero un éxito. Ha sido uno de los mayores retos, sino el mayor, al que me he enfrentado y deja un sabor agridulce no haberlo conseguido, pero, no obstante, sigo orgulloso de lo conseguido y seguiré trabajando sobre el proyecto hasta conseguirlo.

7.3 Trabajo futuro

Cara el futuro, en este trabajo se puede hacer muchas cosas. Lo primero de todo sería realizar una nueva iteración con todos los conocimientos obtenidos hasta el momento. A continuación, con un prototipo 100% funcional, se podría encapsular todo en un mismo programa y desarrollar una interfaz de usuario para que sea más accesible para las personas que no estén muy enteradas del proyecto y por tanto acercarlo más al público y hacerlo más fácil de usar para el usuario medio (hasta el momento, el simulador MsPacMan y el programa que lo conecta con los robots, son dos programas diferentes).

Una posible nueva implementación sería basarse en los coches de miniatura llamados *slots* (popularizados y mayormente conocido en España como Scalextric). Se eliminarían muchos componentes de los coches empezando por el más pesado, la batería ya que, en los *slots,* los coches obtienen su energía de las propias vías del mismo modo que lo hacen

los coches de choque en las ferias. Al ir todo guido por carriles por todo el recorrido, se podrían eliminar los sensores y el giroscopio. Los cambios de carriles estarían controlados por las propias vías las cuales cambiarían físicamente el camino (como ocurre en las vías de los trenes) por el cual iría la guía del coche permitiéndonos eliminar el arduino de cada robot.

Chapter 7

Conclusion and future work

"I didn't come all this way just to see you quit"
-Doc Hudson

7.1 Conclusions

The final prototype is able to navigate a map for about thirty seconds passing through twenty intersections without getting lost, which, although it is not enough for an entire game, it is already a breakthrough which will serve as a starting point for future work. In addition, during game time, it's able to maintain a two-way communication with the server by exchanging directional messages.

While in motion, the robot is constantly self-correcting its course based on the data read by the sensors by increasing and decreasing the power delivery to each motor to obtain a smooth motion while correcting its course. In parallel to this, it is reading data from the gyroscope that will help it calculate the average direction in which it is heading, so that later, it knows exactly how many degrees to turn at an intersection regardless of the angle of entry.

When analyzing the final prototype, we also consider the Arduino server as part of the prototype. After all, it is the one that controls the direction of the robot and is responsible for its operation. Until the arduino server is synchronized with the MsPacMan engine and the robot itself, the robot will not start moving. The arduino server is constantly listening to see if it receives messages from both the MsPacMan engine and the robot. In the last prototype of the work, the only messages received by the robot are those corresponding to the debugging part through the led, and the messages received from the engine only correspond to the MsPacMan directions. In this last part, the instructions related to the ghosts have been omitted since they are not yet represented in the real world.

The representation of the map is quite refined to the current robot model. The margins on each side of the robot on the straight and at intersections are well calculated. The only thing that could affect the final product is the distance between intersections, not the representation as such. In case you want to change this distance between intersections, just increase the size of the map, but keep the thickness of the black lines the same.

Throughout the development, numerous iterations have been implemented, evolving the prototype to its current state. The first five iterations are the main ones of the project,

since they are the ones that have narrowed down the scope of the project. In these iterations, the chassis stiffness, component repositioning and introduction, tolerances... were perfected. But these iterations were not the only one, later on, more were emerging. In the sixth iteration a second battery was introduced, in the seventh iteration tracks were introduced, and finally in the eighth and ninth iteration, the possibility of using robots with only three wheels was explored.

Constantly, in each iteration, numerous tests were carried out in order to perfect the prototype as much as possible, thus allowing both the hardware and the software to evolve.

Thanks to this project, a deep knowledge in various fields has been gained, covering fields such as 3D design and printing, the world of the arduino and its components, controllers, communication systems both wired and via Bluetooth, simulator software, IoT, sensor operation...

As it's a very striking and elaborate project, it has been chosen for an advertising spot of the faculty itself[22]. This cannot be achieved without a good project, which is a good sign. In addition, the project has also been chosen to be exhibited at *Aula* 2023[23], the international student fair and educational offer that will be held next year at IFEMA, Madrid to represent de faculty.

Illustration 59, filming day for the advertising spot of the faculty

7.2 Limitations

The first limitation is the accuracy of the robot. It has not yet been possible to achieve a communication between technologies robust enough to reproduce an entire game. This limitation is conditioned by others such as the accumulated error of the gyroscope, the power of the batteries and the strength of the motors in certain conditions. As the robot is

made up of many different components and as it does not have any PCB custom PCB that joins several of these components into one, the weight of the robot is quite high, thus influencing the final behavior of the robot.

Another limitation is the size of the map. The main idea was to make a map about two meters wide, but with the final robot being about twenty centimeters, the map would have to be quite larger to get the expected result.

The final result that was proposed at the beginning has not been achieved completely, but still, a very high functionality has been achieved. The software part could be kept as it is for the future as it fulfills its function completely and is robust enough to withstand projects of this scale. This part serves as an example of how to link video games to the internet of things and how it is really feasible to make a physical video game.

Considering the little information available on the internet, being one of the few works of its kind, and having developed it all by a single student of the videogame development degree, I consider it a success. It has been one of the biggest challenges, if not the biggest, that I have faced and it leaves a bittersweet taste not having achieved it, but, nevertheless, I am still proud of what I have achieved, and I will continue working on the project until I get it.

7.3 Future work

Looking ahead, there are many things that can be done in this work. The first thing would be to make a new iteration with all the knowledge obtained so far. Then, with a 100% functional prototype, we could encapsulate everything in a single program and develop a user interface to make it more accessible to people who are not very knowledgeable about the project and thus, bring it closer to the public and make it easier to use for the average user (so far, the MsPacMan simulator and the program that connects it with the robots are two different programs).

A possible new implementation would be based on the miniature cars called slots (popularized and mostly known in Spain as Scalextric). Many components of the cars would be eliminated, starting with the heaviest, the battery, since in the slots, the cars get their energy from the tracks themselves in the same way that bumper cars do at fairs. As everything is guided by rails along the entire route, sensors and gyroscope could be eliminated. Lane changes would be controlled by the tracks themselves which would physically change the path (like train tracks) along which the car's guidance would go, allowing us to eliminate the arduino from each robot.

Bibliografía

[1] Wikipedia 2022, internet de las cosas: https://es.wikipedia.org/wiki/Internet_de_las_cosas

[2] Wikipedia 2022, MsPacMan: https://es.wikipedia.org/wiki/Ms._Pac-Man

[3] Grupo GAIA 2022: https://gaia.fdi.ucm.es/

[4] YouTube 2022, competiciones de seguimiento de líneas: https://www.youtube.com/watch?v=db6oKIfeocs

[5] Wikipedia 2022, slot: https://es.wikipedia.org/wiki/Slot_(modelismo)

[6] Scalextric 2022: https://scalextric.es/

[7] Wikipedia 2022, PacMan: https://es.wikipedia.org/wiki/Pac-Man

[8] Wikipedia 2022, Namco: https://es.wikipedia.org/wiki/Namco

[9] Wikipedia 2022, *esports:* https://es.wikipedia.org/wiki/Deportes_electr%C3%B3nicos

[10] Wikipedia 2022, Fdm: https://es.wikipedia.org/wiki/%C3%81cido_polil%C3%A1ctico

[11] Wikipedia 2022, PLA: https://es.wikipedia.org/wiki/%C3%81cido_polil%C3%A1ctico

[12] Arduino 2022, documentación oficial de Arduino: https://www.arduino.cc/en/main/docs

[13] YouTube, tutoriales de arduino: https://www.youtube.com/channel/UC4unPLtykzwO7MB3IvaQZaA

[14] Arduino 2022, librería *SoftwareSeria:* https://docs.arduino.cc/learn/built-in-libraries/software-serial

[15] GitHub 2022, librería EasyTransfer/SoftTransfer: https://github.com/madsci1016/Arduino-EasyTransfer

[16] AutoDesk 2022, modelos en AutoDesk: https://gallery.autodesk.com/

[17] GitHub 2022, librería I2cdev: https://github.com/jrowberg/i2cdevlib

[18] GitHub 2022, librería Simple_MPY6050: https://github.com/ZHomeSlice/Simple_MPU6050

[19] GitHub 2022, API giroscopio BMI160: https://github.com/BoschSensortec/BMI160_driver

[20] Arduino 2022, librería curie: https://docs.arduino.cc/retired/archived-libraries/CurieIMU

[21] GitHub 2022, librería jSerialComm: https://github.com/Fazecast/jSerialComm

[22] YouTube, Spot publicitario FDI: https://www.youtube.com/watch?v=lXZ_973CllA

[23] Ifema 2022, Aula 2023: https://www.ifema.es/aula

[24] Arduino 2022, arduino Uno: https://arduino.cl/arduino-uno/

[25] Components101 2022, puente en H I298N: https://components101.com/modules/l293n-motor-driver-module

[26] Datasheet 2022, sensor ky-033: https://datasheetspdf.com/datasheet/KY-033.html

[27] Components101 2022, modulo Bluetooth HC-05: https://components101.com/wireless/hc-05-bluetooth-module