

E.T.S. de Ingeniería Industrial,  
Informática y de Telecomunicación

# Diseño e implementación de un robot para la automatización de un almacén



Grado en Ingeniería  
en Tecnologías Industriales

Trabajo Fin de Grado

Pablo Ferro Laspidea  
Javier Goicoechea Fernández  
Pamplona, 25 de Junio de 2014

## Contenidos

---

1. Descripción y Objetivos .....	7
1.1. Descripción del proyecto .....	7
1.2. Objetivo.....	8
2. Justificación del proyecto.....	8
3. Antecedentes.....	9
4. Alternativas .....	11
4.1. Alternativas para el rastreamiento .....	11
4.1.1. Guías mecánicas.....	11
4.1.2. Cintas negras .....	11
4.1.3. Patrones de pegatinas en el suelo .....	12
4.2. Sensores reconocimiento ubicación.....	12
4.2.1. Sensores de paso.....	12
4.2.2. Sensores ópticos .....	12
4.2.3. Sensores de código.....	12
4.2.4. Elección .....	13
4.3. Sensores para reconocimiento de la línea.....	13
4.4. Disposición, situación y número de sensores .....	14
4.5. Tratamiento de la señal de los sensores .....	16
4.5.1. Utilización señal analógica.....	16
4.5.2. Utilización de señal digital .....	16
4.5.3. Elección .....	17
4.6. Microcontrolador .....	17
4.6.1. ¿Qué es?.....	17
4.6.2. Alternativas en el mercado .....	19
4.6.3. Arduino.....	19
4.7. Motores.....	22
4.7.1. Alternativas.....	22
4.7.1. Elección .....	24
4.8. Driver motor .....	25
4.9. Ruedas.....	28
4.10. Comunicación inalámbrica .....	29
4.10.1 Alternativas.....	29
4.10.2 Elección.....	32
4.11. Sensor distancia.....	33

5. Diseño .....	34
5.1. Acondicionamiento de los sensores .....	34
5.2. Determinación y cuantificación del error .....	38
5.3. Algoritmo de control .....	39
5.4. Conexionado driver.....	43
5.5. Configuración módulo Bluetooth.....	44
5.6. Programa LabView .....	47
5.6.1. Alternativas.....	47
5.6.2. Desarrollo del programa.....	48
5.7. Arduino .....	52
5.7.1. Pines .....	53
5.7.2. Programación.....	54
5.8. Diseño del PCB .....	59
5.7.1. ¿Qué es un PCB? .....	60
5.7.2. ¿Monocapa o multicapa?.....	60
5.7.3. Componentes fijos.....	61
5.7.4. Componentes libres.....	62
5.7.5. Planos.....	66
5.7.6. Dificultades y retos .....	67
5.7.7. Listado y descripción de las nets.....	68
5.7.8. Diseño final .....	70
5.7.9. Montaje y soldadura.....	73
6. Presupuesto.....	76
6.1. Coste de material robot.....	76
6.2. Coste mano de obra.....	77
6.3. Coste material variado.....	77
6.4. Presupuesto total.....	77
7. Posibles mejoras.....	78
7.1. Mejoras en los sensores.....	78
7.2. Toma de datos analógicos .....	78
7.3. ZigBee .....	78
7.4. Código de barras .....	79
7.5. Base de datos.....	79
7.6 Malla pegatinas QR.....	79
7.7. Diferentes modos de funcionamiento.....	80
7.8. Monitorización de datos a través de LabView.....	80

8. Conclusiones y valoración .....	81
9. Bibliografía .....	83
9.1. Recursos electrónicos .....	83
9.2. Recursos bibliográficos .....	83

Ilustración 1: Diagrama bloques proyecto.....	7
Ilustración 2: Seguimiento de la línea.....	15
Ilustración 3: Disposición de los sensores .....	16
Ilustración 4: Estructura de un microcontrolador.....	19
Ilustración 5: Tabla comparativa de todas las placas Arduino (web oficial).....	20
Ilustración 6: Comparación de tamaño entre Arduino Uno, Arduino Nano y una moneda .....	22
Ilustración 7: Estructura de puente en H.....	26
Ilustración 8: Diagrama pines del integrado L293D .....	27
Ilustración 9: Diagrama lógico entradas y salidas del 293D .....	27
Ilustración 10: Bola loca y rueda loca .....	28
Ilustración 11: Clasificación estándares de comunicación .....	30
Ilustración 12: Diferentes topologías de ZigBee.....	31
Ilustración 13: Módulo Bluetooth JY-MCU .....	33
Ilustración 14: Disposición de las patillas del fotodiodo y del fototransistor dentro del encapsulado TCRT1000 .....	35
Ilustración 15: Esquemático circuito sensores TCRT1000 .....	36
Ilustración 16: Esquemático en detalle de la conexión de un sensor TCRT1000	37
Ilustración 17: Voltajes de ciclo de histéresis según tensión alimentación .....	37
Ilustración 18: Ciclo de histéresis.....	37
Ilustración 19: Estructura clásica del controlador PID .....	40
Ilustración 20: Diagrama de pines driver L293D .....	43
Ilustración 21: Esquema de conexionado del driver 293d.....	43
Ilustración 22: Esquema conexionado del módulo Bluetooth.....	47
Ilustración 23: Estructura configuración puerto serie.....	49
Ilustración 24: Estructura escritura de caracteres al puerto serie.....	50
Ilustración 25: Estructura lectura de datos del puerto serie.....	51
Ilustración 26: Estructura while de escritura y lectura de datos.....	51
Ilustración 27: Panel frontal escritura caracteres .....	52
Ilustración 28: Esquema conexionado Arduino Nano .....	54
Ilustración 29: Disposición de los sensores (amarillo) respecto a la línea (morado) con el plano de GND quitado.....	62
Ilustración 30: Disposición del driver (U10) respecto a los motores, morado en los laterales .....	62
Ilustración 31: Zócalo Arduino Nano (U1).....	63
Ilustración 32: Integrado 74HC14 (U5) con la bola loca superpuesta (U17).....	63
Ilustración 33: Interruptor (U18) y conector de la batería (U15). .....	64
Ilustración 34: Módulo Bluetooth (U2) con sus resistencias R1 y R2 .....	64
Ilustración 35: Zócalo para sensor HC-SR04 (U8) .....	64
Ilustración 36: Resistencias del circuito de sensores .....	65
Ilustración 37: Zumbador y resistencia. ....	65
Ilustración 38: Los 3 planos de GND (verde) unidos por pistas GND.....	66
Ilustración 39: Pista de conexión entre planos .....	67
Ilustración 40: Conectores (PL1 y PL2) para sortear las pistas de en medio.....	68
Ilustración 41: Detalle 3d de los conectores PL1 y PL2 .....	68
Ilustración 42: Diseño final del PCB sin plano de tierra GND.....	70
Ilustración 43: Diseño final del PCB con plano de tierra GND.....	71
Ilustración 44: Vista en 3d de la parte inferior de la placa .....	72
Ilustración 45: Vista en 3d de la parte superior de la placa .....	72

Ilustración 46: Vista general en 3d del PCB.....	73
Ilustración 47: Vista posterior PCB soldado y montado.....	74
Ilustración 48: Vista frontal robot.....	75
Ilustración 49: Vista lateral del robot .....	75

## 1. Descripción y Objetivos

---

El presente proyecto tiene como objetivo el diseño e implementación de un robot para la automatización de un almacén industrial mediante el uso de comandos a través de un ordenador.

### 1.1. Descripción del proyecto

La realización del proyecto está orientada a la realización de un robot que nos automatice un almacén industrial por lo que necesitaremos una tecnología de comunicación entre nosotros y el robot así como una interfaz para su control y monitoreo.

El robot deberá estar capacitado para realizar procesos automáticos y reaccionar en función de las condiciones de su entorno, por lo que necesitaremos un microcontrolador con algoritmos de control como sistema de gobierno y sensores.

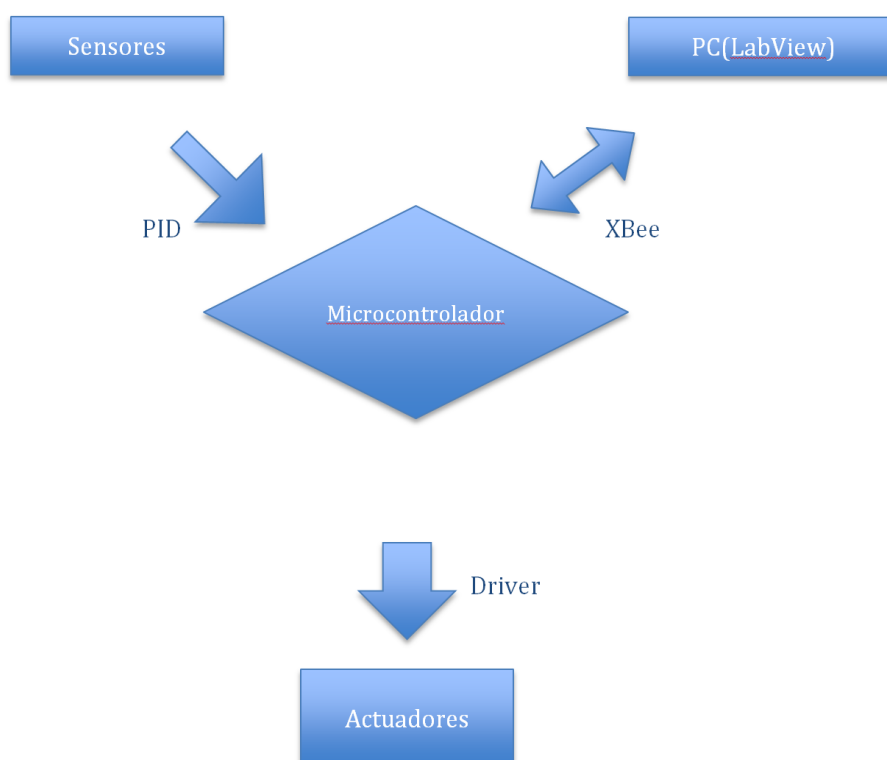


Ilustración 1: Diagrama bloques proyecto

El sistema de gobierno general de nuestro robot será una placa comercial Arduino Nano, el robot será capaz de seguir una línea negra y de girar en intersecciones según le indiquemos de forma inalámbrica, gracias a la tecnología Bluetooth, a través del ordenador con una aplicación de LabView. Para ello el robot contará con sensores infrarrojos para detectar la línea y con un sensor ultrasónico para detectar los obstáculos que se pudiera encontrar.

## 1.2. Objetivo

El objetivo del presente proyecto es desarrollar un robot siguelíneas con las funcionalidades antes descritas aplicando los conceptos aprendidos en el grado.

Además del desarrollo teórico en sí, el objetivo es también enfrentarnos a un proyecto real fuera un poco de los estudios y aprender a resolver los problemas que puedan aparecer de forma autónoma investigando y aprendiendo conceptos y procedimientos que no se han visto en el grado.

## 2. Justificación del proyecto

---

La tecnología nos permite obtener grandes mejoras en la sociedad y en las personas. Los avances tecnológicos que vemos cada día cambian la forma de ver e interactuar con nuestro entorno, vivimos en la era de las telecomunicaciones, todo está conectado, lo que hoy es algo normal dentro de 5 años estará anticuado u obsoleto.

Dentro de esta vorágine de mejoras y cambios se encuentra de lleno y de forma ineludible la industria. La industria ha sido, es y será el motor de la sociedad y todos estos cambios le afectan. Muchísimos procesos industriales son realizados por robots manipuladores en líneas de producción en industrias como la automovilística o la industria química.

Sin embargo en el campo de la automatización considero que los almacenes industriales no están cambiando al ritmo que lo están haciendo otros campos como puede ser la producción, organización o logística donde todo está conectado, comunicado y automatizado.



Las grandes fábricas y empresas si tienen en su mayoría implementado un sistema automático en la logística de sus almacenes y naves pero en empresas no tan punteras no se hace uso de una tecnología ya existente y no demasiado cara

Por ello quiero enfocar mi proyecto en el ámbito de la automatización industrial ya que considero que existe una necesidad de mejora tecnológica y además hay un nicho de mercado importante.

El objetivo del presente proyecto no será la realización de un robot para la automatización de un almacén completo si no que será el diseño e implementación del preámbulo de ese robot, la intención de este proyecto es que se pueda utilizar y extrapolar a un futuro proyecto completo de un robot con completa funcionalidad. Además se va a realizar con una herramienta de libre acceso y con una gran comunidad detrás como es Arduino.

La realización de este proyecto me va a permitir utilizar y mejorar conocimientos que se han adquirido a lo largo del grado en materias como el control automático, la programación de microcontroladores, el uso de sensores analógicos, el establecimiento de una comunicación entre 2 sistemas con una tecnología inalámbrica y el uso de programas informáticos vistos en la carrera como son LabView y DesignSpark.

### 3. Antecedentes

---

Como se ha comentado anteriormente el uso de los robots está aumentando cada vez más tanto en la industria, como en el uso para mejorar la vida de las personas en donde la domótica intenta abrirse paso. El uso de sistemas robóticos ha permitido alcanzar niveles de producción y de calidad inimaginables hace unos años. Los países más punteros en el uso de robots en sus industrias son Japón, EEUU y Alemania. Varias empresas tecnológicas punteras están empezando a usar robots en temas de logística y almacenamiento como es el caso de Amazon que está desarrollando un servicio de entrega de productos mediante drones o Google que ha comprado recientemente la empresa Boston Dynamics. Como podemos ver el auge

Pablo Ferro Laspidea

de la robótica en la industria es tremendo y la situación recuerda al desarrollo de los ordenadores en los años 80.

Centrándonos en el campo que nos atañe existen diferentes tipos de robots siguelíneas y rastreadores tanto para aplicaciones domesticas como industriales.

Dentro del ámbito doméstico los robots más famosos son los robots aspiradores Roomba de la casa iRobot. Existen múltiples modelos con diferentes funcionalidades y son los más vendidos del mundo en su campo. Pero si nos centramos en los robots siguelíneas se pueden encontrar productos comerciales como el Pololu 3pi.

Pero el campo que más nos interesa es el campo de la utilización de los robots en la industria no ya en la cadena de producción si no en la logística de los almacenes y es ahí donde entra en escena los robots de KIVA Systems.

KIVA Systems es una empresa creada en 2003 que se dedica al diseño y fabricación de robots de almacén y que fue adquirida por Amazon en 2012 por una cantidad cercana a los 780 millones de dólares.

Lo que diferencia a los robots de KIVA Systems respecto a otras alternativas existentes en el mercado es que no solo transportan el producto que se precisa si no que transportan directamente la estantería entera. Estos robots apoyados por un programa informático sincronizado con los pedidos y los stocks, revolucionan la forma de gestionar un almacén ya que ahora son los productos los que van hacia el operario y no al revés. El operario está en una estación de trabajo y según van entrando los pedidos le van llegando las estanterías con los productos que tiene que coger. El sistema informático tiene la capacidad de detectar que productos tienen menor demanda y los aparta a zonas más alejadas del puesto de trabajo del operario. Con este sistema se ahorra tiempo, dinero y esfuerzos aumentando la productividad y la flexibilidad del almacén.

El robot se mueve por el almacén a través de una malla de pegatinas QR que le proporcionan la información de su ubicación en el almacén, además cuentan con un sensor de obstáculos y esta información la comparten con el resto de robots.

Pablo Ferro Laspidea

Realmente este es el futuro de los grandes almacenes industriales ya que este sistema ahorra dinero, aumentando la productividad, la flexibilidad y reduciendo tiempos.

Reduciendo las distancias, esto es un poco lo que se intenta desarrollar, un robot autónomo que se pueda mover por un entorno grande y que se pueda controlar a través de un ordenador

## 4. Alternativas

---

### 4.1. Alternativas para el rastreamiento

En principio las alternativas para que nuestro robot se mueva por un camino o trayectoria que le fijemos nosotros son múltiples y variadas, a continuación se analizan las diferentes alternativas:

#### 4.1.1. Guías mecánicas

La solución más sencilla y más fácil sería la instalación de unas guías mecánicas en el suelo o en el techo de la nave a las que el robot sería anclado y mediante un actuador se movería por el camino prefijado que quisiéramos. Esta técnica se utiliza mucho en la industria en el transporte de materias primas y productos en una cadena de montaje y en la sustentación de herramientas pesadas con cierta libertad en los movimientos para el uso factible de los operarios. Esta alternativa aporta robustez y sencillez para la aplicación que deseamos sin embargo, tiene poca flexibilidad ya que una vez instalado no se puede modificar y además su precio es más elevado al de otras alternativas.

#### 4.1.2. Cintas negras

Consiste en la instalación de cintas negras, o de otro color, en el suelo “dibujando” las trayectorias y caminos que se desea que realice el robot. Esta opción nos dota de más flexibilidad en el diseño de los itinerarios y en una hipotética modificación en los mismos, pero tiene menos robustez que el caso anterior.

### 4.1.3. Patrones de pegatinas en el suelo

Esta tercera opción consiste en la realización de una malla cuadriculada de pegatinas en el suelo que el robot pueda identificar como pueden ser pegatinas QR, códigos de barras o pegatinas NFCs. Es la opción que nos permite una mayor flexibilidad, no demasiado cara y bastante robusta.

## 4.2. Sensores reconocimiento ubicación

El robot debe conocer en todo momento donde se encuentra y hacia dónde va ya que se supone que estamos implementando un sistema automático autónomo. Para acometer esta funcionalidad tenemos diferentes alternativas.

### 4.2.1. Sensores de paso

Se trata de colocar sensores inductivos o laser para informar al robot de que está pasando por un punto determinado o una bifurcación concreta. Son útiles en el caso de las guías metálicas ya que unos simples sensores inductivos serían suficientes.

### 4.2.2. Sensores ópticos

Se trata de sensores que perciban diferentes situaciones según se refleja la luz. Es el caso de los fotodiodos con fototransistores en los que según el color de la superficie en la que la emisión de los diodos refleje los fototransistores se excitaran más o menos y variaran su estado. Este tipo de sensores es la base con la que funcionan los robots seguidores de líneas pegadas en el suelo.

### 4.2.3. Sensores de código

Se trata de sensores de códigos de barras y códigos QR. Los códigos de barras han sido tradicionalmente los más utilizados y aún siguen y seguirán teniendo muchos ámbitos de aplicación aunque el código QR posee un mayor interés desde el punto de vista de la información ya que poseen más información que los códigos de barras y los tiempos de identificación son parecidos. Esta tecnología es muy útil en las mallas y redes de pegatinas en el suelo.

Pablo Ferro Laspidea

Comunicación inalámbrica: se trata de etiquetas de radio frecuencia. La tecnología RFID abarca diferentes opciones y la que más nos interesa en nuestro caso es el Near Field Communication. La tecnología nfc se trata de pegatinas capaces de proporcionar información a un chip nfc cuando este se encuentra a una distancia reducida. Esto nos puede resultar muy útil si lo utilizamos en el mismo ámbito que el caso anterior, si realizamos una malla de etiquetas nfc pegadas en el suelo podemos dotar al robot de la capacidad de conocer en muchos puntos de la nave de muchísima información.

#### 4.2.4. Elección

De entre todos estos sistemas en principio descartamos el guiado mecánico porque a pesar de su robustez no nos da la flexibilidad que necesitamos y además su instalación sería más costosa que en otros casos.

Entre las cintas negras y la malla de etiquetas en el suelo resulta una obviedad que el segundo sistema posee muchísimas más posibilidades en el diseño de la instalación ya que además de servirnos para el seguimiento del recorrido deseado, si la “resolución” de la malla es relativamente grande el robot sabe perfectamente donde está en todo momento, nos da la capacidad de recibir información cada vez que el robot pasa por encima de una etiqueta.

Por motivos de sencillez y de magnitud del proyecto nos decantaremos por la opción de las líneas negras en el suelo.

### 4.3. Sensores para reconocimiento de la línea

Para que el robot sea capaz de reconocer la línea necesitamos sensores que sean sensibles al cambio de color en el suelo.

Los sensores más populares en ese aspecto son los fototransistores con un fotodiodo acoplado. El diodo emitirá luz constantemente y ésta rebotará en el suelo que según sea éste negro o no absorberá más radiación lumínica o menos, llegando a los fototransistores más o menos intensidad lumínica y excitándose estos últimos en mayor o menor medida. En el mercado existen diferentes soluciones para esto.

Pablo Ferro Laspidea

La primera y más obvia es la utilización de fotodiodos y fototransistores separados y colocados en una posición cercana. Esta posibilidad se descarta ya que existen sensores con el diodo y el fototransistor integrados en un mismo encapsulado garantizando la proximidad entre ellos y que además son muy accesibles económicamente.

Entre todas las opciones disponibles encontramos 2 alternativas destacadas y validas que son el cny70 (poner fabricante y alguna característica precio) y el tcr1000 (poner fabricante y alguna característica precio). Como podemos las prestaciones y utilidad de ambos sensores son muy similares.

Por ultimo encontramos la posibilidad de utilizar arrays de integrados de sensores ya implementados con las resistencias y conexiones necesarios. Existen de diferentes tecnologías, con una disposición de los sensores variada y con distintos número de sensores. Esta es la opción más fácil y más cómoda ya que solo es necesario conectar y obtener la tensión del sensor.

Nos decantaremos por los sensores integrados ya que son más cómodos y eficientes que los sensores por separado, la diferencia económica no es un factor relevante, y porque los arrays de sensores son muy útiles y cómodos pero nos viene prefijado tanto el número de sensores como la disposición de los mismos. Entre el cny70 y el tcr1000 debido a la similitud en sus características nos decantaremos por el tcr1000 por el hecho de que se encuentra disponible para su utilización en los laboratorios de la UPNA.

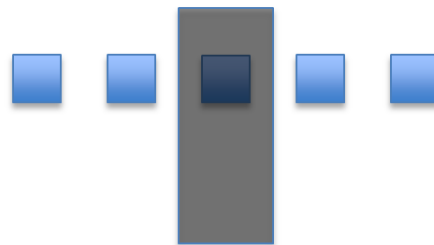
#### 4.4. Disposición, situación y número de sensores

En el caso más sencillo solo necesitaríamos saber si el robot está sobre la línea o si está a la izquierda o a la derecha de la línea bastándonos un solo sensor, para mandar la señal tanto al motor derecho como al izquierdo para corregir la desviación, mediante un algoritmo de control muy básico y rudimentario.

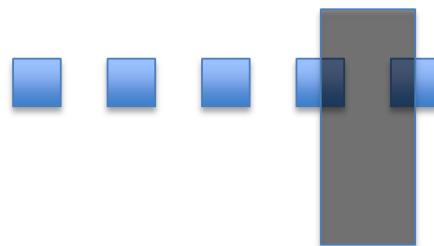
Sin embargo para controlar el posicionamiento del robot respecto de la línea de una forma más fiel y exacta deberemos aumentar el número de sensores de nuestro robot. En principio a mayor número de sensores mayor información de la posición de la línea y por lo tanto un control más preciso sobre el movimiento del robot. La tener

Pablo Ferro Laspidea

varios sensores podremos cuantificar el error ya que sabremos según que sensores detecten la línea como de desviado está el robot. Debido a limitaciones de espacio en la placa decidimos que colocando 5 sensores obtenemos suficiente información estando el sensor central sobre la línea cuando el robot está en posición correcta



Robot centrado en la línea



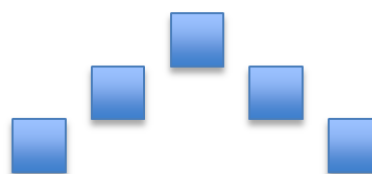
Robot perdiendo trayectoria deseada

### Ilustración 2: Seguimiento de la línea

Los sensores se colocaran en la parte delantera del robot estando las ruedas motrices en la parte trasera del mismo y estando a una distancia de unos 2 mm del suelo para obtener un mejor comportamiento de los mismo tal y como nos indican en la hoja de características.

Esta configuración se realiza de este modo para que el robot obtenga la información sobre la línea lo antes posible y que a las ruedas estén en situación de corregir la desviación de manera satisfactoria.

En cuanto a la disposición de los sensores en la placa existen diferentes alternativas según las necesidades de nuestro robot. Podemos colocarlos en línea o en V. La primera configuración se suele utilizar en los robots velocistas para detectar antes las curvas y la configuración en V con los sensores centrales más adelantados se suele utilizar en los robots rastreadores para detectar mejor la curva.

Configuración en línea

Configuración en V

**Ilustración 3: Disposición de los sensores**

En el caso de nuestro robot la velocidad no es un requerimiento necesario ya que un robot de un almacén no puede ir demasiado rápido. Adoptaremos la configuración en línea debido a que, como veremos más adelante, las intersecciones en nuestro circuito serán a 90°.

## 4.5. Tratamiento de la señal de los sensores

Una vez que hemos elegido la forma en la que vamos a guiar a nuestro robot, el tipo de sensores que vamos a utilizar y que hemos concretado cual vamos a usar se nos plantea la cuestión de que vamos a hacer con la señal de los sensores y como la vamos a tratar hasta utilizarla para el control del robot. En este punto existen diversas alternativas que vamos a explicar a continuación:

### 4.5.1. Utilización señal analógica

Se trata de utilizar la señal “íntegra” del sensor, convirtiéndola en señal digital que podamos leer y utilizar mediante un conversor analógico digital. Es la alternativa que nos proporciona una mayor información de la posición del robot ya que podemos discernir según la intensidad de la señal si el sensor está justo encima de la línea o mitad de la línea mitad suelo o fuera de la línea.

### 4.5.2. Utilización de señal digital

Se trata de pasar la lectura del sensor en tensión analógica a un nivel digital, es decir HIGH o LOW. Esta opción nos proporciona una menor información ya que no



discierne si estamos encima de la línea o a medias. Para su implementación se podría diseñar circuitos con amplificadores operacionales con una tensión de referencia que nos proporcionen una señal digital a la salida o también la utilización de integrados como un schmitt trigger inversor que a partir de un determinado nivel de señal nos invierte al nivel lógico contrario.

#### 4.5.3. Elección

A pesar de que obtenemos menor información nos decantaremos por la opción de utilizar la señal de los sensores a un nivel digital ya que consideramos que para la complejidad del control del seguimiento de la línea es suficiente con conocer si estamos encima o no de la misma y además el algoritmo de control, como veremos más adelante es más sencillo que si utilizáramos el valor analógico.

Además utilizaremos un integrado para convertir la señal analógica en digital ya que si acondicionamos de manera correcta los sensores nos aseguramos que la tensión analógica si estamos sobre negro o sobre blanco sea bastante diferente por lo que si establecemos high o low a partir de un determinado valor no estaremos cometiendo un error considerable salvo en los casos en los que el sensor se encuentre mitad negro mitad blanco ya que en ese caso la tensión proporcionada por el sensor si será un valor intermedio entre el high y el low y el valor que obtengamos no representara la realidad como si lo hubiéramos pasado a digital a través de un conversor.

## 4.6. Microcontrolador

Para interpretar los datos de los sensores, ejecutar el algoritmo de control y recibir las ordenes inalámbricas y actuar en consecuencia necesitaremos un sistema de gobierno que nos gestione todas estas tareas, en definitiva necesitaremos un microcontrolador.

### 4.6.1. ¿Qué es?

Un microcontrolador es un circuito integrado capaz de recibir, interpretar y generar señales digitales internas y/o externas. Son utilizados para controlar sistemas de una forma versátil, en un espacio reducido y a un coste muy asequible.

Un microcontrolador está compuesto por:

- Unidad Central de proceso (CPU):

Es lo más importante, es el corazón del microcontrolador desde donde se interpretan y ejecutan las instrucciones controlando el resto de bloques del mismo. El microprocesador puede ser de 4, 8, 32 y 64 bits con diferentes configuraciones entre la memoria de datos, la de programa y los buses. Los más usados y típicos suelen ser de 8 bits.

- Memoria: la memoria del microcontrolador se divide en memoria de programa y memoria de datos.
  - Memoria de programa: es donde se almacena las instrucciones en código máquina que el microprocesador ejecutara. Existen distintos tipos de memoria como son FLASH,ROM, EPROM,EEPROM... y se diferencian si son solo de lectura, si se pueden programar, borrar...En principio la memoria de programa es solo de es solo de lectura
  - Memoria de datos: es la memoria que utiliza el microprocesador para guardar datos, ya sean temporales o permanentes. Típicamente es de lectura y escritura, RAM.
- Buses: Caminos que unen las diferentes partes del sistema. A través de ello circula la información en forma de señales eléctricas. Se dividen en bus de direcciones, bus de datos y bus de control.
- Modulo entradas y salidas: es donde el sistema se relaciona con su entorno, donde se conectan todos los periféricos. Está compuesto por diferentes puertos; Puerto esclavo paralelo (PSP), puertos de entrada/salida...
- Otros módulos:
  - Temporizadores/contadores
  - Comunicación serie:
    - Transmisor Receptor Asíncrono Síncrono Universal (USART ó SCI)
    - Puerto Serie Síncrono Básico o Maestro (BSSP ó MSSP)
  - Módulo de Captura / Comparación / PWM:
  - Conversión Analógica / Digital (A/D)
  - Comparador analógico

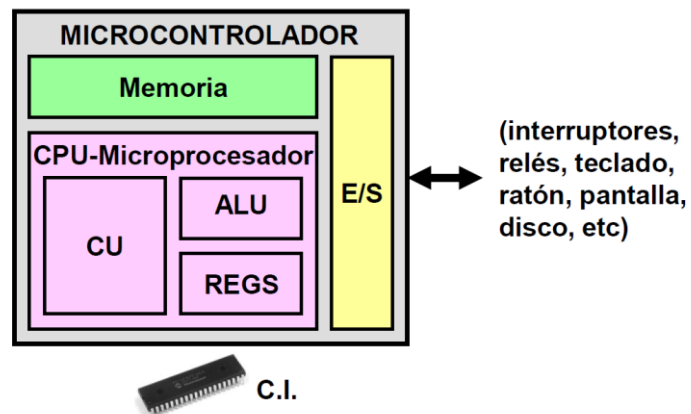


Ilustración 4: Estructura de un microcontrolador

#### 4.6.2. Alternativas en el mercado

En el mercado encontramos 2 alternativas diferentes entre sí pero que destacan sobre el resto, los microcontroladores PIC de la empresa Microchip Technology Inc. y el entorno de programación Arduino. En este proyecto nos decantaremos por el segundo.

#### 4.6.3. Arduino

Arduino es una plataforma electrónica enfocada a la creación de prototipos mediante el uso de hardware y software dinámicos y con una curva de aprendizaje asequible. Es una plataforma open source, es decir, es una plataforma libre, se puede utilizar para hacer cualquier proyecto sin la necesidad de tener ninguna licencia y se puede modificar libremente. El lenguaje de programación Arduino está basado en Wiring y el entorno de desarrollo en Processing

Name	Processor	Operating Voltage/Input Voltage	CPU Speed	Analog In/Out	Digital IO/PWM	EEPROM [KB]	SRAM [KB]	Flash [KB]	USB	UART
Uno	ATmega328	5 V/7-12 V	16 Mhz	6/0	14/6	1	2	32	Regular	1
Due	AT91SAM3X8E	3.3 V/7-12 V	84 Mhz	12/2	54/12	-	96	512	2 Micro	4
Leonardo	ATmega32u4	5 V/7-12 V	16 Mhz	12/0	20/7	1	2.5	32	Micro	1
Mega 2560	ATmega2560	5 V/7-12 V	16 Mhz	16/0	54/15	4	8	256	Regular	4
Mega ADK	ATmega2560	5 V/7-12 V	16 Mhz	16/0	54/15	4	8	256	Regular	4
Micro	ATmega32u4	5 V/7-12 V	16 Mhz	12/0	20/7	1	2.5	32	Micro	1
Mini	ATmega328	5 V/7-9 V	16 Mhz	8/0	14/6	1	2	32	-	-
Nano	ATmega168	5 V/7-9 V	16 Mhz	8/0	14/6	0.512	1	16	Mini-B	1
	ATmega328		1			2	32			
Ethernet	ATmega328	5 V/7-12 V	16 Mhz	6/0	14/4	1	2	32	Regular	-
Esplora	ATmega32u4	5 V/7-12 V	16 Mhz	-	-	1	2.5	32	Micro	-
ArduinoBT	ATmega328	5 V/2.5-12 V	16 Mhz	6/0	14/6	1	2	32	-	1
Fio	ATmega328P	3.3 V/3.7-7 V	8 Mhz	8/0	14/6	1	2	32	Mini	1
Pro (168)	ATmega168	3.3 V/3.35-12 V	8 Mhz	6/0	14/6	0.512	1	16	-	1
Pro (328)	ATmega328	5 V/5-12 V	16 Mhz	6/0	14/6	1	2	32	-	1
Pro Mini	ATmega168	3.3 V/3.35-12 V	8 Mhz	6/0	14/6	0.512	1	16	-	1
		5 V/5-12 V	16Mhz							
LilyPad	ATmega168V	2.7-5.5	8 Mhz	6/0	14/6	0.512	1	16	-	-
		V/2.7-5.5 V								
LilyPad USB	ATmega32u4	3.3 V/3.8-5V	8 Mhz	4/0	9/4	1	2.5	32	Micro	-
LilyPad Simple	ATmega328	2.7-5.5	8 Mhz	4/0	9/4	1	2	32	-	-
		V/2.7-5.5 V								
LilyPad SimpleSnap	ATmega328	2.7-5.5 V/2.7-5.5 V	8 Mhz	4/0	9/4	1	2	32	-	-

Ilustración 5: Tabla comparativa de todas las placas Arduino (web oficial)

La plataforma Arduino presenta múltiples ventajas:

- Accesible: una placa Arduino puede ser adquirida por un precio muy razonable 20-30 euros y además están disponibles en multitud de tiendas y portales en internet. Si fuera el caso, te puedes construir tu propia placa ya

que es una plataforma libre y los planos y esquemáticos están colgados en su página web.

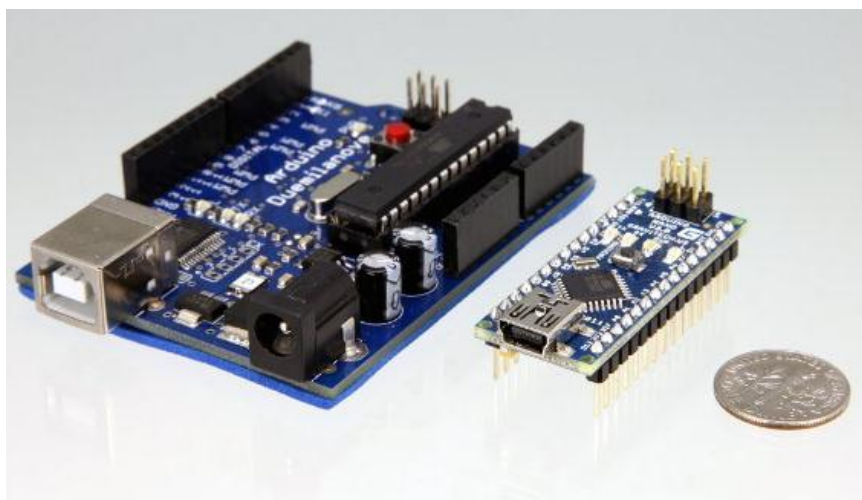
- Plataforma libre: el hecho de que sea open source hace que no sea necesario ninguna licencia para su uso y que se puede modificar cualquier componente sin ningún problema.
- Comunidad: el hecho de que sea una plataforma libre hace que exista una gran comunidad de usuarios detrás, desde profesionales del sector a usuarios por hobby personal. Esta comunidad aporta multitud de ventajas ya que se pueden encontrar múltiples consejos, alternativas, foros con dudas, tutoriales y existe una gran cooperación entre los usuarios de Arduino
- Multidisciplinar: los productos Arduino se pueden usar en campos tan diversos como la robótica, automatización de sistemas, programación de otros microcontroladores, control de actuadores... Además se usa desde un nivel profesional o de empresa hasta en pequeños robots para la iniciación de los jóvenes en la programación.

Dentro de la gama de productos de la familia Arduino podemos encontrar múltiples y diferentes alternativas tanto de placas, como de shields y de accesorios. Si nos atenemos a nuestro proyecto, necesitamos una placa de reducido tamaño, con un módulo de entradas y salidas considerable, con un puerto serie UART y que trabaje a 5V.

La placa más utilizada y adecuada para proyectos estándar en los que no se necesitan excesivas entradas/salidas y en los que el tamaño de la placa no es un factor crucial es la placa Arduino UNO. Dentro de las placas de reducido tamaño y de propósito general encontramos la Arduino Nano y la Arduino Pro Mini. A continuación vamos a comparar estas 2 placas con la Arduino Uno para poder analizar cual se adapta mejor a nuestras necesidades.

	Arduino Uno	Arduino Nano	Arduino Pro Mini
Procesador	ATmega328	ATmega328	ATmega168
Voltaje operación/alimentación	5V/ 7-12V	5V/7-9V	5V/7-12V
Velocidad CPU	16MHz	16MHz	16MHz
Analog IN/OUT	6/0	8/0	6/0
Digital IO/PWM	14/6	14/6	14/6
USB	Normal	Mini-B	-
UART	1	1	1

Como se puede observar en la tabla las características generales de estas 3 placas son bastante similares aunque en principio descartamos la placa Arduino Pro Mini ya que carece de una interfaz comunicación serie USB por lo que sería necesario utilizar hardware adicional para su programación. Entre las 2 placas restantes elegimos la Arduino Nano por cuestiones de tamaño ya que es sensiblemente más pequeña que la Arduino Uno.



**Ilustración 6: Comparación de tamaño entre Arduino Uno, Arduino Nano y una moneda**

## 4.7. Motores

### 4.7.1. Alternativas

Todo robot necesita de un sistema capaz de producir movimiento, ese sistema es el motor que en el caso de los robots se trata de un motor eléctrico.

Un motor eléctrico es un sistema que transforma la energía eléctrica en energía mecánica por medio de la interacción de un campo magnético con sus bobinas. La mayoría de ellos están formados por un estator y un rotor. Existen diferentes tipos de motores eléctricos cada uno con una estructura diferente que determinara la interacción de los flujos magnéticos que producirán la fuerza o par del motor. A continuación se resumen de forma muy breve los tipos de motores eléctricos más importantes:

- Motor corriente continua:

Están formados principalmente por 2 partes: el rotor y el estator. El estator es la parte que no se mueve y es donde se encuentran los devanados o

polos del motor que pueden ser imanes o devanados de cobre sobre un núcleo de hierro. Además el estator aporta la rigidez y soporte al motor.

El rotor, cilíndrico en la mayor parte de las ocasiones, también está devanado y tiene un núcleo. A través de él pasa la corriente continua mediante escobillas fijas.

El mayor inconveniente de estos motores es el mantenimiento que es caro y complicado, debido principalmente al desgaste de las escobillas con el rozamiento. Existen también motores DC sin escobillas, conocidos también como brushless. Se utilizan para hacer motores paso a paso y servomotores.

- Motor corriente alterna:

Un motor de corriente alterna al igual que el motor DC transforma la energía eléctrica en energía mecánica pero a diferencia del caso anterior es alimentado por corriente alterna. Existen diferentes tipos de motores AC:

- Universal
- Asíncrono
- Síncrono
- De jaula de ardilla.

- Motor paso a paso:

Es un dispositivo electromecánico que convierte impulsos eléctricos en desplazamientos angulares discretos. Lo que se mueve en grados (pasos) depende de sus entradas de control que son impulsos procedentes de sistemas lógicos. Su principal ventaja es la alta precisión y repetitividad en cuanto a la posición del rotor. Existen principalmente 3 tipos: motor de reluctancia variable, motor de magnetización permanente y motor paso a paso híbrido.

- Servomotor:

Un servomotor es un motor al que se le ha añadido un sistema de control, un potenciómetro y una caja de engranajes, tiene la capacidad de ubicarse en cualquier ángulo de su rango operativo pudiendo controlarse tanto en velocidad como en posición. Se puede pasar de un servomotor a un motor dc conservando la fuerza, velocidad y baja inercia del servomotor.

#### 4.7.1. Elección

Debido a que no necesitamos precisión alguna en la posición del rotor de nuestros motores descartamos utilizar servomotores. Además como necesitamos unos cambios de velocidad rápidos y precisos nos decantaremos por un motor dc. En el mercado existen múltiples motores dc de tamaño muy reducido que ofrecen pares y velocidades que cumplen con nuestras necesidades.

Analizaremos 3 modelos de micro motor dc de la tienda bricogeeek que son bastante populares en robots de dimensiones reducidas, que tienen el mismo precio (13'20€) y se diferencian en la caja reductora que tienen.

Dimensiones	24 x 10 x 12 mm
Diámetro del eje	3,9mm
Voltaje nominal	6Vcc (puede funcionar entre 3 a 9Vcc)
Peso	10 gramos
Consumo sin carga/máximo	40mA (Max: 360mA)

A mayor factor de reducción mayor es el par que ofrece de salida sacrificando la velocidad angular del rotor. Buscaremos un equilibrio entre ambos parámetros ya que no necesitamos un par especialmente grande ya que el robot es bastante ligero y tampoco queremos una velocidad angular demasiado alta ya que sería más ingobernable y además no es una característica importante para un robot de almacén.

	Micro metal DC 10:1	Micro metal DC 30:1	Micro metal DC 50:1
Caja reductora	10:1	30:1	50:1
Par o torque salida	0,2 kg-cm (Max)	0,3 kg-cm (Max)	0,4 kg-cm (Max)
Velocidad angular Sin carga	1250rpm	440rpm	250rpm

Debido a que, como he comentado ya, buscamos un compromiso nos decantaremos por el modelo con reductora 30:1 ya que sus prestaciones son bastante razonables.



Pablo Ferro Laspidea

La velocidad máxima que alcanzará el robot, con ruedas de 32mm como comentaremos más adelante, será:

$$v = \omega \cdot R = 440 \frac{\text{rev}}{\text{min}} \cdot \frac{\text{min}}{60\text{s}} \cdot \frac{2\pi \text{ rad}}{\text{rev}} \cdot \frac{32 \text{ mm}}{2} \cdot \frac{1\text{m}}{1000\text{mm}} = 0.737 \text{ m/s}$$

Una velocidad máxima de 0.737m/s es más que suficiente para lo que necesitamos.

## 4.8. Driver motor

La placa Arduino Nano, al igual que todas las placas Arduino, no es capaz de aportar la intensidad necesaria para alimentar los 2 motores de nuestro robot ya que la corriente de salida máxima por un pin E/S es de 40mA, claramente insuficiente ya que ese es el consumo del motor sin carga.

Así pues necesitaremos un hardware adicional que nos proporcione una intensidad superior y que tenga una estructura capaz de proporcionar un control de la velocidad variable y que los motores puedan girar en los 2 sentidos.

Para que el motor sea capaz de girar en los 2 sentidos necesitamos una estructura como la del puente en H.

Los puentes en H son ampliamente utilizados en robótica ya que están disponibles en circuitos integrados muy asequibles. Está compuesto por 4 interruptores, típicamente transistores, que van conmutando estando S1 y S4 cerrados (S2 Y S3 abiertos) y el motor girando en un sentido y S2 Y S3 cerrados (S1 y S4 abiertos) y el motor girando en sentido contrario.

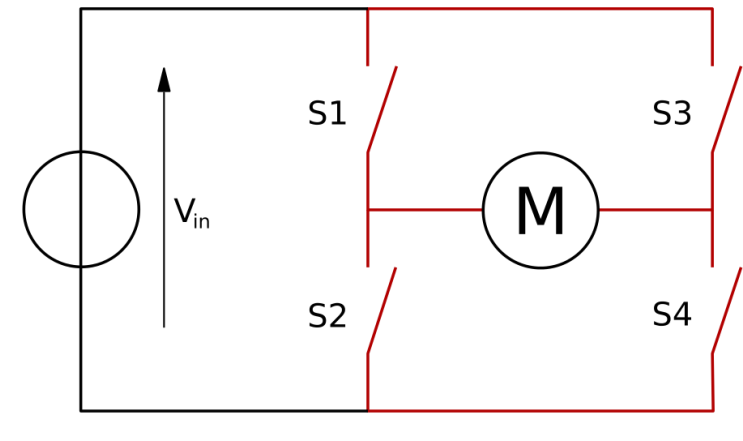


Ilustración 7: Estructura de puente en H

Además necesitaremos regular la velocidad de giro de los motores, para ello utilizaremos la técnica de PWM. Pero, ¿qué es PWM? La modulación por ancho de pulso (Pulse Width Modulation) de una señal eléctrica es una técnica que consiste en variar el ciclo de trabajo de una señal periódica controlando así la cantidad de energía que enviamos a una carga determinada.

¿Qué es el ciclo de trabajo de una señal? El ciclo de trabajo,  $D$ , de una señal periódica, cuadrada por ejemplo, es la relación entre la parte HIGH de la señal respecto a su periodo total.

$$D = \frac{\tau}{T}$$

Donde:

$T$ : periodo de la señal

$\tau$ : tiempo en alto de la señal

De este modo, el ciclo de trabajo siempre será menor que la unidad. Lo bueno de esta técnica de regulación de velocidad es que mantiene el par constante ya que la tensión que le llega al motor es constante, aunque si que modifique la tensión media que le llega.

Un integrado muy interesante y accesible es el L293 de Texas Instruments ya que está formado por 4 medios puente que se activan por parejas formando cada una un

Pablo Ferro Laspidea

puente completo H, así pues tenemos 2 puentes completos H que se activan de forma independiente.

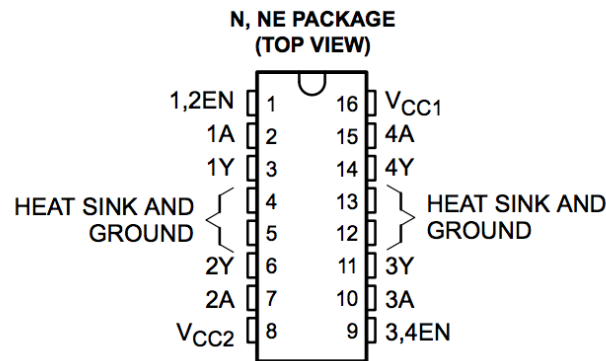


Ilustración 8: Diagrama pines del integrado L293D

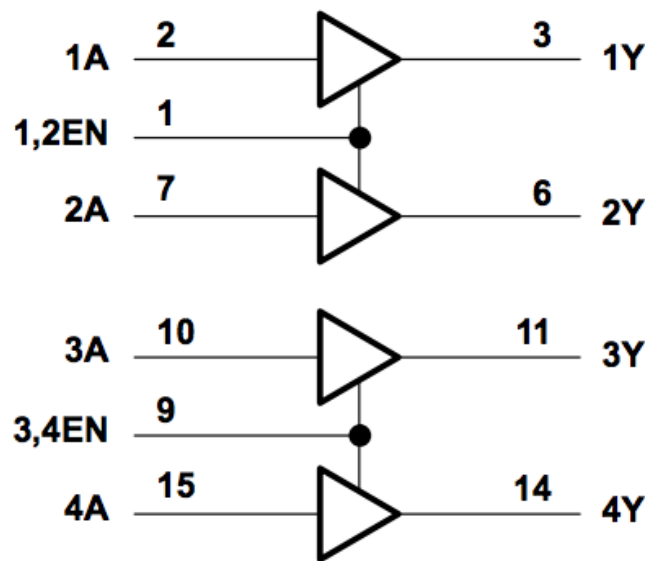


Ilustración 9: Diagrama lógico entradas y salidas del 293D

De este modo podemos hacer que los motores giren en ambos sentidos, al tener la estructura puente en H, y además podemos controlarlos de manera independiente mandando una señal PWM al enable de las 2 parejas de medios puentes. La versión L293D viene con diodos incorporados que nos evitan que corrientes parásitas nos dañen el integrado. Este integrado es capaz de proporcionar 600mA de salida y tensiones desde 4.5V hasta 36V, además tiene lógica TTL por lo que es compatible

Pablo Ferro Laspidea

con la mayoría de los microcontroladores. Consideramos que el driver 293D es bastante óptimo para lo que necesitamos.

## 4.9. Ruedas

Al principio del proyecto se plantearon varias cuestiones acerca del número, tipo y posicionamiento de las ruedas del robot. Las alternativas que se barajaron fueron las de utilizar 2 o 4 ruedas.

En la configuración de 2 ruedas, el giro se realizaría por la diferencia de velocidad entre ambas ruedas, además precisaríamos de una bola o rueda loca para aportar sostenibilidad al chasis del robot y que éste no rozara con el suelo. Para la configuración de 4 ruedas se podría girar por el método de velocidad diferencial como con 2 ruedas, o con algún sistema de dirección en las ruedas que no son motoras. Este último caso se descartó desde el principio debido a la complejidad extra innecesaria que añadía al diseño del robot.

Así pues se decidió por la configuración de 2 ruedas con una bola loca, ya que las ruedas locas a veces no giran de una forma completamente libre y nos podría ocasionar problemas en los giros.



**Ilustración 10: Bola loca y rueda loca**

Las 2 ruedas motoras se colocarán en la parte posterior del robot, colocándose los sensores en la parte delantera del robot y la rueda loca en un punto intermedio entre ambos.

Pablo Ferro Laspidea

En cuanto a las dimensiones de las ruedas, se descartó desde un primer momento la utilización de ruedas con diámetros grandes ya que los sensores debían estar a una distancia de entre 1 y 4 mm hasta el suelo siendo más beneficioso cuanto menor fuera esta distancia.

Así pues se utilizaron 2 ruedas, de 32mm de diámetro, y una bola loca que se encontraban disponibles en el laboratorio. La anchura de las mismas era un aspecto ciertamente irrelevante en el diseño del robot.

## 4.10. Comunicación inalámbrica

### 4.10.1 Alternativas

Debido a que desde el principio del proyecto se quería que estuviera enfocado a funcionar en un almacén la comunicación inalámbrica con el robot desde otro dispositivo remoto era una funcionalidad que no podía faltar.

En principio se valoraron 3 tipos de tecnología a utilizar:

- ZigBee
- Bluetooth
- Radiofrecuencia

Desde un principio se descartó la utilización de un módulo RF ya que el alcance era menor que el de las otras 2 alternativas y además presentaba menos posibilidades para interactuar con otros sistemas.

El IEEE, instituto de Ingeniería Eléctrica y Electrónica, es una asociación mundial en la que ingenieros y técnicos se dedican a la estandarización y desarrollo en áreas técnicas. Esta institución clasifica las diferentes tecnologías inalámbricas en determinados estándares según la frecuencia a la que trabajan y el alcance que soportan.

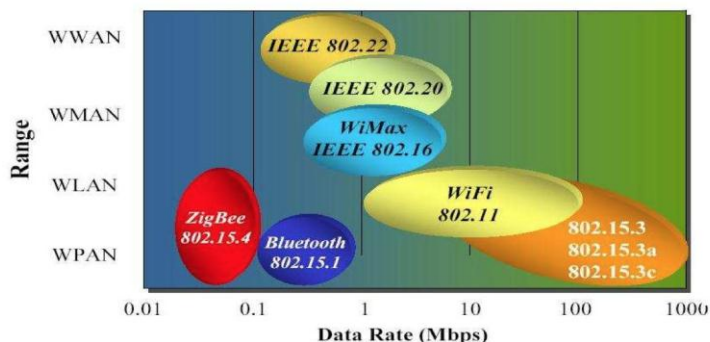


Ilustración 11: Clasificación estándares de comunicación

- WPAN: 802.15.1 (Bluetooth), 802.15.3 (UWB) y 802.15.4 (Zigbee),
- WLAN: 802.11 (WiFi)
- WMAN: 802.16 (WiMax)
- WWAN: 802.22 y 802.20

El estándar que nos interesa a nosotros es WPAN (Wireless Personal Area Networks) ya que abarca redes cuyo alcance es de unos pocos metros y eso es lo que necesitamos para nuestro robot.

A continuación detallaremos muy brevemente en qué consisten estas 2 tecnologías

### Bluetooth

Es la más utilizada y popular en este campo, como consecuencia de su flexibilidad ha tenido una gran expansión estos últimos años y hoy en día existen multitud de dispositivos con tecnología bluetooth incorporada tales como smartphones, cámaras de fotos, impresoras...

Su alcance está entre 1m y 100m, los dispositivos bluetooth se agrupan en piconets que están formadas por dispositivos que se conectan sobre la marcha. En una piconet con 2 dispositivos, uno ejerce de maestro y el otro de esclavo.

Los dispositivos bluetooth se pueden clasificar en:

- Clase 1: 100mW, 100m
- Clase 2: 2.5mW, 10 m
- Clase 3: 1mW, 1m

Además existen diferentes versiones de bluetooth:

- Versión 1.2 (1 Mbps)
- Versión 2.0 (3 Mbps)
- Versión 3.0:24 Mbps
- Versión 4.0 (smart bluetooth)

### Zigbee

Es un conjunto de protocolos de alto nivel de comunicación inalámbrica basado en el estándar IEE 802.15.4 WPAN. Es muy útil para comunicaciones con una tasa de envío de datos baja y con un consumo reducido. Su topología de red es en malla. Soporta un máximo de 65535 nodos distribuidos en subredes de 255 nodos, frente a los ocho máximos de una subred Bluetooth

Frecuencia	Banda	Cobertura	Datos	# de canales	Sensitividad Rx	Modulación
2.4 GHz	IMS	Mundial	250 Kbps	16	-85 dBm	O-QPSK
868 MHz		Europa	20 Kbps	1	-92 dBm	BPSK
915 MHz	IMS	America	40 Kbps	10	-92 dBm	BPSK

**Ilustración 12: Diferentes topologías de ZigBee**

Además posee un menor consumo eléctrico frente al Bluetooth. En términos exactos, ZigBee tiene un consumo de 30 mA transmitiendo y de 3 µA en reposo, frente a los 40 mA transmitiendo y 0,2 mA en reposo que tiene el Bluetooth. Puede soportar una velocidad de hasta 250 kbit/s.

Pablo Ferro Laspidea

La velocidad del ZigBee se hace insuficiente para tareas que requieran una gran tasa de envío de datos, centrándose su uso en aplicaciones como la Domótica por ejemplo.

#### 4.10.2 Elección

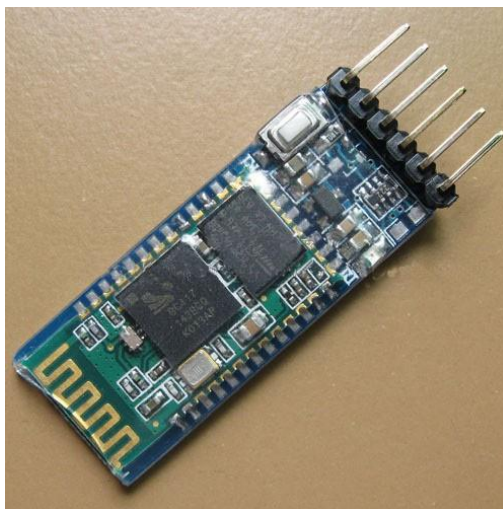
En un principio, se optó por utilizar la tecnología ZigBee por las ventajas que presenta respecto a Bluetooth y porque se trataba de una tecnología completamente desconocida para mí. En el mercado hay diferentes alternativas disponibles para utilizar en el proyecto, aunque los módulos más populares en este tipo de proyectos son los módulos XBee de la casa MaxStream.

Se estuvo investigando sobre la tecnología y su uso mirando foros y tutoriales, y se intentó configurar un módulo XBee que se encontraba disponible en el laboratorio con el programa X-CTU pero hubo muchos problemas con los drivers y con la imposibilidad de configurarlo. Investigando sobre el error que daba el programa cuando se intentaba configurar el módulo, se llegó a la conclusión de que el módulo estaba bloqueado y que había que realizar complejas operaciones con el firmware para su desbloqueo. Al final se optó por utilizar la tecnología Bluetooth que también satisfacía nuestras necesidades.

Se investigó que soluciones había disponibles para implementar la tecnología bluetooth en nuestro robot y se optó por utilizar el módulo chino JY-MCU debido principalmente a 2 motivos:

- Su bajo precio: su precio, 7.36\$, es menor al que podemos encontrar en otros módulos de la competencia aunque si es verdad que los módulos más caros presentaban mejores características y una mayor fiabilidad.
- Su popularidad: este módulo es muy famoso porque es muy accesible y funciona. Tiene una gran comunidad de usuarios detrás lo que garantiza el poder resolver las dudas que nos surgieran ya que nunca había configurado y utilizado un módulo bluetooth previamente.





**Ilustración 13: Módulo Bluetooth JY-MCU**

Observando con la perspectiva del tiempo, elegir la tecnología ZigBee para implementar la conectividad del robot fue un tremendo error ya que supuso una pérdida de tiempo enorme, con el consecuente retraso en el desarrollo del proyecto. Además, como ya se ha comentado, la tecnología Bluetooth satisfacía nuestras necesidades siendo además una opción considerablemente más económica.

#### 4.11. Sensor distancia

Un robot que funciona de forma autónoma tiene que tener un sensor que le alerte de la presencia de una persona o de otros objetos para evitar un hipotético choque o accidente. Así pues desde el principio del desarrollo del proyecto se valoró la utilización de un sensor de distancia y tras investigar se valoraron 2 alternativas accesibles que cumplieran con lo que queríamos.

El sensor de infrarrojos Sharp GP2Y0A02YK proporciona una tensión continua proporcional a la distancia que detecta. Su rango de medición va desde los 20cm hasta los 150cm y funciona con una alimentación de 5V, siendo su consumo medio de funcionamiento de 32mA. Su precio en el mercado es de 20,45 euros.

La otra opción que se encontró fue utilizar un sensor ultrasónico como el HC-SR04. Este sensor funciona como un sonar, es decir emite ultrasonidos y espera hasta recibir el eco. Tiene un rango de medición de 2cm a 400cm, con una resolución de 0.3cm. Al igual que el otro sensor, también se alimenta a 5V pero su consumo es

Pablo Ferro Laspidea

menor ya que ronda los 15mA. Se puede encontrar fácilmente en el mercado por 3 euros.

Debido a su mayor rango de medida y a su menor consumo eléctrico, se eligió el sensor HC-SR04 para utilizarlo en el robot. El funcionamiento de este sensor consiste en que emite un pulso de 10 us por el terminal de trigger y recibe el eco por el terminal de echo poniéndose a HIGH cuando lo recibe. Lo único que se debe hacer, aparte de enviar el pulso, es contar el tiempo desde que se envía el pulso hasta que se recibe el eco. Una vez que se conoce cuanto tiempo ha tardado el pulso en ir y en volver solo debemos multiplicarlo por la velocidad del sonido y la distancia que se obtiene dividirla entre 2 ya que si no estaríamos obteniendo el doble de la distancia que queremos medir ya que la onda realiza tanto el camino de ida como el de vuelta.

Velocidad del sonido en cm/us:

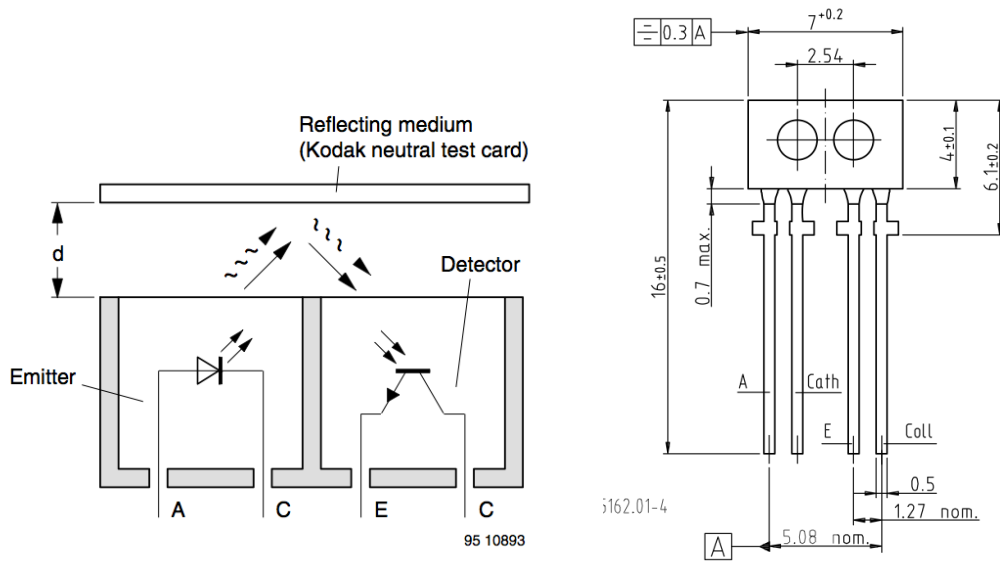
$$\frac{\text{Velocidad del sonido}}{2} = \frac{343m}{s} \cdot \frac{1s}{10^6\mu s} \cdot \frac{100cm}{1m} \cdot \frac{1}{2} = 0.01715cm/\mu s$$

## 5. Diseño

---

### 5.1. Acondicionamiento de los sensores

Como ya se ha comentado se utilizaran los sensores TCRT1000 para detectar la línea de una forma digital, es decir, o HIGH o LOW según este o no encima de la línea. Para ello se conectará la salida del sensor a un inversor schmitt trigger para que la salida de éste último sea 5V o 0V y desde el Arduino no sea necesario realizar ninguna conversión analógica digital



**Ilustración 14: Disposición de las patillas del fotodiodo y del fototransistor dentro del encapsulado TCRT1000**

Como el integrado 74HC14 invierte la señal a partir de cierto umbral, como se verá más adelante, y en el programa el negro se detecta como un “1” lógico se deberá diseñar el circuito de tal forma que el negro proporcione una señal cercana a 0V, para que el inversor de una salida de HIGH, y el blanco deberá dar una señal por encima del umbral del inversor para que la salida de éste sea de LOW.

Así pues el circuito diseñado para los sensores es el siguiente:

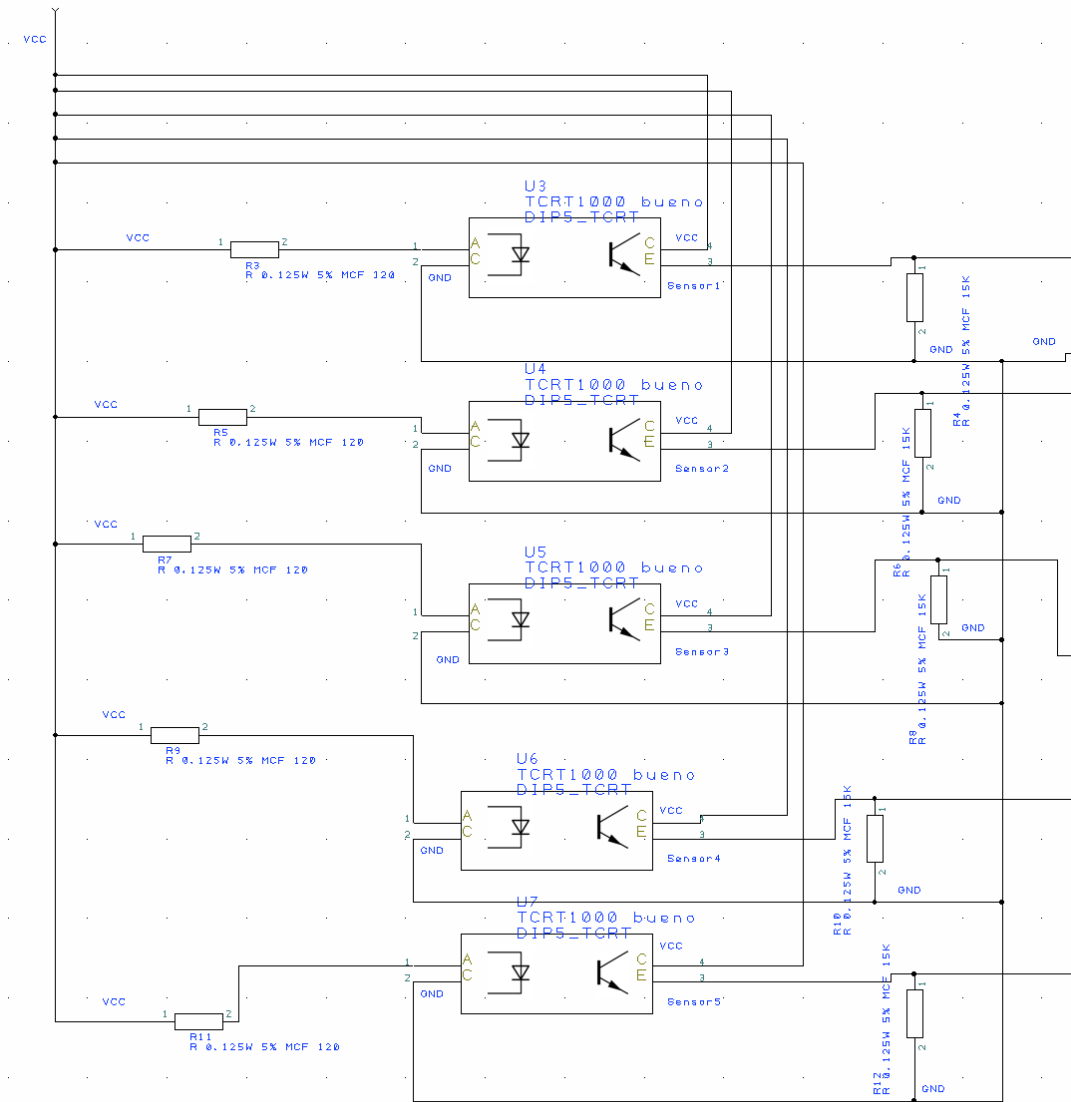


Ilustración 15: Esquemático circuito sensores TCRT1000

Como se puede observar el ánodo del fotiodo está conectado a una resistencia de 220Ω que a su vez está conectada a la net VCC (5V), y el cátodo del diodo está conectado a la net GND. Por otra parte el colector del fototransistor está conectado a la net VCC y el emisor está conectado a una resistencia de 15kΩ que está conectada a tierra. Las señales que nos interesan son las que están conectadas al emisor de cada encapsulado y que se llaman en el esquemático Sensor1, Sensor2, Sensor3, Sensor4 y Sensor5.

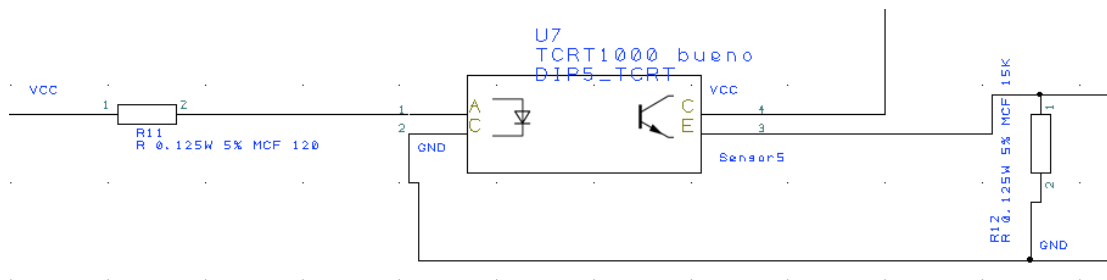


Ilustración 16: Esquemático en detalle de la conexión de un sensor TCRT1000

El encapsulado 74HC14 es un schmitt trigger que nos invierte la señal de la entrada a través de una ventana de histéresis que depende de la alimentación como se puede observar en las siguientes capturas tomadas de las hojas de especificaciones del integrado.

Symbol	Parameter	Conditions	T <sub>amb</sub> = 25 °C			T <sub>amb</sub> = -40 °C to +85 °C		T <sub>amb</sub> = -40 °C to +125 °C		Unit
			Min	Typ	Max	Min	Max	Min	Max	
<b>74HC14</b>										
V <sub>T+</sub>	positive-going threshold voltage	V <sub>CC</sub> = 2.0 V	0.7	1.18	1.5	0.7	1.5	0.7	1.5	V
		V <sub>CC</sub> = 4.5 V	1.7	2.38	3.15	1.7	3.15	1.7	3.15	V
		V <sub>CC</sub> = 6.0 V	2.1	3.14	4.2	2.1	4.2	2.1	4.2	V
V <sub>T-</sub>	negative-going threshold voltage	V <sub>CC</sub> = 2.0 V	0.3	0.52	0.9	0.3	0.9	0.3	0.9	V
		V <sub>CC</sub> = 4.5 V	0.9	1.4	2.0	0.9	2.0	0.9	2.0	V
		V <sub>CC</sub> = 6.0 V	1.2	1.89	2.6	1.2	2.6	1.2	2.6	V
V <sub>H</sub>	hysteresis voltage	V <sub>CC</sub> = 2.0 V	0.2	0.66	1.0	0.2	1.0	0.2	1.0	V
		V <sub>CC</sub> = 4.5 V	0.4	0.98	1.4	0.4	1.4	0.4	1.4	V
		V <sub>CC</sub> = 6.0 V	0.6	1.25	1.6	0.6	1.6	0.6	1.6	V

Ilustración 17: Voltajes de ciclo de histéresis según tensión alimentación

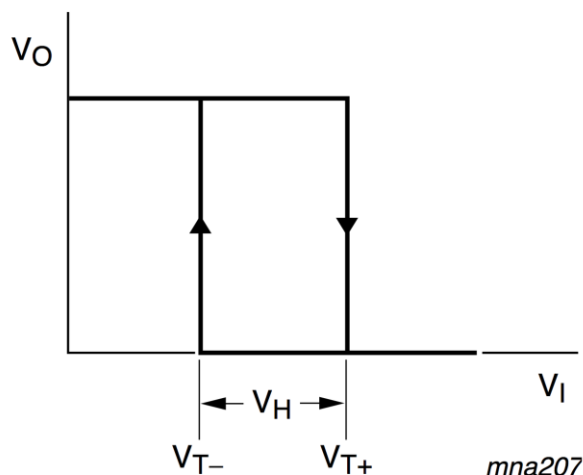


Ilustración 18: Ciclo de histéresis

El integrado estará alimentado por la net VCC que son 5V aproximadamente por lo que la ventana del ciclo de histéresis será de 1V aproximadamente. El límite inferior será 1.5V aproximadamente y el límite superior será 2.5V aproximadamente. Esto significa que para valores menores de 1.5V de entrada el integrado proporcionará una salida alta de 5V y para valores mayores de 2.5V proporcionara una señal baja de 0V a la salida.

Realizando pruebas y mediciones con los sensores en el laboratorio se comprobó que midiendo a la distancia a la que iban a estar los sensores del suelo en el robot, 5mm, sobre negro proporcionaba una señal de 0,5V-0,6V y sobre blanco proporcionaba una señal de 4,4V-4,5V. Como se puede observar que con esas resistencias se tiene un margen grande para posibles errores en las mediciones ya que los valores normales están bastante lejos de la ventana de histéresis, sobre todo los de lectura sobre blanco.

## 5.2. Determinación y cuantificación del error

Para que el robot siga la línea debe conocer en todo momento donde se encuentra ésta respecto a su posición. Según el número de sensores y la disposición en la que se coloquen se conseguirá más o menos información. En nuestro caso se eligió utilizar 5 sensores TCRT1000 dispuestos en línea recta. Tomaremos la situación ideal cuando solo lea la línea el sensor de en medio, esta situación será de error 0. Con las distintas combinaciones de lecturas de los sensores ponderaremos el error ya que no es lo mismo que el robot este ligeramente desviado a que este casi completamente fuera de la línea. Esta ponderación consistirá en asignar valores de error discretos a las distintas combinaciones de lectura, estos valores de error serán tanto positivos como negativos. La razón por la que se opta por utilizar valores positivos y negativos es porque de alguna manera se necesita distinguir si el robot está perdiendo la línea por la izquierda o por la derecha ya que la acción de control no puede ser la misma.

Pablo Ferro Laspidea

En definitiva lo que estamos haciendo es construir un sensor de posición compuesto más sofisticado a partir de sensores simples.

La ponderación del error será la siguiente:

Lectura sensores	Error
00001	4
00111	3
00011	3
00010	2
00110	1
00100	0
01100	-1
01000	-2
11000	-3
11100	-3
10000	-4
00000	5 -5 según valor anterior

Como se verá más adelante, el hecho de que el error tenga signo será bastante útil para realizar la acción de control.

### 5.3. Algoritmo de control

Una vez que el robot conoce que error tiene su ubicación respecto a la línea debe contar con un algoritmo que tenga en cuenta ese error y produzca una acción de control en consecuencia para corregirlo. Además sería conveniente que no solo tuviera en cuenta el error que tiene en ese momento si no que tuviera en cuenta el error anterior, es decir, como está evolucionando respecto al error. Así mismo sería una buena idea que controlara en todo momento el error que va acumulando para poder corregir desviaciones constantes. Estas acciones respecto al error se resumen en una estructura de control, el controlador PID.

Un controlador PID calcula la desviación que se tiene respecto a un valor objetivo, en este caso error 0, y aplica una acción de control para corregir y ajustar esta

Pablo Ferro Laspidea

desviación. Este algoritmo de control se puede dividir en tres partes o acciones correctoras: acción proporcional, acción derivativa y acción integral.

La acción de control proporcional es la que actúa sobre el error actual, que en este caso sería el valor discreto positivo o negativo que proporcionara el sensor compuesto por los 5 sensores simples. La acción de control derivativa actúa sobre la evolución del error en el tiempo. Por último la parte proporcional es la que actúa sobre la integral del error, es decir, sobre la suma de los errores de seguimiento.

La acción de control final será el resultado de sumar las contribuciones de estas 3 acciones correctivas. Cada proceso o acción a controlar tiene dinámicas muy diferentes por lo que la contribución de cada acción correctiva a la acción de control se determina a través del ajuste de tres variables que son las constantes del controlador,  $K_p$ ,  $K_i$  y  $K_d$ .

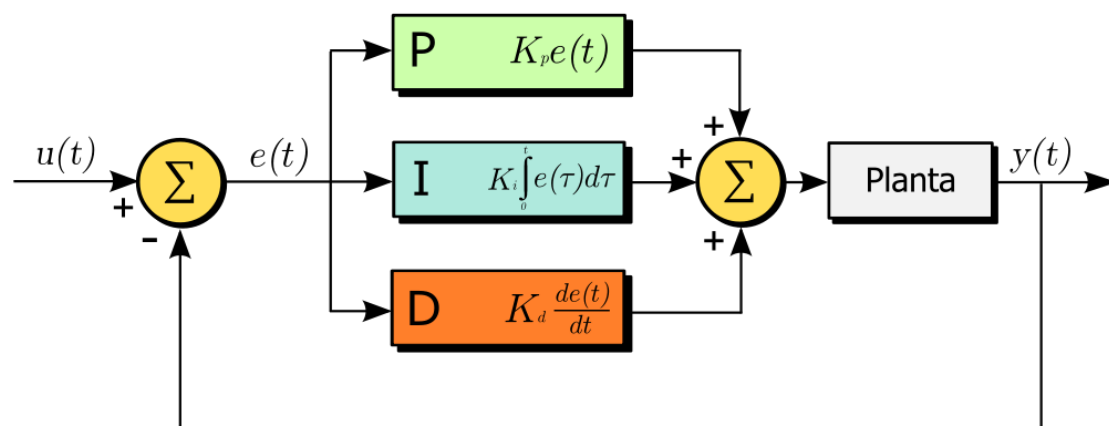


Ilustración 19: Estructura clásica del controlador PID

En este proyecto no se va a usar un controlador PID ya que se considera que no es necesaria tanta sofisticación en el controlador para corregir el error. En su caso se usará un PID que en esencia hace lo mismo, pero de una forma más simple. Para generar la acción de control se usará la siguiente estructura de control:

$$PID = K_p \cdot error + K_d \cdot derivada + K_i \cdot integral$$



Pablo Ferro Laspidea

El término error será el error que nos proporcione nuestro sensor compuesto, el término derivada será la resta del error actual menos el error anterior y por último el término integral será la suma del error actual + el término integral anterior.

Una vez escogido el algoritmo para realizar la acción de control, el siguiente paso es sintonizar las constantes, es decir, dar los valores necesarios para que la acción de control sea la adecuada para corregir el error. Generalmente la elección de los valores de estas constantes se realiza de forma experimental, existen algunos métodos sistemáticos que son a prueba y error pero que tienen un método con pasos a seguir. La ventaja de los ajustes experimentales es que no es necesario conocer las ecuaciones que rigen el comportamiento de la planta a controlar. Uno de los métodos de sintonización de constantes de forma experimental más famosos y más utilizados de la estructura de PID es el método de Ziegler–Nichols que consiste en poner las constantes derivativa  $K_d$  y proporcional  $K_p$  a cero, someter al sistema a un escalón de error constante e ir dando valores a la constante  $K_p$  hasta que el sistema oscile de forma constante ante cualquier perturbación, esta oscilación debe ser lineal, sin que se produzcan saturaciones. Una vez alcanzado este estado se habrá determinado cual es la ganancia crítica. Ziegler y Nichols desarrollaron unas tablas experimentales a partir de las cuales con el valor de la ganancia crítica  $K_p$  se podían obtener los valores de las otras 2 constantes,  $K_d$  y  $K_i$ .

En un primer momento la forma escogida para sintonizar las variables fue éste método y se empezaron a realizar ensayos experimentales variando la  $K_p$ , pero no se consiguió que el robot oscilara de forma estable ya que había ocasiones en las que parecía que era estable pero se volvía inestable. Además aun no siendo estable del todo se probó a utilizar esa ganancia para sacar los valores de las otras 2 a través de las tablas pero el resultado del control no fue satisfactorio ya que no seguía bien el circuito.

Tras este intento fallido se decidió determinar los valores de las constantes mediante un método de ensayo y error. Al igual que en el método de Ziegler–Nichols se pusieron las constantes  $K_d$  y  $K_i$  a cero y se empezó a dar valores a la  $K_p$  hasta que el robot más o menos seguía el circuito sin salirse en los tramos sencillos. Llegados a ese punto se empezó a dar valores a la  $K_d$  hasta perfeccionar el comportamiento del robot llegando a un nivel de respuesta casi excelente. Una vez sintonizadas las constantes  $K_p$  y  $K_i$  se procedió a ajustar la constante  $K_i$  empezando

Pablo Ferro Laspidea

con Ki igual a la unidad. El resultado fue que el seguimiento de la línea se fue al traste, perdía la línea nada más empezar a moverse.

Se llegó a la conclusión de que aun siendo igual a la unidad, el valor de la Ki era demasiado grande. Por ello se cambió el tipo de las variables a un tipo float con decimales y se volvió a hacer la prueba introduciendo valores bastante pequeños en torno a 0.01 pero el sistema seguía perdiendo la línea de forma demasiado evidente. Tras probar varias veces se tomó la decisión de descartar el uso de la parte integral del PID y dejarlo en un PD.

Una vez que está diseñado el algoritmo para calcular la acción de control hay que implementar la forma en la que se lo vamos a transmitir a los motores a través de una señal PWM. No se va a entrar en detalle ahora en el control de los motores pero hay que reseñar lo siguiente.

Si tenemos un error negativo significa que el robot está dejando la línea a su izquierda por lo que se deberá acelerar la rueda derecha y decelerar la rueda izquierda. Por el contrario si el error es positivo significa que el robot está dejando la línea a su derecha por lo que la rueda izquierda deberá acelerar y la rueda derecha decelerar. Para cumplir con estos requerimientos se implementará el siguiente algoritmo:

$$\text{Velocidad motor derecha} = \text{velocidad motor} - \text{PID}$$

$$\text{Velocidad motor izquierda} = 0.91 * (\text{velocidad motor} + \text{PID})$$

Velocidad motor es la velocidad nominal del robot que se determinará al principio del código. El valor que se introducirá en la rueda izquierda está multiplicado por el factor 0.91 porque se observó que el motor izquierdo gira más rápido que el derecho ante la misma entrada ya que se hicieron ensayos de meter los mismos valores para que el robot fuera en línea recta y se comprobó que sufría una desviación en su trayectoria rectilínea importante.

En el caso de que el valor de PWM, que va desde 0 hasta 255, sature por debajo, es decir que sea negativo, se cambiará ese valor a positivo y se invertirá el sentido de giro de las ruedas.

## 5.4. Conexionado driver

Como ya se había comentado, se utilizará el integrado 293d de Texas Instruments ya que nos permitirá controlar de forma independiente los 2 motores y además permite que gire cada uno en ambos sentidos.

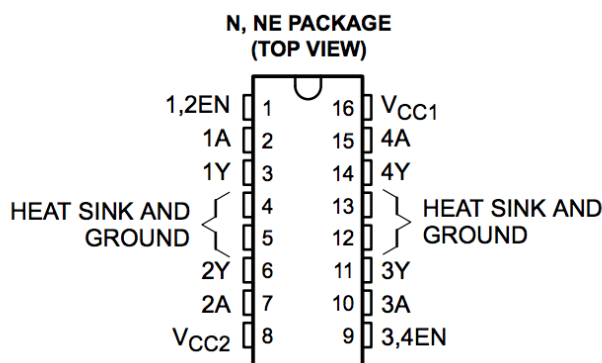


Ilustración 20: Diagrama de pines driver L293D

Para regular la velocidad de los motores se utilizara la técnica de PWM (Pulse Width Modulation). Desde dos pines con salida PWM del Arduino controlaremos el enable de cada motor, pines 1 y 9, introduciendo un valor entre 0 y 255 que vendrá dado por el algoritmo de control ya explicado. Las patillas 1A, 2A, 3A y 4A, pines 2, 7,10 y 15 respectivamente, determinan el sentido de giro de cada motor y nos permitirán cambiar el sentido de las ruedas cuando el control sature hacia valores negativos de PWM.

El resto de patillas del integrado son los pines 4, 5,12 y 13 que van conectados a tierra, los pines 3, 6, 11 y 14 que son las salidas a los motores, la patilla 16 VCC1 que se conecta que es la alimentación lógica (5V) y por último la patilla 8 VCC2 que es la que se conecta a la batería de 9V.

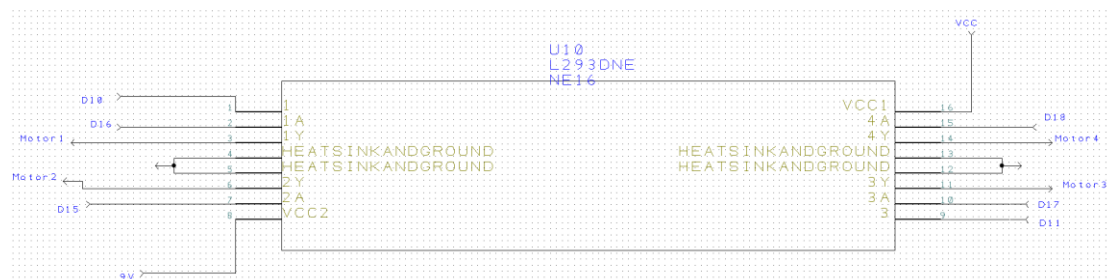


Ilustración 21: Esquema de conexionado del driver 293d

## 5.5. Configuración módulo Bluetooth

Esta parte del desarrollo del proyecto fue completamente novedosa ya que en el grado se había estudiado la tecnología Bluetooth pero desde un punto de visto teórico y sin profundizar demasiado así que aprender a configurar y a usar un módulo bluetooth se presentaba como todo un desafío.

Se investigó por Internet las propiedades del módulo y si venia configurado por defecto o no y al ser un módulo chino existiendo múltiples fabricantes había fuentes que decían que estaba sin configurar y otras que decían que estaban con una configuración u otra. Ante esta incertidumbre se decidió configurarlo con los parámetros que nosotros quisiéramos.

Buscando las distintas opciones que se existían para configurarlo se descartaron aquellas que precisaban un adaptador usb-RS232. Se decidió configurarlo mediante el propio Arduino a través del envío de comando AT. Los comandos AT son una interfaz de comunicación se inventaron para poder configurar y enviar órdenes a módems.

Lo primero que se debe comprobar es que el módulo no este emparejado con ningún dispositivo, a priori es de suponer que no lo estará ya que es un módulo nuevo. Para realizar esta comprobación habrá que fijarse si la luz roja del módulo parpadea o no. Una vez comprobado que no está emparejado con ningún dispositivo se carga el programa en el Arduino y cuando se indique se conectan el terminal RX del módulo con el TX del Arduino y el TX del módulo con el RX del Arduino. Esto se hace así porque para poder programar el Arduino los pines RX y TX del mismo deben estar desconectados

El programa para configurar el módulo JY-MCU utilizado es el siguiente:

```
int cont = 2;

void setup(){

pinMode(13,OUTPUT); // Led para controlar la configuración

Serial.begin(57600); // Velocidad del módulo bluetooth
```

Pablo Ferro Laspidea

```

digitalWrite(13, LOW); // Apagamos el LED 13
    }

void loop(){
while (cont==1){
    digitalWrite(13, HIGH); // Enciende el LED

    delay(1200);

    digitalWrite(13, LOW); // Apaga el LED

    delay(1200);

    }

else{
digitalWrite(13, HIGH);

Serial.println("Configuración modulo JY MCU");

Serial.println("Conectar RX del módulo al TX del Arduino y TX del
modulo al RX del Arduino") ;
delay(10000); // esperamos 10 segundos para que se conecte

digitalWrite(13, LOW); // apagamos Led indicando que el tiempo ha acabado

Serial.print("AT"); // empieza la configuración

delay(1000); // Espera para el envío de comandos

Serial.print("AT+NAMEBlurobot"); // Nombre del dispositivo, Blurobot

delay(1000); // Espera para el envío de comandos

Serial.print("AT+BAUD4"); // Establecemos la velocidad en 9600 baudios

delay(1000); // Espera para el envío de comandos

Serial.print("AT+PIN1234"); // Establecemos el pin de contraseña 1234

delay(1000); // Espera para el envío de comandos

digitalWrite(13, HIGH); // Todo ha funcionado según lo esperado

cont=1; // Para que no vuelva a entrar en la rutina de configuración salvo reseteo

    }

}

```

Pablo Ferro Laspidea

Una vez se tenía el módulo Bluetooth configurado se cargó el siguiente programa para comprobar que realmente funcionaba.

```
void setup() {
  char pas=0 ;
  Serial.begin(9600); // abre el puerto serie y establece la velocidad en 9600 bps
  pinMode (13, OUTPUT) ;
}

void loop(){
  if(Serial.available(>0)
    {char pas=Serial.read(); // se guarda en la variable pas el carácter que se mande
      switch(pas){ //estructura case determina las acciones según carácter recibido
        case'0':
          digitalWrite(13, LOW) ;
          break;
        case'1':
          digitalWrite(13, HIGH) ;
          delay(1000) ;
          digitalWrite(13, LOW) ;
          delay(1000) ;
          break;
        case'2':
          digitalWrite(13, HIGH) ;
          delay(500) ;
          digitalWrite(13, LOW) ;
          delay(500) ;
          break;
        case'3':
          digitalWrite(13, HIGH) ;
          delay(200) ;
          digitalWrite(13, LOW) ;
          delay(200) ;
          break;
        case'4':
          digitalWrite(13, HIGH) ;
          delay(100) ;
```

Pablo Ferro Laspidea

```

digitalWrite(13, LOW) ;
delay(100) ;
break;
}
}
}

```

Los distintos comandos se mandaban con el programa de LabView que se ha desarrollado y se comprobó que efectivamente el Arduino recibía los comandos ya que el encendido y apagado del led cambiaba.

Hay que tener en cuenta que el modulo JY-MCU trabaja con una lógica de 3.3V y el Arduino manda por su terminal TX una lógica de 5V por lo que habrá que colocar un divisor de tensión conectado al terminal RX del módulo Bluetooth.

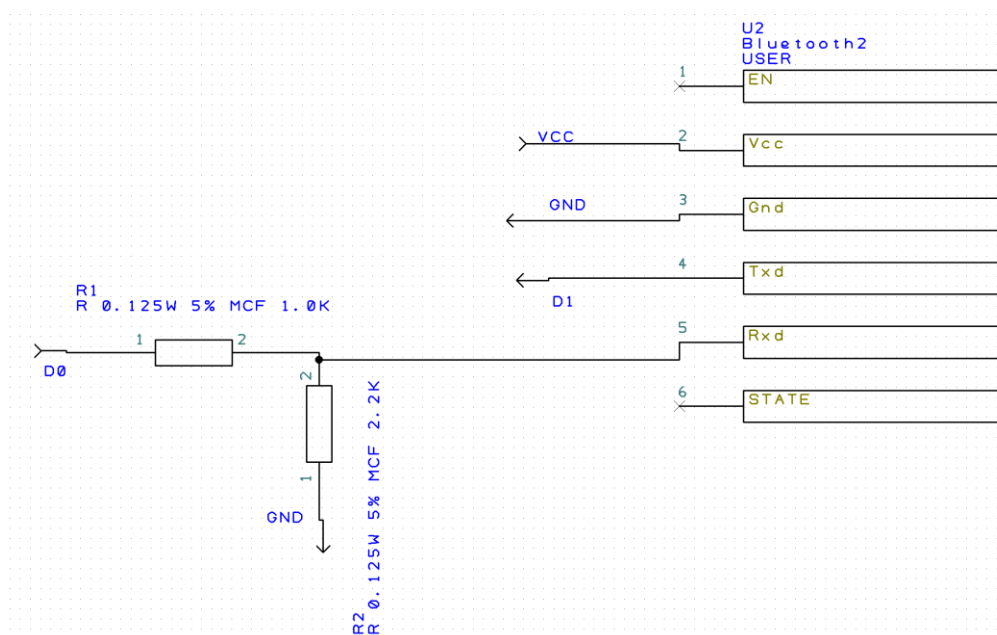


Ilustración 22: Esquema conexionado del módulo Bluetooth

## 5.6. Programa LabView

### 5.6.1. Alternativas

Desde las primeras fases del proyecto uno de los objetivos era que se pudiera controlar o interactuar con el robot a distancia sin la utilización de cables ya que es un dispositivo en movimiento. Una vez elegida la tecnología bluetooth para establecer la comunicación se presentaban diversas opciones para comunicarnos

Pablo Ferro Laspidea

con el robot entre las que destacaba la utilización de algún programa de ordenador específico para mandar órdenes a través de bluetooth, la utilización de alguna aplicación Android ya desarrollada para comunicarnos vía el Smartphone y, por último el desarrollo de un programa con la herramienta LabView.

Se escogió esta última opción porque se considera que es la opción más potente de las 3, implica un reto ya que se desconoce casi por completo el programa, ya que a pesar de haberlo visto en el grado han sido pinceladas, y porque el grado de personalización y de adaptación a lo que se quiere hacer es muy alto.

### 5.6.2. Desarrollo del programa

El objetivo era desarrollar un programa para enviar y recibir datos entre el ordenador y el Arduino. Debido a que no se domina el programa el apartado de enviar caracteres al Arduino se consiguió desarrollar de forma satisfactoria pero la parte de recibir datos se descartó porque generaba problemas de estabilidad, el programa en ocasiones se cerraba de forma inesperada, y además se tuvo ciertos problemas a la hora de generar graficas con los datos recibidos, en nuestro caso probamos con el dato del error.

A continuación se muestran algunas capturas del diagrama de bloques del programa y una breve explicación de cada parte.

#### Configuración puerto serie

Esta primera parte se encarga de configurar el puerto serie introduciendo parámetros como la velocidad de transferencia de datos, el puerto por el que se conecta el bluetooth, la paridad, los bits de parada...



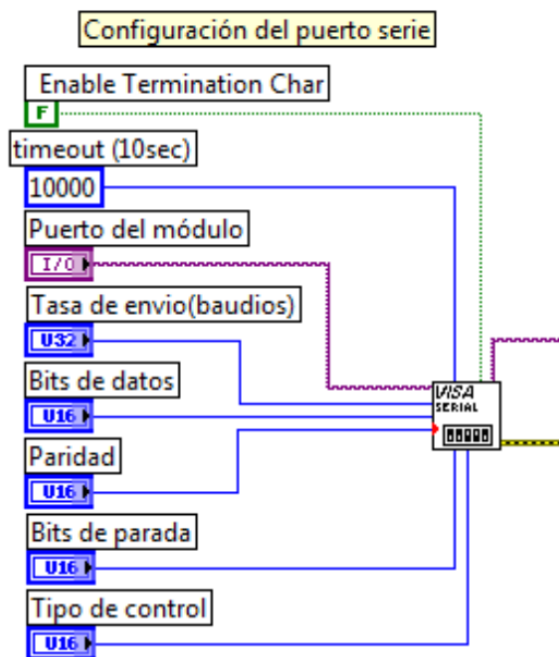


Ilustración 23: Estructura configuración puerto serie

El resto de bloques están introducidos en una estructura while que se ejecuta cada segundo gracias a la utilización del bloque wait until next ms multiple. Al principio no estaba puesto este cronometro y el bucle se ejecutaba de forma continua en el tiempo dando problemas de estabilidad ya que el programa se colgaba.

Escritura de datos

La estructura que nos permite escribir los caracteres en el puerto serie esta englobada en una estructura case, ya que se activa y desactiva a través de un booleano, y está formada por un VISA Write a la que está conectada una segunda estructura case en la que dictaminamos el carácter que escribimos según como estén pulsados los pulsadores de pasillo.

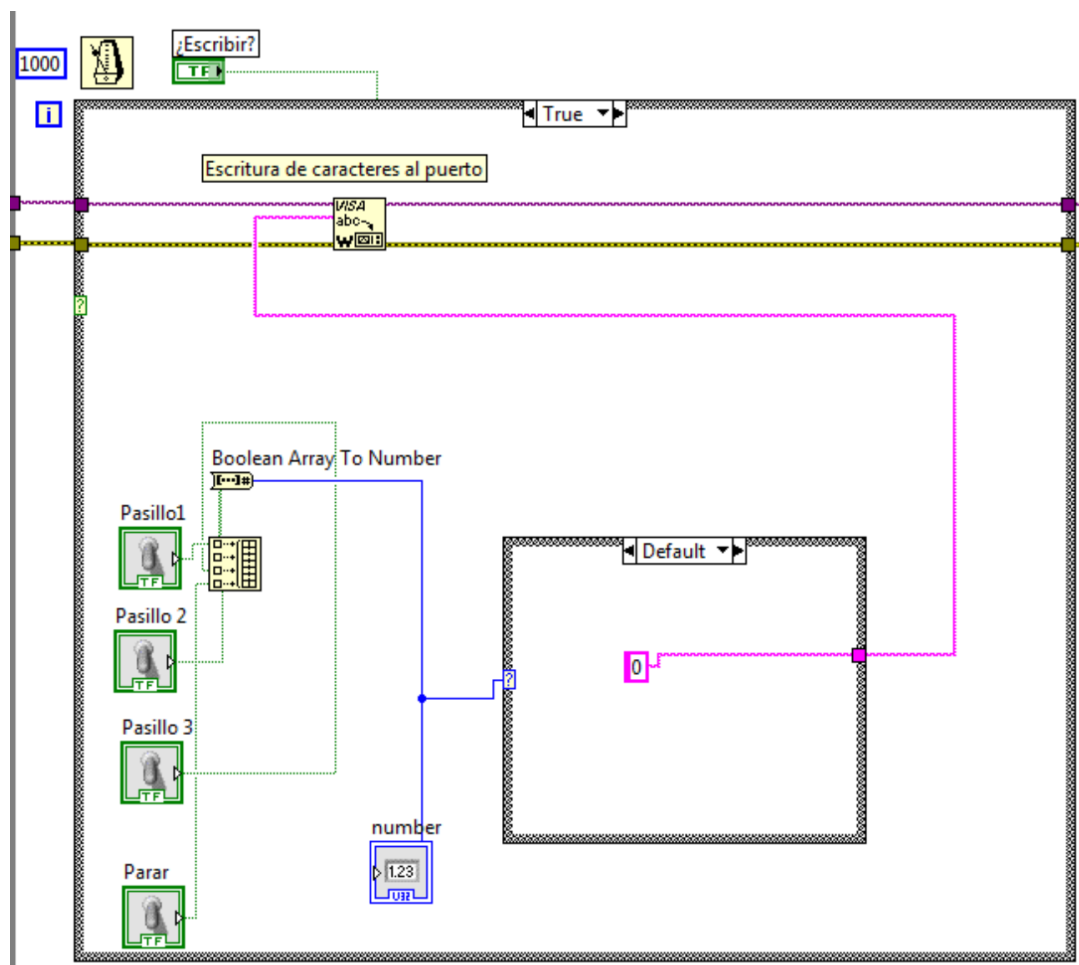


Ilustración 24: Estructura escritura de caracteres al puerto serie

Lectura de datos

La estructura que nos permite leer los datos que nos envía el Arduino está englobada en una estructura case igual a la del caso anterior para activar o desactivar la función mediante un booleano y está formada por un bloque VISA Read. Los datos que se leen están en forma de string y hay múltiples conversiones a otros tipos de datos cuyo propósito era intentar leerlos de la forma más adecuada. También hay un bloque de XY graph cuyo propósito era hacer una gráfica con los errores que nos mandaba el Arduino pero no termino de funcionar del todo. Recordar que esta parte del programa al final no se usará debido a los problemas antes mencionados.

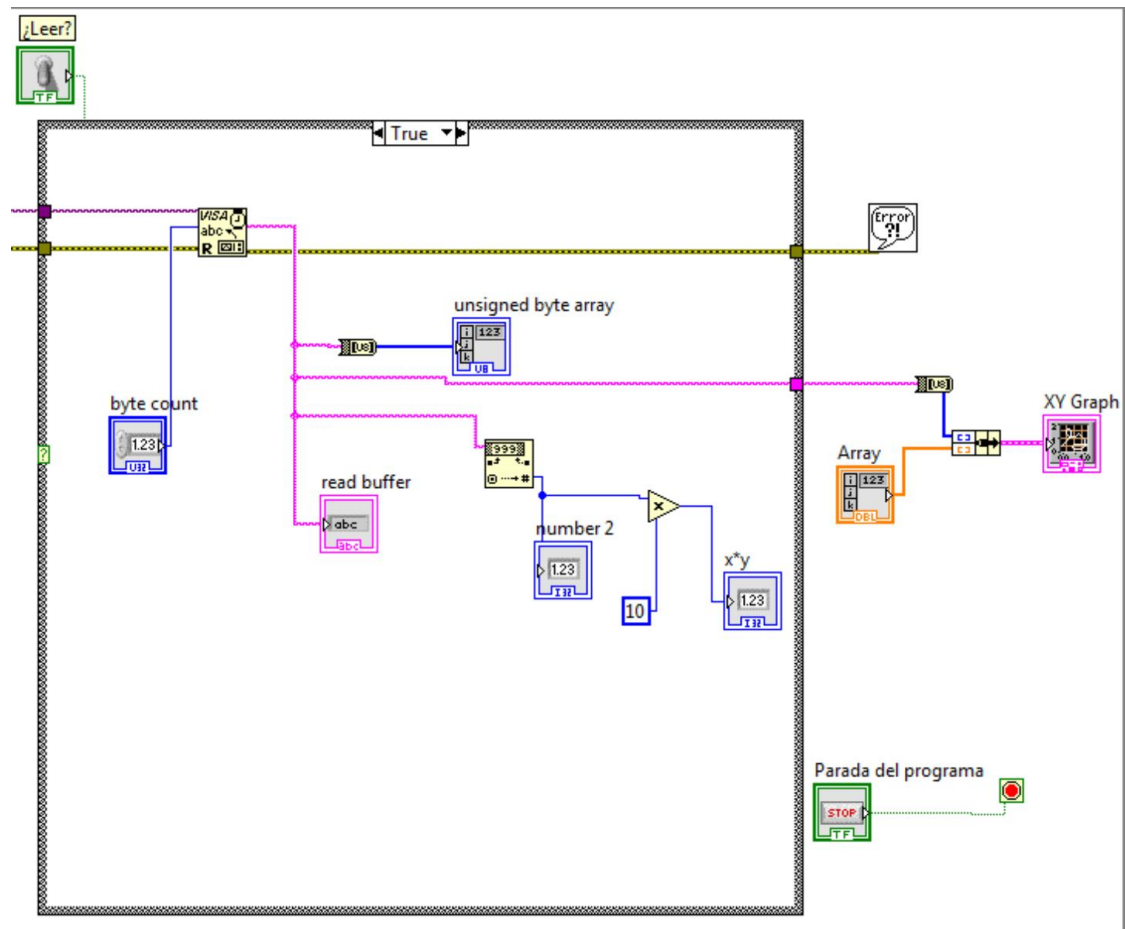


Ilustración 25: Estructura lectura de datos del puerto serie

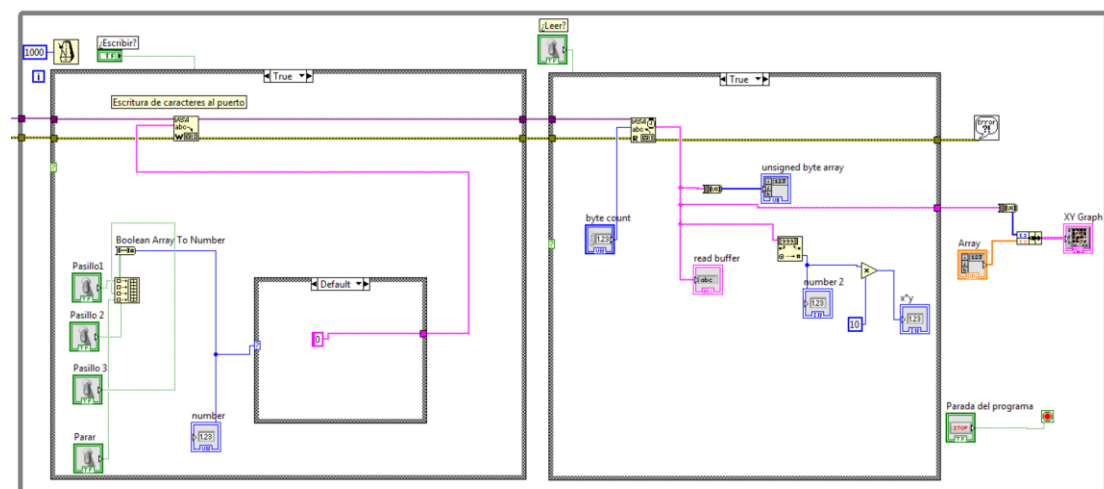


Ilustración 26: Estructura while de escritura y lectura de datos

A parte del diagrama de bloques en LabView tenemos también el panel frontal que es el que se supone que va a ser manipulado por el usuario final del programa. Desde este panel frontal se podrá configurar el puerto serie introduciendo los parámetros de: puerto, tasa de envío, bits de datos, paridad, bits de parada y tipo de

Pablo Ferro Laspidea

control. Además se podrá elegir si queremos escribir o no, la opción de leer deberá estar desactivada, elegir por que pasillo queremos que vaya el robot o si queremos que se quede parado en la zona de carga y descarga y por último parar el programa dándole al botón de STOP.

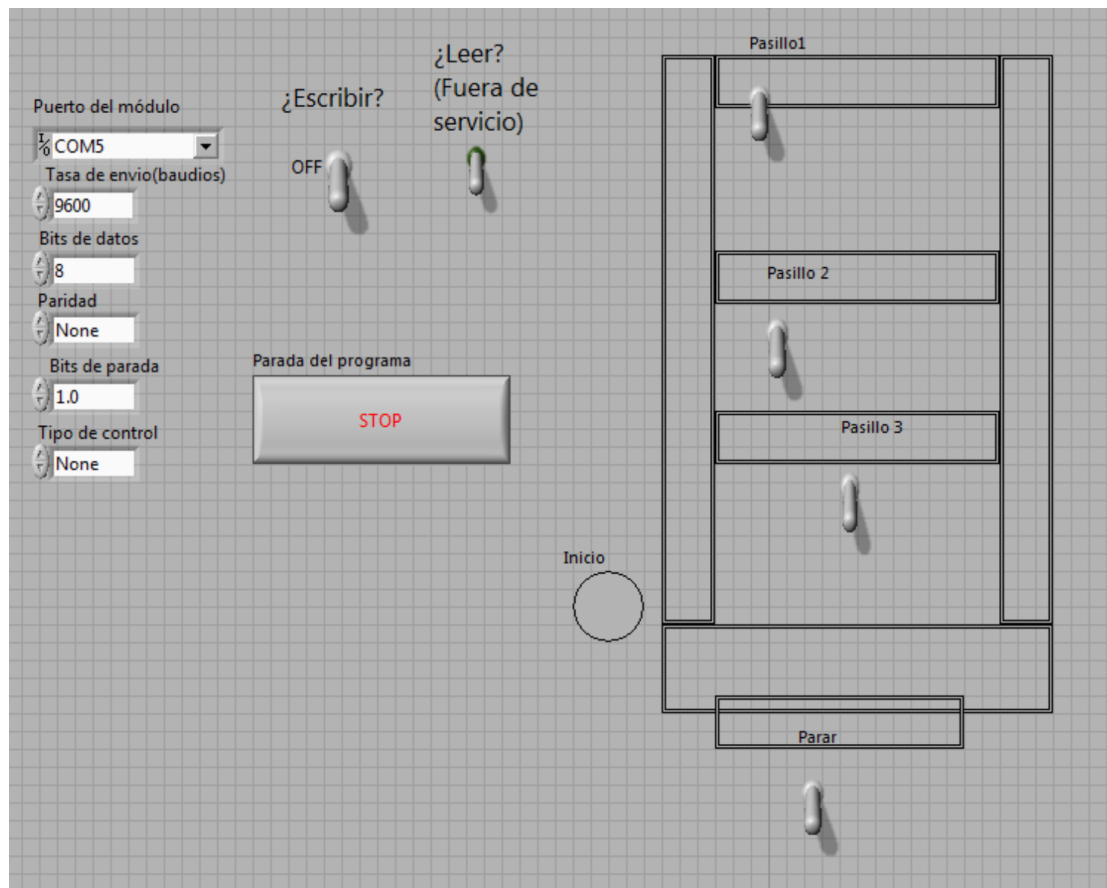


Ilustración 27: Panel frontal escritura caracteres

## 5.7. Arduino

Como ya se ha comentado anteriormente, se eligió el Arduino Nano V.3 como microcontrolador del robot.

### 5.7.1. Pines

A continuación se muestra a que está conectado cada pin del Arduino.

Patilla	Net	Destino
0(RX)	D1	Tx Bluetooth
1(TX)	D0	Rx Bluetooth
2	D2	Sensor5
3	D3	Sensor4
4	D4	Sensor3
5	D5	Sensor2
6	D6	Zumbador
7	D7	Sensor1
8	D8	Trigger HC-SR04
9	D9	Echo HC-SR04
10	D10	Enable motor izda.
11	D11	Enable motor dcha.
12	-	-
13	-	-
A0	-	-
A1	D15	Sentido giro motor izda.
A2	D16	Sentido giro motor izda.
A3	D17	Sentido giro motor dcha.
A4	D18	Sentido giro motor dcha.
A5	-	-
A6	-	-
A7	-	-

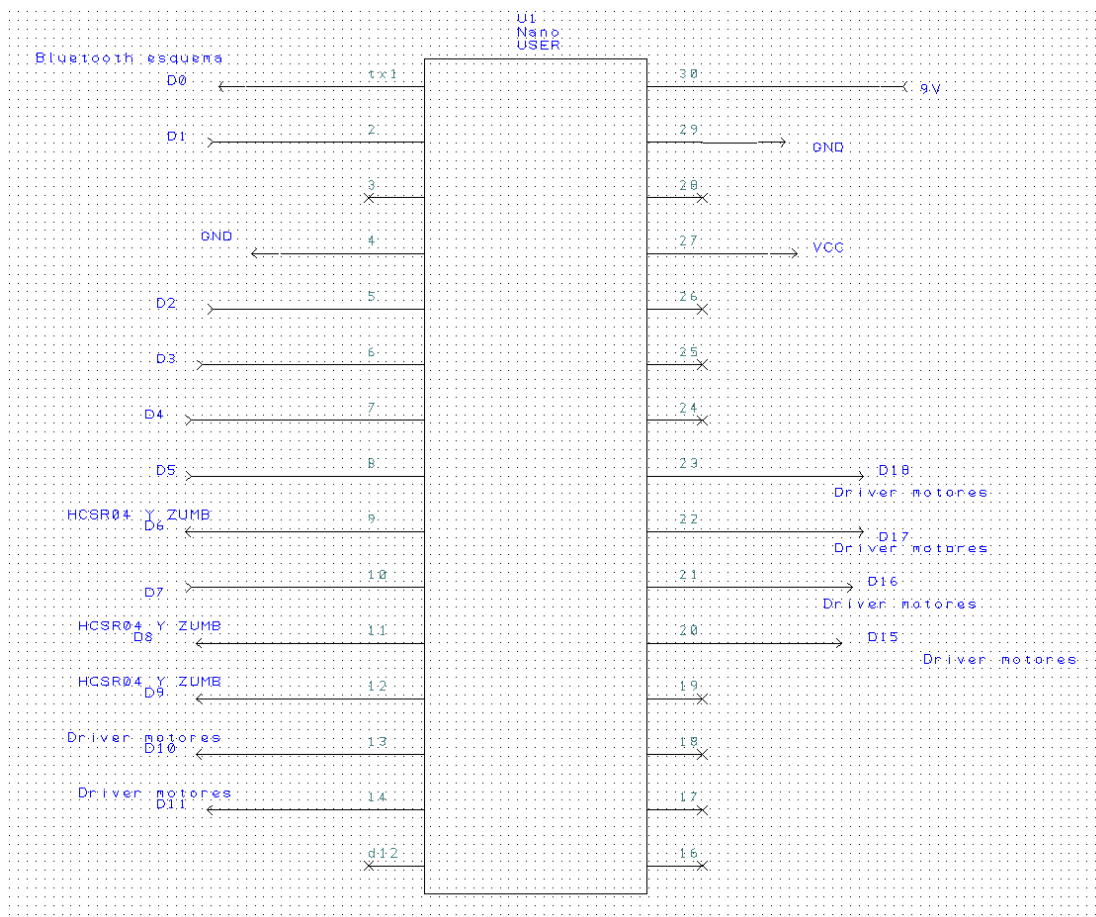


Ilustración 28: Esquema conexionado Arduino Nano

### 5.7.2. Programación

```
// INICIO DEL PROGRAMA

//Declaración de variables
int sensor[]={0, 0, 0, 0, 0};
int frecuencia=120 ; // valor para meter en el pwm del zumbador
float error=0 , error_old=0 , derivada=0 , integral=0 , integral_old=0, PID=0 ;// Variables y
parametros para el calculo de PID
float duracion=0 ;
float distancia=0 ; // Variable para guardar la distancia medida
int pasnum=0; //variable para contar el pasillo
int pasil=0; //Variable de pasillo actual en numero y no en caracter como nos llega del pc
int retorno=0; //Variable para saber que estamos en la rama de vuelta y que no haga caso
interesciones
float Kp=40,Ki=0,Kd=7; // Variables flotantes para el controlador PID
int velocidad_motor=80 ; // El valor es orientativo habra que probar distintos valores
int motizda=0 ;
int motdcha=0 ;

//Definimos las funciones que se utilizaran en el programa

void leer_sensores(void);
void calculo_pid(void);
```

```

void control_motores(void);
void destino(void);
void obstaculo(void);
void girar(void);
void recto(void);
void setup() // Declaramos los pines e inicializamos la comunicación serie
{
  for(int i=2;i<6;i++){ //Sensores conectados a los pines 2,3,4,5
    pinMode(i,INPUT);
  }
  pinMode(6,OUTPUT); // Para el zumbador
  pinMode(7,INPUT); // para el ultimo sensor
  // pinMode(sensor_barras,INPUT);
  pinMode(8,OUTPUT); //Trigger
  pinMode(9,INPUT); //Echo
  pinMode(10,OUTPUT); //Control para los motores PWM
  pinMode(11,OUTPUT); //Control para los motores PWM
  pinMode(15,OUTPUT); //Sentido para los motores IZDA
  pinMode(16,OUTPUT); //Sentido para los motores IZDA
  pinMode(17,OUTPUT); //Sentido motores DCHA
  pinMode(18,OUTPUT); // Sentido motores DCHA
  Serial.begin(9600);
}
//Bucle principal donde se van a ir ejecutando todas las subrutinas
void loop()
{
  leer_sensores();
  calculo_pid();
  control_motores();
  destino();
  obstaculo();
}

void leer_sensores() //Nos saltamos el 6, porque el 6 va al zumbador
{
  sensor[0]=digitalRead(7);
  sensor[1]=digitalRead(5);
  sensor[2]=digitalRead(4);
  sensor[3]=digitalRead(3);
  sensor[4]=digitalRead(2);

  if((sensor[0]==0)&&(sensor[1]==0)&&(sensor[2]==0)&&(sensor[3]==0)&&(sensor[4]==1))
    error=4;
  else
  if((sensor[0]==0)&&(sensor[1]==0)&&(sensor[2]==1)&&(sensor[3]==1)&&(sensor[4]==1))
    error=3;
  else
  if((sensor[0]==0)&&(sensor[1]==0)&&(sensor[2]==0)&&(sensor[3]==1)&&(sensor[4]==1))
    error=3;
  else
  if((sensor[0]==0)&&(sensor[1]==0)&&(sensor[2]==0)&&(sensor[3]==1)&&(sensor[4]==0))
    error=2;
}

```

```

    else
if((sensor[0]==0)&&(sensor[1]==0)&&(sensor[2]==1)&&(sensor[3]==1)&&(sensor[4]==0))
    error=1;
    else
if((sensor[0]==0)&&(sensor[1]==0)&&(sensor[2]==1)&&(sensor[3]==0)&&(sensor[4]==0))
    error=0;
    else
if((sensor[0]==0)&&(sensor[1]==1)&&(sensor[2]==1)&&(sensor[3]==0)&&(sensor[4]==0))
    error=-1;
    else
if((sensor[0]==0)&&(sensor[1]==1)&&(sensor[2]==0)&&(sensor[3]==0)&&(sensor[4]==0))
    error=-2;
    else
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor[2]==0)&&(sensor[3]==0)&&(sensor[4]==0))
    error=-3;
    else
if((sensor[0]==1)&&(sensor[1]==1)&&(sensor[2]==1)&&(sensor[3]==0)&&(sensor[4]==0))
    error=-3;
    else
if((sensor[0]==1)&&(sensor[1]==0)&&(sensor[2]==0)&&(sensor[3]==0)&&(sensor[4]==0))
    error=-4;
    else
if((sensor[0]==0)&&(sensor[1]==0)&&(sensor[2]==0)&&(sensor[3]==0)&&(sensor[4]==0))
    {if(error==-4){ // dependiendo de su error anterior para saber por que lado nos hemos
salido
        error=-5; // de la linea
    }
    else {
        error=5;
    }
    }
}
}
// Funcion para calcular la accion de control
void calculo_pid()
{
    derivada=error - error_old ; //Calculamos el cambio del error
    integral= error + integral_old ; // Calculamos el error que se acumula
    PID= (Kp*error)+(Kd*derivada)+(Ki*integral) ; // Calculamos el PID
    integral_old=integral ; //Ponemos los valores de error e integral como los actuales listos
para la siguiente iteracion
    error_old=error ;
}

```

/\*Ahora a la hora de introducir el PID en el calculo de velocidad de los motores hay que tener en cuenta una cosa:

Si PID resulta negativo significa que estamos dejando la linea a la izquierda del robot por lo que la rueda derecha debera

girar mas rapido que la izquierda para que gire y viceversa.En otras palabras, cuando el PID sean negativo la velocidad del motor derecho se incrementera

y la velocidad del motor izquierdo se decrementara y viceversa. \*/



```

void control_motores()
{
  //Configuramos sentido de giro de los motores
  digitalWrite(16,LOW); //Motor izquierdo
  digitalWrite(15,HIGH); //Motor izquierdo
  digitalWrite(18,LOW); //Motor derecho
  digitalWrite(17,HIGH); //Motor derecho

  int velocidad_motor_derecha= velocidad_motor - PID ;
  int velocidad_motor_izquierda= 0.91*(velocidad_motor + PID) ;
  if(velocidad_motor_derecha < 0){ // si nos pasamos por debajo cambiamos le ponemos
positivo y cambiamos signo
    velocidad_motor_derecha= 0 - velocidad_motor_derecha ;
    digitalWrite(17,LOW);
    digitalWrite(18,HIGH);
  }
  if(velocidad_motor_izquierda < 0){
    velocidad_motor_izquierda= 0 - velocidad_motor_izquierda ;
    digitalWrite(15,LOW);
    digitalWrite(16,HIGH);
  }
  if(velocidad_motor_izquierda > 255){ // si nos desbordamos por arriba lo ponemos al
máximo
    velocidad_motor_izquierda=255 ;
  }
  if(velocidad_motor_derecha > 255){ // si nos desbordamos por arriba lo ponemos al
máximo
    velocidad_motor_derecha=255 ;
  }

  // Regulamos velocidad de giro con el PWM al enable de cada motor
  motizda = (int) velocidad_motor_izquierda;
  motdcha = (int) velocidad_motor_derecha;
  analogWrite(10, motizda); // El pin 10 es el que va al motor izquierdo y el 11 al derecho
  analogWrite(11, motdcha);
}

```

```

void destino()
{
  if(Serial.available(>0)
  {
    delay(5);
    /*pas es la variable donde se va a guardar el pasillo donde tengo que ir
y que me llegara por el puerto serie a traves de LabView
*/
    char pas=Serial.read(); //Sentencia case para ir a distintos escenarios
    switch(pas){
      case'1':
        pasil=1;
        break;

```

Pablo Ferro Laspidea

```

    case'2':
    pasil=2;
    break;

    case'3':
    pasil=3;
    break;
  }
}
delay(5);
/*

```

Si detecta interseccion pasillo, y no esta volviendo, que aumente el contador de pasillo y que lo compare con el pasillo donde tiene que ir

Pas es a donde queremos ir y pasnum es en el que estamos.

Si detectamos intersección pasillo pero estamos volviendo decrementamos hasta detectar que es el ultimo pasillo

en el camino de vuelta y ponemos que ya no estamos en retorno

```
*/
```

```

if((sensor[0]==1)&&(sensor[1]==0)&&(sensor[2]==1)&&(sensor[3]==1)&&(sensor[4]==1)) //
Intersección para giro pasillo

```

```

{
  if(retorno==0){ // si no estamos volviendo
    pasnum++; // aumenta pasnum que es el contador de pasillos
    if(pasnum==pasil){ //comprueba si es el pasillo que queremos
      girar(); // vamos a la funcion de girar
    }
  }
  else{//rutina para que siga recto
    recto();
  }
}
else{ //se detecta intersección de pasillo pero estamos volviendo
  recto();
  pasnum=pasnum-1;
  if(pasnum==1){ // hemos salido del circuito de vuelta
    retorno=0;
  }
}
}
}

```

```

if((sensor[0]==1)&&(sensor[1]==1)&&(sensor[2]==1)&&(sensor[3]==1)&&(sensor[4]==1))
//Interseccion total, a izda y dcha

```

```

{ retorno=1 ;
  girar();
}

```

```

}

```

/\*Definimos la función obstaculo en la que determinaremos que si se detecta un obstaculo a menos de 20cm que se pare el robot

hasta que la distancia con el obstaculo hay aumentado a mas de 20cm

```

    */
void obstaculo()
{
  digitalWrite(8,LOW);
  delayMicroseconds(2);
  digitalWrite(8,HIGH);
  delayMicroseconds(10);
  digitalWrite(8,LOW);
  duracion= pulseIn(9,HIGH); // Medimos el tiempo del flanco del eco en microsegundos
  distancia= duracion*0.01715 ;//multiplicamos por la velocidad del sonido en cm/us
  dividia por 2
  while(distancia <= 20) // Si detecta algo a menos de 20cm
  {
    analogWrite(6, frecuencia); // pita el zumbador
    digitalWrite(16,LOW); // detenemos los motores
    digitalWrite(15,LOW);
    digitalWrite(18,LOW);
    digitalWrite(17,LOW);
  }
  analogWrite(6, 0); // apaga el zumbador
}
void girar(){ // rutina para que gire
  digitalWrite(16,LOW);
  digitalWrite(15,HIGH);
  digitalWrite(18,HIGH); // Invertimos sentido giro motor derecha
  digitalWrite(17,LOW);
  analogWrite(10, velocidad_motor); // El pin 10 es el que va al motor izquierdo y el 11 al
derecho
  analogWrite(11, velocidad_motor);
  delay(500); //medio segundo girando
}
void recto(){
  analogWrite(10, velocidad_motor); // El pin 10 es el que va al motor izquierdo y el 11 al
derecho
  analogWrite(11, velocidad_motor);
  delay(500); //medio segundo avanzando recto
}
}

```

## 5.8. Diseño del PCB

Una de las motivaciones más importantes al plantear como iba a ser el proyecto fue el poder realizar algo tangible, algo que funcionara. Casi desde el principio se descartó la idea de utilizar una protoboard, una placa de montaje de circuitos. Se descartó porque se quería realizar un diseño más completo, no solo el diseño del circuito del robot sino también el diseño de un PCB específico para este proyecto

aplicando los conceptos aprendidos en la asignatura de “Fabricación y ensayo de equipos electrónicos”.

### 5.7.1. ¿Qué es un PCB?

Un PCB es una tarjeta de circuito impreso donde sobre una superficie no conductora se disponen de pistas o caminos de material conductor. Las siglas comúnmente conocidas de PCB vienen del inglés (Printed Circuit Board). Los circuitos impresos se utilizan para conectar componentes electrónicos a los que además les sirve como soporte y sostén mecánico. Las pistas o caminos de material conductor suelen ser casi siempre de cobre mientras que la base no conductora se suele fabricar de resinas de vidrio reforzadas, cerámicas, de teflón o con polímeros como la baquelita.

Su aplicación más común es en dispositivos electrónicos o que contengan algo de electrónica en su interior ya que su producción automatizada en masa reduce considerablemente los costes alcanzando unos grados de calidad y robustez óptimos. También se pueden producir de forma manual para prototipos y pequeños lotes.

Un circuito impreso puede contar con una o varias capas laminadas entre sí, en la mayoría de los casos un PCB está compuesto por entre una a dieciséis capas conductoras, separadas y soportadas por capas de material aislante. Este material aislante debe tener ciertas propiedades como que sea rígido, fácil de taladrar, que no prenda, que disipe bien el calor, que tenga un punto de ruptura dieléctrico alto...

Si el PCB cuenta con más de una capa, caso muy común, éstas se pueden conectar a través de orificios en las superficies llamados vías. Estas vías pueden ser conductoras si están recubiertas por un material conductor o no conductoras siendo necesario un remache para conectar con las pistas. Si el PCB cuenta con muchas capas podemos encontrarnos con vías ciegas que solo son visibles por un lado de la tarjeta e incluso con vías enterradas que se encuentran entre capas interiores y no son visibles desde el exterior.

### 5.7.2. ¿Monocapa o multicapa?

En una fase inicial se pensó en realizar un diseño de doble capa para que la complejidad del circuito no afectara al tamaño de la placa e incluso se empezó a diseñar con el programa DesignSpark pero el tutor del proyecto me aconsejó

Pablo Ferro Laspidea

realizarlo a una capa por la mayor simplicidad a la hora de fabricarlo lo que repercutía en un menor plazo de tiempo en su producción. Este consejo me hizo ver que no era necesario un diseño a doble capa ya que planteando muy bien la ubicación de los componentes y de los pines se podría conseguir un tamaño aceptable.

### 5.7.3. Componentes fijos

Así pues se comenzó el diseño de la placa ubicando primero los componentes fijos, es decir, aquellos componentes que debían de estar en un sitio específico de la placa ya que su ubicación era un factor primordial para la funcionalidad del robot.

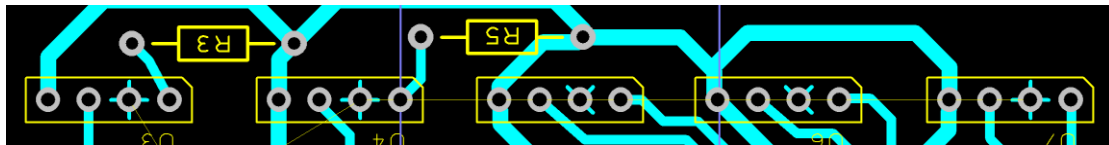
Estos componentes eran:

- Las 2 ruedas motoras en la parte trasera del robot.
- La rueda loca en la parte delantera para que sirva de soporte y apoyo y a una distancia un poco menor que la distancia entre ruedas motoras
- Los 5 sensores TCRT1000.

La ubicación de estos sensores era lo más crítico de toda la distribución ya que una mala disposición podría arruinar el diseño. Se dispusieron de tal forma que el sensor central estuviera justo en el centro de la pcb y los otros cuatro a los lados, dos a la izquierda y dos a la derecha.

La distancia del sensor central a los contiguos fue tal que el sensor central pudiera leer la línea y que el resto leyeran blanco. Esto es fundamental ya que como veremos más adelante la disposición de línea en el centro tendrá un error cero.

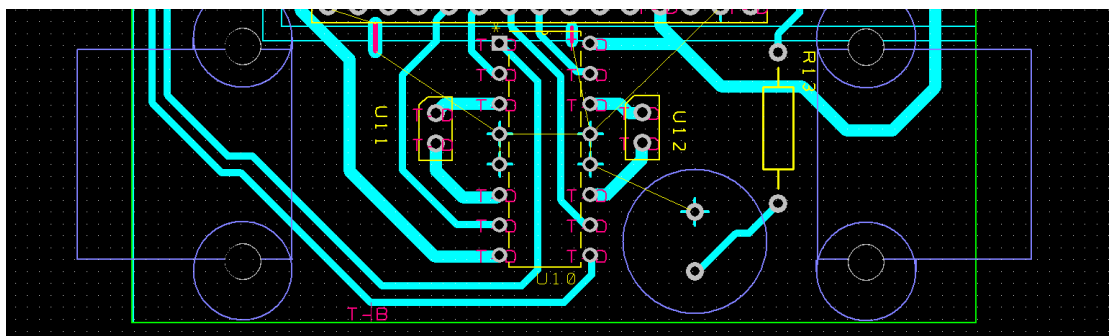
En la siguiente captura del programa DesignSpark se puede ver como en el caso de línea en el centro del robot, solo el sensor central ve la línea, en color morado. Hay que tener en cuenta que lo que vemos en la imagen son los zócalos de los sensores, el fotodiodo y el fototransistor se situarían en la mitad del zócalo aproximadamente. De aquí en adelante las capturas se mostrarán sin el plano de tierra GND para mejorar la visualización de las pistas.



**Ilustración 29:** Disposición de los sensores (amarillo) respecto a la línea (morado) con el plano de GND quitado.

#### 5.7.4. Componentes libres

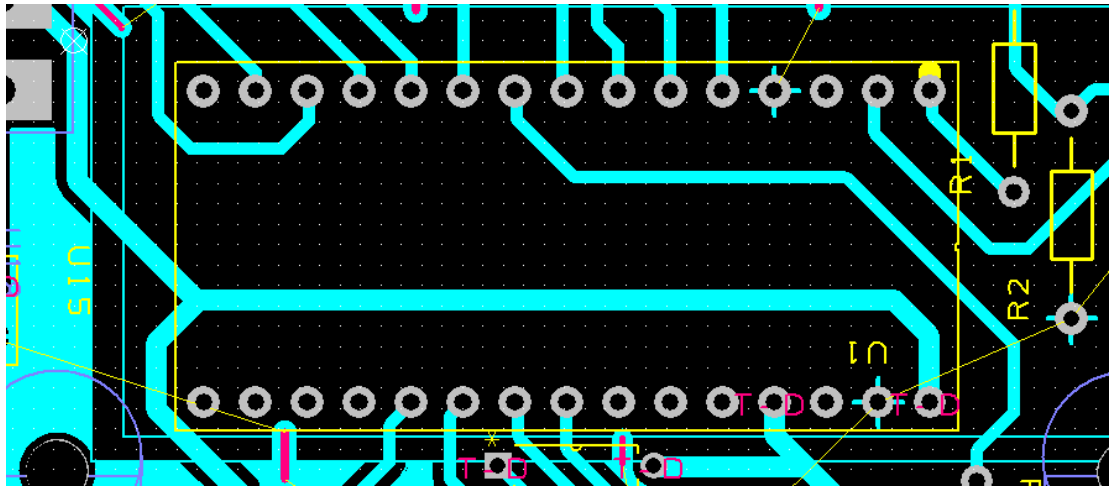
Una vez ubicados los componentes fijos se trataba de ir colocando los distintos componentes. Lo primero que se colocó fue el driver de los motores en el medio de las 2 ruedas. Se colocó en ese lugar para que las corrientes altas quedaran en una zona concreta sin afectar al resto del circuito y por proximidad a los motores, de esta forma se podría delimitar la zona de potencia con su plano de tierra aislado dedicado y “separado” del resto del circuito.



**Ilustración 30:** Disposición del driver (U10) respecto a los motores, morado en los laterales

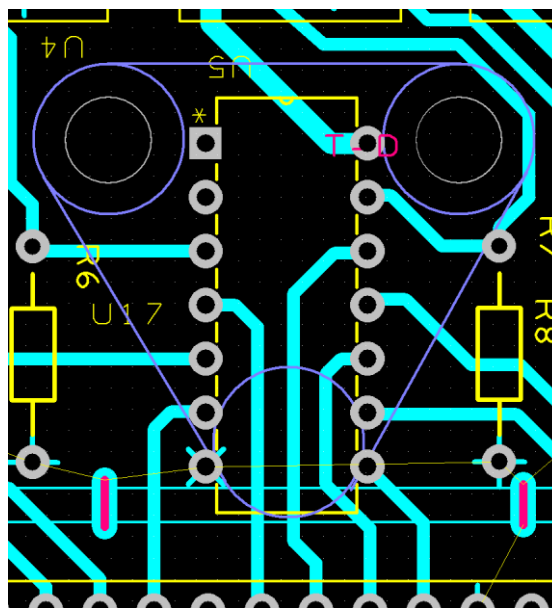
Como se observa en la captura el driver U10 está situado en el medio de las 2 ruedas, y sus motores, en morado. Además los conectores de los motores, U11 y U12, están situados de forma contigua a cada lado del driver.

Lo siguiente que se colocó fue el zócalo para el Arduino Nano. El objetivo era colocarlo en una zona que quedará accesible al resto de componentes, es decir, en torno al centro de la placa. La situación y disposición final fue resultado de ir moviendo y girando el mismo al ir añadiendo conexiones hasta encontrar la forma final.



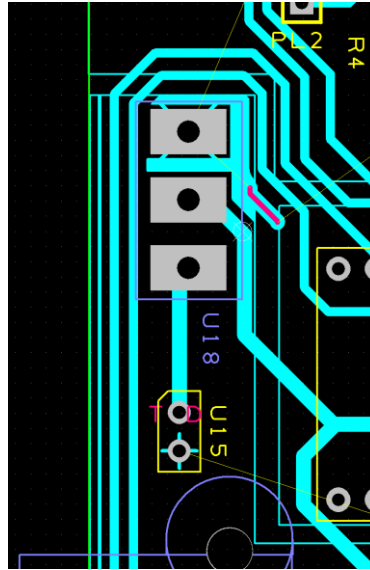
**Ilustración 31: Zócalo Arduino Nano (U1)**

El integrado schmitt trigger debía estar cerca de los sensores y cerca del Arduino así que se colocó en medio de los sensores y el Arduino. Su ubicación final planteó ciertas dudas ya que coincidía con la bola loca, aunque estén cada uno en una capa diferente los pines del integrado quedan bastante cerca de los tornillos de la bola loca. Se probó a girarlo y colocarlo de forma horizontal pero las conexiones se hacían más complicadas al final se colocó de la siguiente forma.



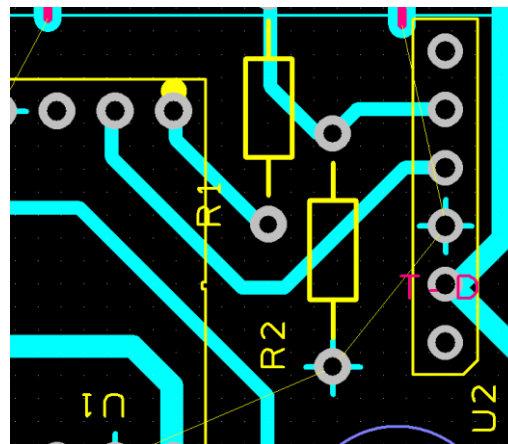
**Ilustración 32: Integrado 74HC14 (U5) con la bola loca superpuesta (U17)**

El interruptor de potencia debía estar cerca de los motores y el driver, para que tuvieran la misma tierra y se decidió colocarlo en el lateral izquierdo del robot. A su vez el conector de la batería se colocó de forma próxima al interruptor.



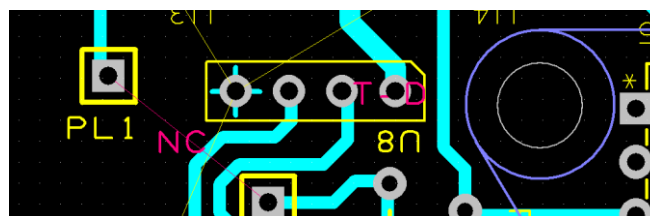
**Ilustración 33: Interruptor (U18) y conector de la batería (U15).**

El módulo Bluetooth, con sus correspondientes resistencias, no precisaba de una disposición específica por lo que se decidió colocar en un lateral de la placa próximo a los pines TX y RX del Arduino.



**Ilustración 34: Módulo Bluetooth (U2) con sus resistencias R1 y R2**

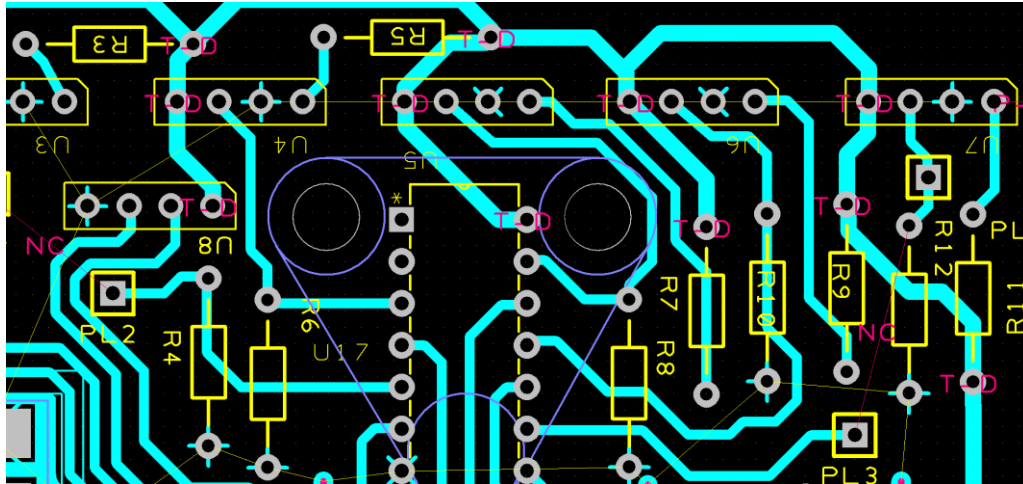
El sensor de ultrasonidos HC-SR04 se colocó en la parte delantera del robot ligeramente hacia un lateral. La situación ideal habría sido colocarlo en la parte central pero para ello había que hacer muchos cambios y ampliar el tamaño de la placa. Se priorizó el tamaño reducido de la placa y se en el lateral.



**Ilustración 35: Zócalo para sensor HC-SR04 (U8)**

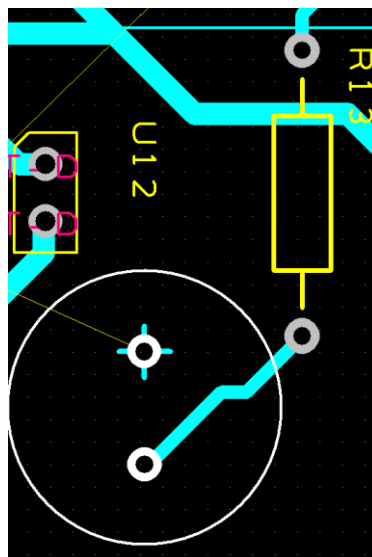


Las resistencias del circuito de los sensores se colocaron lo más próximas posibles a los sensores para evitar posibles pérdidas de señal.



**Ilustración 36: Resistencias del circuito de sensores**

El zumbador y su resistencia se colocó en un espacio libre entre el Arduino y los motores.



**Ilustración 37: Zumbador y resistencia.**

### 5.7.5. Planos

En cuanto a los planos lo ideal habría sido poner 2 planos, uno de tierra GND y otro de alimentación VCC, pero por razones evidentes esto no era posible ya que nuestro PCB es de una capa, por ello se optó por utilizar un plano de tierra GND. Para que no hubiera interferencias y acoplamientos entre las señales analógicas, digitales y de potencia se intentó agrupar los elementos que funcionaban con señales parecidas para diferenciar el plano de tierra en 3 partes: parte analógica, parte digital y parte de potencia.

Así pues en el plano analógico se colocaron los 5 sensores infrarrojos, las resistencias del circuito de los sensores, el integrado 74hc14 y el sensor ultrasónico. A su vez en el plano digital se colocaron el Arduino y el módulo Bluetooth con sus resistencias correspondientes. Y por último en el plano de potencia se colocaron el interruptor, el conector de la batería, el zumbador y su resistencia, los conectores de los motores y el driver de los motores.

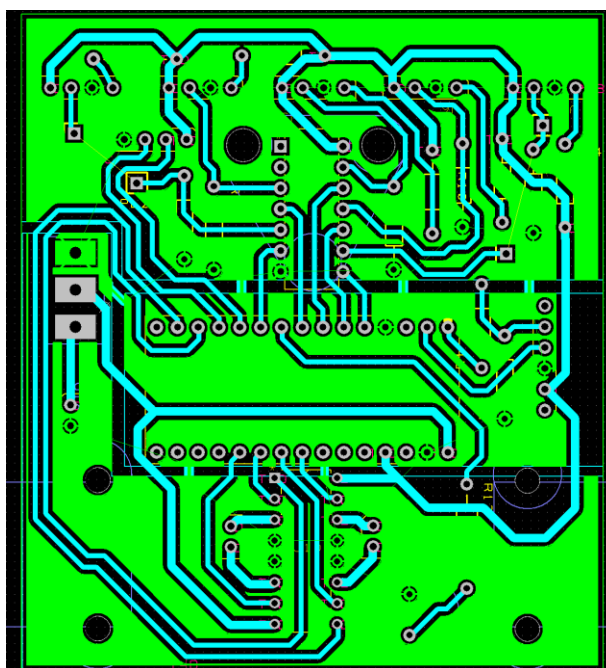
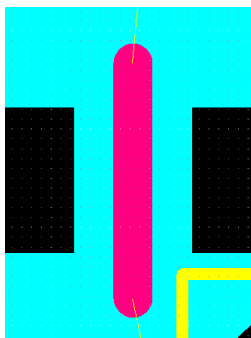


Ilustración 38: Los 3 planos de GND (verde) unidos por pistas GND

Los 3 planos se conectaron entre sí mediante pistas gruesas de net GND, además de para unir los 3 planos entre sí se tuvieron que poner más pistas GND de conexión debido a que en un mismo plano existían islas desconectadas del resto de plano.



**Ilustración 39: Pista de conexión entre planos**

### 5.7.6. Dificultades y retos

A lo largo del proceso de diseño del PCB se presentaron varios retos. Siendo una de las premisas que el tamaño fuera reducido, un hándicap muy importante era el hecho de realizarlo a una capa. Otro hándicap fue que se estableció desde el principio la premisa de que ninguna pista pasara entre los pines de algún componente para evitar posibles cortocircuitos o pistas demasiado estrechas. Si se hubieran pasado pistas entre pines la complejidad del diseño habría sido bastante menor.

Al principio del proceso de conectar los pines de cada net no había problema ya que se iban conectando los pines intentando que las pistas estuvieran colocadas de forma ordenada y con las distancias adecuadas. Pero conforme se iban haciendo pistas cada vez había menos espacios y había que realizar recorridos más largos para conectar los pines llegando al punto de necesitar utilizar vías para seguir por la otra capa. Pero esto obviamente no era posible ya que se estaba trabajando a una capa. Llegados a este punto se reorganizaron pines y se movieron los componentes que se podían mover para poder realizar las conexiones.

La parte más difícil del circuito fueron las resistencias del circuito de sensores que provocaron más de un quebradero de cabeza llegando al punto de necesitar poner 4 conectores para poder hacer 2 puentes para atravesar pistas ya que era imposible conectarlo de otra forma. Las pistas a las que hubo que hacer un puente con un cable fueron las que van del emisor del sensor a la entrada del schmitt trigger correspondiente de los sensores de los extremos.

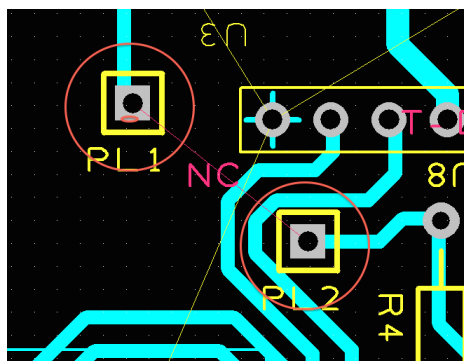


Ilustración 40: Conectores (PL1 y PL2) para sortear las pistas de en medio

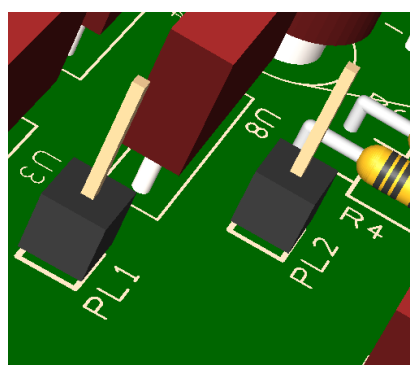


Ilustración 41: Detalle 3d de los conectores PL1 y PL2

Las combinaciones de posiciones, orientaciones y cambio de pines, con el consiguiente cambio en el código, que se produjeron hasta llegar a la configuración final fueron incontables. Las dimensiones finales de la placa son 71mm de ancho por 80mm de largo.

### 5.7.7. Listado y descripción de las nets

Nombre de la Net	Descripción
9 V Bat	Conecta conector positivo batería con interruptor
9V	9V que alimentan Arduino y driver
D0	Del TX del Arduino al divisor de tensión al que está conectado también el RX del Bluetooth
D1	Del TX del Bluetooth a RX del Arduino
D2	La salida del schmitt trigger correspondiente al sensor 5 y que va conectada al pin digital 2
D3	La salida del schmitt trigger correspondiente al sensor 4 y que va conectada al pin digital 3

D4	La salida del schmitt trigger correspondiente al sensor 3 y que va conectada al pin digital 4
D5	La salida del schmitt trigger correspondiente al sensor 2 y que va conectada al pin digital 5
D6	Salida PWM del pin digital 6 al zumbador
D7	La salida del schmitt trigger correspondiente al sensor 1 y que va conectada al pin digital 7
D8	Pin digital 8 envía pulsos al trigger del HC-SR04
D9	Pin digital 9 recibe el eco de la señal del HC-SR04
D10	Señal PWM al enable del motor izquierdo
D11	Señal PWM al enable del motor derecho
D15	Junto con D16 determina sentido de giro motor izda.
D16	Junto con D15 determina sentido de giro motor izda.
D17	Junto con D18 determina sentido de giro motor dcha.
D18	Junto con D17 determina sentido de giro motor dcha.
GND	Plano de tierra que cubre toda la placa, está dividido en 3 zonas: potencia, analógico y digital
Motor1	Conecta salida 1 del driver a un borne del motor izquierdo
Motor2	Conecta salida 2 del driver a un borne del motor izquierdo
Motor3	Conecta salida 3 del driver a un borne del motor derecho
Motor4	Conecta salida 4 del driver a un borne del motor derecho
N0006!Sensores tcrt1000!	Conecta resistencia con el ánodo del sensor 1
N0007!HCSR04 Y ZUMB!	Conecta la resistencia del zumbador al zumbador
N0007!Sensores tcrt1000!	Conecta resistencia con el ánodo del sensor 2
N0008!Sensores tcrt1000!	Conecta resistencia con el ánodo del sensor 3
N0009!Bluetooth esquema!	Conecta Rx del Bluetooth al divisor de tensión
N0009!Sensores tcrt1000!	Conecta resistencia con el ánodo del sensor 4
N0011!Sensores tcrt1000!	Conecta resistencia con el ánodo del sensor 5
Sensor1	Conecta emisor del sensor 1 con la entrada 3 del schmitt trigger
Sensor2	Conecta emisor del sensor 2 con la entrada 2 del schmitt trigger

Sensor3	Conecta emisor del sensor 3 con la entrada 6 del schmitt trigger
Sensor4	Conecta emisor del sensor 4 con la entrada 5 del schmitt trigger
Sensor5	Conecta emisor del sensor 5 con la entrada 4 del schmitt trigger
VCC	5V que alimenta a 5 TCRT1000, al módulo bluetooth y a las resistencias de los ánodos de los sensores

### 5.7.8. Diseño final

A continuación se muestra el diseño completo del PCB:

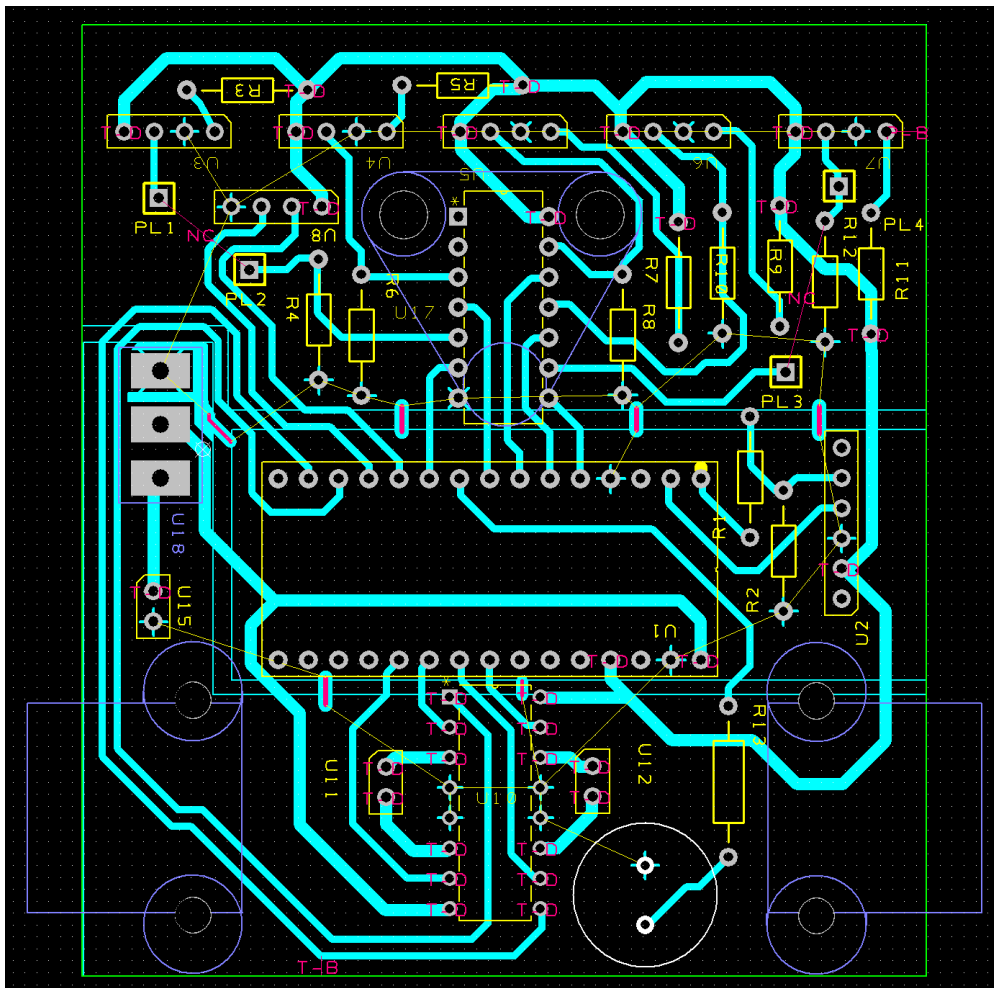


Ilustración 42: Diseño final del PCB sin plano de tierra GND

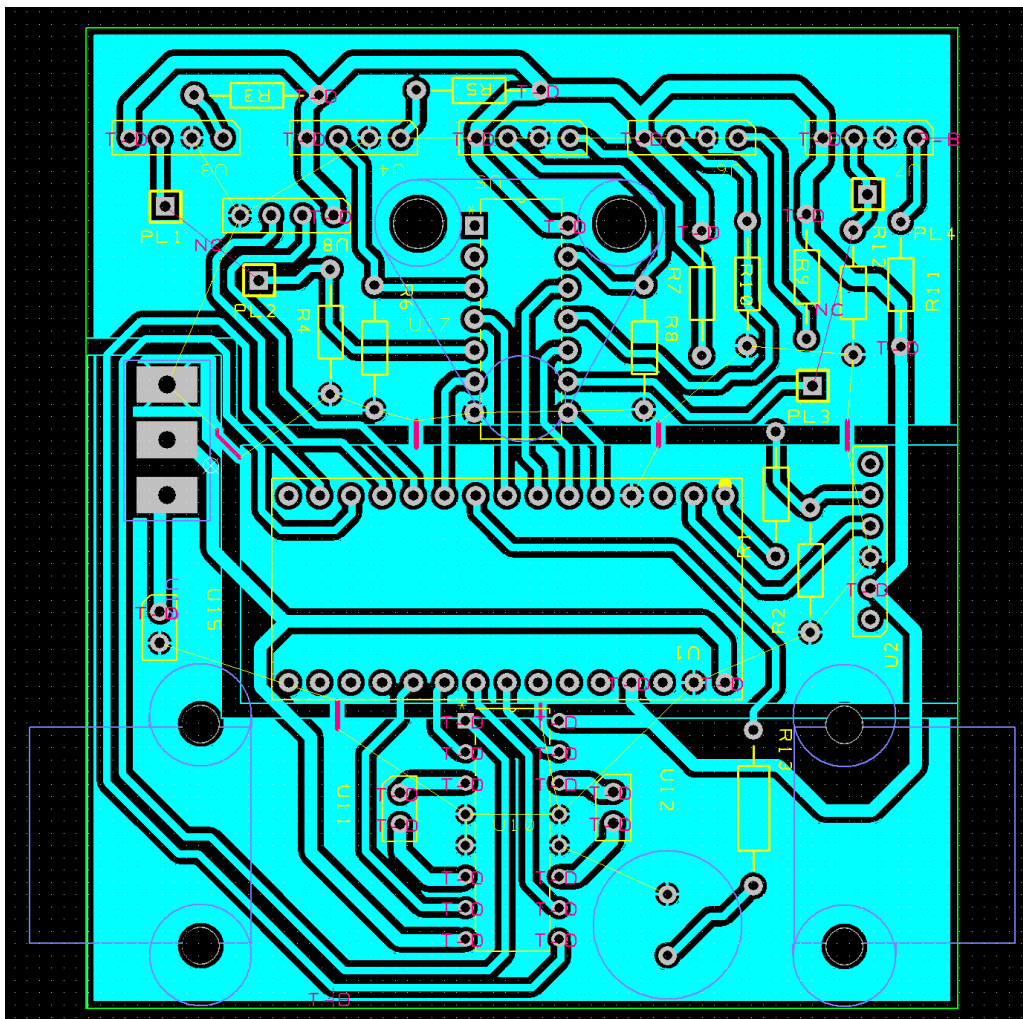


Ilustración 43: Diseño final del PCB con plano de tierra GND.

El programa DesignSpark tiene disponible la herramienta de visualización en 3d del diseño del PCB. A pesar de no proporcionar una representación en 3d fiel a la realidad en algunos componentes, como puede ser en la bola loca o las ruedas, sí que sirve para hacerse una idea de cómo va a quedar el PCB y tiene cierta utilidad. A continuación se muestran varias capturas.

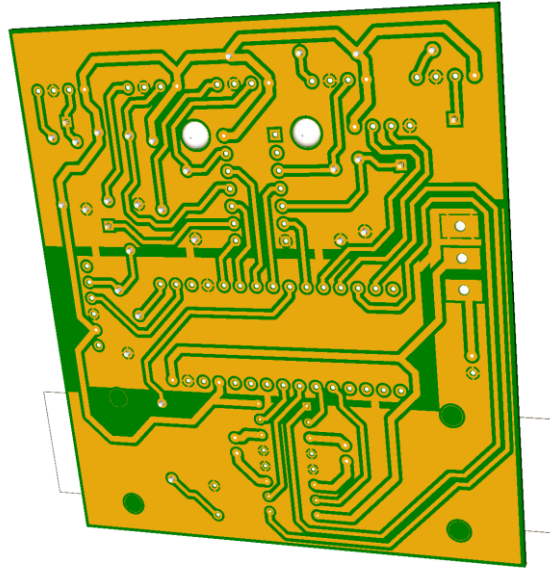


Ilustración 44: Vista en 3d de la parte inferior de la placa

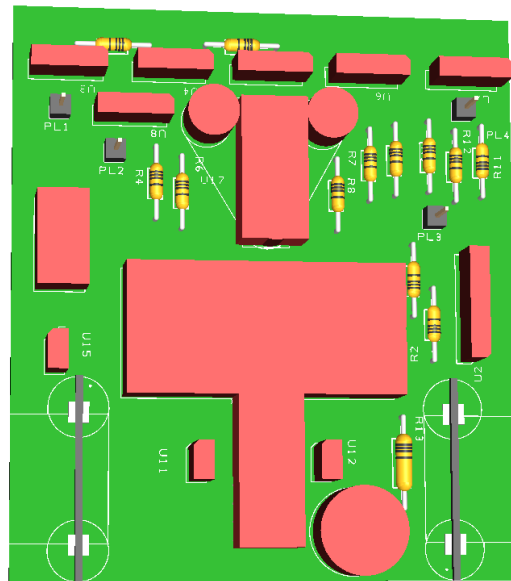
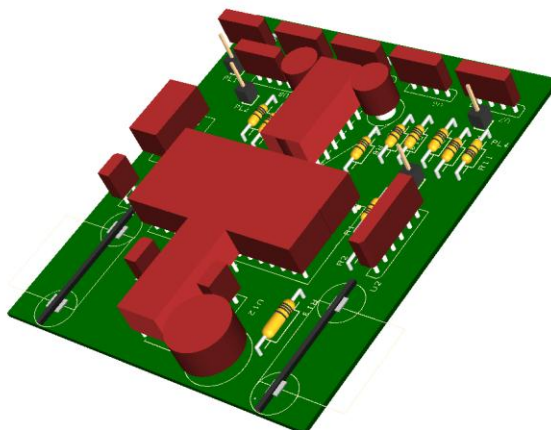


Ilustración 45: Vista en 3d de la parte superior de la placa





**Ilustración 46: Vista general en 3d del PCB**

### 5.7.9. Montaje y soldadura

Una vez diseñado el PCB se entregó el archivo correspondiente al tutor para que se fabricara la placa. Una vez recibida la placa se procedió con la soldadura y el montaje de los distintos componentes. Este proceso fue todo un reto ya que no se tenía experiencia de ningún tipo con la utilización de un soldador. Pensando en este detalle se eligieron las resistencias de carbón frente a las de montaje superficial a pesar de ser éstas una mejor elección a priori por su reducido tamaño.

La parte que presentó mayor complejidad a la hora de soldar fueron los zócalos de los sensores ya que estos zócalos se colocan en la capa bottom para que los sensores queden “mirando” al suelo, de este modo las pistas que llegan al zócalo están en la misma capa que el propio zócalo. Este hecho hace que soldar los pines del zócalo a las pistas sea bastante complejo. Se empezó con el primer zócalo colocándolo un poco por encima de la placa para que hubiera cierto espacio accesible hacia los pines. Al final se consiguió soldar el zócalo pero las conexiones no eran de demasiada calidad y además se había perdido bastante tiempo. Se hizo evidente la necesidad de encontrar alguna alternativa para solucionar este problema. La solución que se adoptó fue limar con cuidado la parte inferior del zócalo para que la parte metálica de los pines escondida tras el plástico quedara a la vista. Con esta solución la soldadura fue muchísimo más sencilla y las conexiones fueron de una calidad más que aceptable.

Pablo Ferro Laspidea

El montaje de los componentes sobre la placa no supuso grandes problemas ya que en el diseño del PCB ya se había calculado los diámetros necesarios para los tornillos y la distancia entre los mismo era la correcta, se utilizaron arandelas de plástico para aislar los tornillos.

Hay que mencionar que fue necesario utilizar un taladro para agrandar los agujeros de las coronas del interruptor ya que los pines del interruptor no son cilíndricos sino que están preparados para conectar un cable. Este hecho estaba previsto y por ello se diseñaron unas coronas muy grandes para el interruptor y no supuso mayor problema que el tiempo dedicado a agrandar los agujeros con el taladro.

Una vez montado y soldado todo se comprobó la continuidad con el polímetro en todas las conexiones, para ello hubo que desmontar la bola loca ya que no permitía el acceso a algunos pines y pistas. Después con la bola loca montada se comprobó que está no nos producía cortocircuitos indeseados aunque a priori esta posibilidad estaba descartada ya que la parte de la bola loca que estaba en contacto con la placa era plástico.

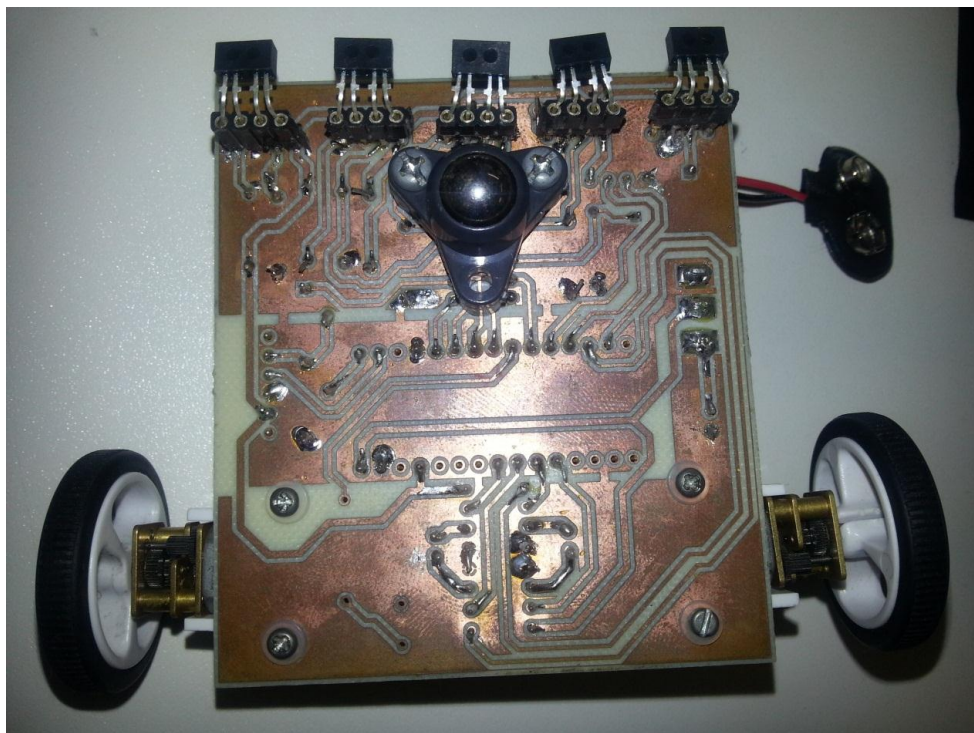


Ilustración 47: Vista posterior PCB soldado y montado

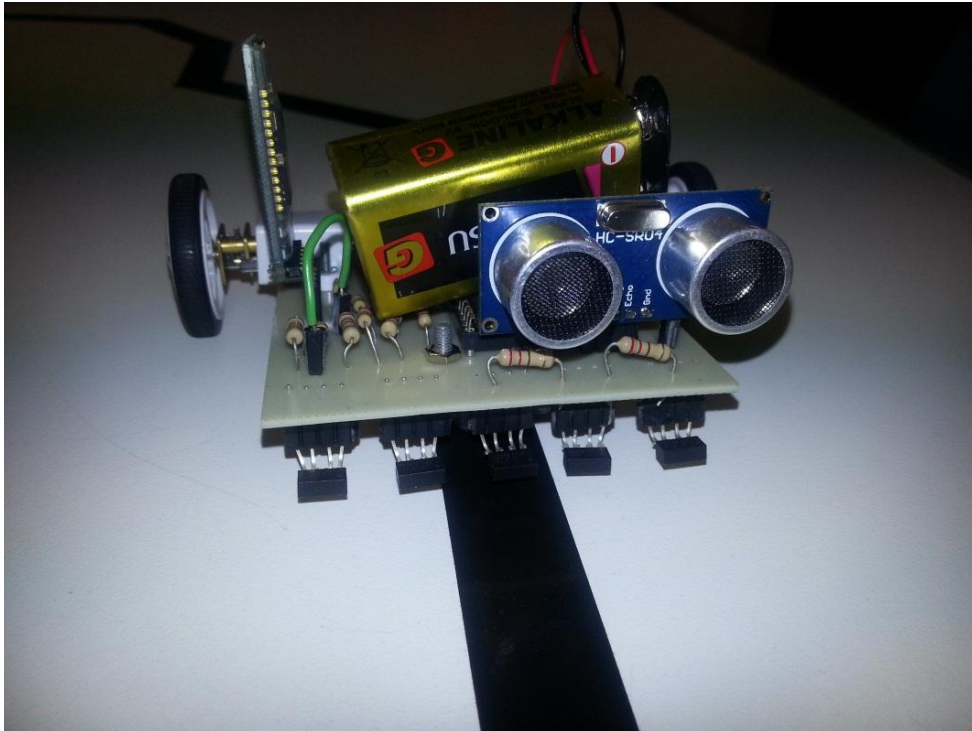


Ilustración 48: Vista frontal robot

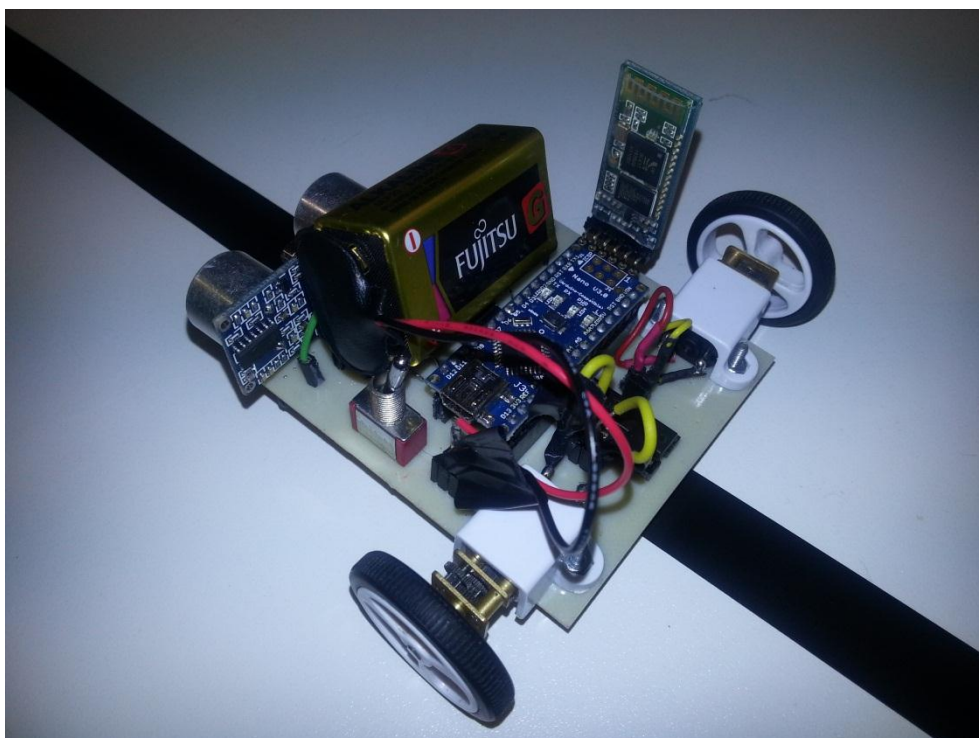


Ilustración 49: Vista lateral del robot

## 6. Presupuesto

### 6.1. Coste de material robot

Apartado	Precio unitario	Cantidad	Importe
Sensores			
TCRT1000	0.886 €	5	4.43 €
HC-SR04	2.71€	1	2.71€
			7.14 €
Alimentación			
Batería 9V	3.15 €	1	3.15 €
			3.15 €
Microcontrolador			
Arduino Nano V.3	31.71 €	1	31.71 €
			31.71 €
Comunicación inalámbrica			
JY-MCU	6.43 €	1	6.43 €
			6.43 €
Motores			
Micrometal motor DC	13.20 €	2	26.40 €
			26.40 €
Integrados			
L293d	3.59 €	1	3.59 €
74HC14	0.81 €	1	0.81 €
			4.4 €
Ruedas			
Ruedas(2)	5.90 €	1	5.90 €
Bola loca	3.15 €	1	3.15 €
			9.05 €
Conectores e interruptores			
Interruptor	1.20 €	1	1.20 €
Tiras de conectores	1.42 €	2	2.84 €
			4.04 €
Componentes electrónicos			
Resistencias	0.02 €	13	0.26 €
Zumbador	2.62 €	1	2.62 €
			2.88 €
PCB (71mm x 80mm)	20 € (estimado)		20 €
<b>TOTAL</b>			<b>115.20 €</b>

El presupuesto de ejecución de material de un prototipo del robot asciende a la cantidad de CIENTO QUINCE EUROS CON VEINTE CÉNTIMOS.

Probablemente el coste del material de una serie de varias decenas o cientos de robots se reduzca considerablemente ya que los precios de los componentes son menores, como por ejemplo el PCB que resulta bastante más barato encargar una tirada que solo uno.

## 6.2. Coste mano de obra

Nº trabajadores	Salario/mes	Meses	Importe
1	1600 €	2	3200 €

El coste de mano de obra asciende a la cantidad de TRES MIL DOSCIENTOS EUROS. El tiempo de trabajo se ha calculado suponiendo una jornada de trabajo de 8 horas diarias, es decir 40 horas semanales. La cantidad de créditos correspondiente a este proyecto es de 12 ECTS, cada uno de ellos equivale a 30 horas de trabajo, por lo que resulta una duración total de 9 semanas de trabajo.

## 6.3. Coste material variado

Concepto	Importe
Material de oficina, fotocopias, cinta aislante, pilas...	25 €

El coste de material variado asciende a la cantidad de VEINTICINCO EUROS.

## 6.4. Presupuesto total

Concepto	Importe
Material de robot	101.20 €
Material variado	25 €
Subtotal	140.20 €
I.V.A.	21%
<b>Total Material</b>	<b>169.64 €</b>
Mano de obra	3200 €
Seguros sociales	30%
<b>Total Mano de obra</b>	<b>4160 €</b>
<b>Presupuesto TOTAL</b>	<b>4329.64 €</b>

El presupuesto total final asciende a la cantidad de CUATRO MIL TRESCIENTOS VEINTINUEVE CON SESENTA Y CUATRO CÉNTIMOS.



## 7. Posibles mejoras

---

Se ha implantado lo que ha dado tiempo pero se tenían múltiples ideas y mejoras en mente que no se han podido implementar.

### 7.1. Mejoras en los sensores

Se podría valorar la posibilidad de aumentar el número de sensores TCRT1000 para obtener mayor información sobre la posición de la línea y mejorar la respuesta del robot. Así mismo se valoraría la opción de colocar pequeño servomotor que girara en un rango de 150 grados y colocar el sensor de ultrasonidos a la salida del servomotor.

De esta forma el sensor barrería un área más grande pudiendo detectar más obstáculos. De todas formas habría que estudiar si el sensor de ultrasonidos nos valdría en esta aplicación en la que está girando continuamente o hay que utilizar otro sensor de otro tipo.

### 7.2. Toma de datos analógicos

Se valoraría la posibilidad de utilizar la tensión que nos proporcionan los sensores sin convertirla a un valor digital de HIGH o LOW, es decir utilizando un conversor analógico digital con la suficiente resolución para que el valor digital convertido sea una representación fiel de la posición de la línea.

Al utilizar valores analógicos contaríamos con muchísima más información pudiendo seguramente desarrollar un sistema de control más sofisticado.

### 7.3. ZigBee

Se podría valorar la utilización de módulos ZigBee en vez de Bluetooth por su menor consumo energético y su mayor capacidad de formar nodos. Por el contrario su velocidad de transferencia de datos es menor y si el sistema se desarrolla volviéndose más complejo podría no ser suficiente. Además se supone que el futuro robot de almacén tendría unas dimensiones considerables por lo que podría llevar una batería de gran capacidad.

## 7.4. Código de barras

Consistiría en colocar pegatinas de códigos de barra en el suelo para identificar las distintas estanterías y que el robot dispusiera de un lector de código de barras laser y al leer la pegatina mandara los datos al ordenador. Esta opción se contempló en las fases iniciales del proyecto e incluso se realizaron pruebas en el laboratorio con sensores CNY70 para ver si tenían la resolución suficiente para detectar los flancos del código de barras. Se observó que leía básicamente ruido por lo que habría que buscar otro método para leer los códigos como podría ser con un lector laser con bastante más precisión.

## 7.5. Base de datos

Relacionado un poco con el concepto anterior consistiría en introducir una base de datos en LabView con la ubicación de los productos y que el robot supiera donde está cada producto y que itinerarios debiera tomar hasta llegar hasta él. Todo esto dependería por supuesto de la magnitud del almacén y para implementarlo no bastaría solo con introducir la base de datos en LabView si no que habría que cambiar las rutinas de intersecciones y de más.

## 7.6 Malla pegatinas QR

Esto más que una mejora sería un nuevo enfoque al proyecto ya que es un cambio bastante importante. Al estilo de los robots KIVA se trataría de implementar una malla cuadriculada de pegatinas QR en la que cada una sería una coordenada dentro del almacén. La información sobre la ubicación del robot en el almacén que nos proporcionaría sería muy valiosa ya que sabiendo el destino y la ubicación exacta se podrían desarrollar algoritmos para implementar los itinerarios que podría seguir el robot para llegar a su destino.

Ya no sería un siguelíneas si no que bastaría con que se moviera en línea recta hasta encontrar la siguiente pegatina y según la posición de ésta respecto a su destino seguiría de frente o giraría 90 grados a izquierda o a derecha. Se podría combinar esta estructura de pegatinas QR con una aplicación de siguelíneas colocando una cinta negra que uniera las distintas pegatinas para evitar que el robot perdiera las pegatinas o incluso que hubiera tramos o zonas del almacén donde solo

Pablo Ferro Laspidea

hubiera cinta negra. Probablemente esta última solución sea mejor que utilizar solo la malla sin nada entre pegatinas QR.

## 7.7. Diferentes modos de funcionamiento

Contando con un sensor de ultrasonidos y con el módulo Bluetooth controlado a través de LabView se podría haber desarrollado un programa en el que a partir de lo que mandáramos desde el ordenador el robot se comportara como un siguelíneas, o como un evita obstáculos en el que seguiría de frente hasta encontrar un obstáculo girando en ese momento hasta que lo perdiera de vista y es entonces cuando continuaría de frente.

Incluso se podría haber implementado un modo coche teledirigido en el que desde el LabView controláramos si va para adelante o para atrás o si gira a la derecha o a la izquierda.

## 7.8. Monitorización de datos a través de LabView

Esta funcionalidad consistiría en que el robot mandara información al ordenador a través del módulo Bluetooth. Podría mandar el error de la línea, pudiéndose hacerse gráficas muy útiles para la sintonización Ziegler–Nichols, o que mandará en que numero de pasillos esta por ejemplo. Esta funcionalidad se intentó implementar con bastante insistencia pero como ya se ha comentado más arriba daba problemas de estabilidad en el programa y además no se consiguió realizar las gráficas del todo bien.



## 8. Conclusiones y valoración

---

El objetivo era desarrollar un robot completo, desde el diseño del circuito de los sensores, hasta el diseño del PCB pasando por la programación o la elección de componentes. Por supuesto el desarrollo del proyecto no ha estado exento de problemas e inconvenientes a los que he tenido que hacer frente investigando de forma autónoma y aprendiendo de mis propios errores.

He aprendido que a la hora de diseñar y montar un PCB la soldadura, a priori un aspecto no demasiado relevante, se convierte en crucial y completamente determinante en el buen funcionamiento del circuito. Ya se puede tener todo el circuito bien diseñado y planteado que si hay un pequeño error de soldadura, nos va a dar problemas que además son muy difíciles de detectar ya que incluso cuando el test de continuidad con el polímetro nos dice que está bien soldado, no garantiza que vaya a funcionar de manera correcta ya que igual hay algún contacto que se mueve o el estaño no recubre al pin de forma adecuada. En el desarrollo del proyecto se ha presentado en varias ocasiones este problema.

He aprendido que cuando se está realizando un proyecto siempre hay que estar sujetos a la posibilidad de que se produzcan imprevistos ajenos a nuestra voluntad y responsabilidad. En el desarrollo del proyecto se produjeron dos imprevistos importantes como son que el módulo de ZigBee no funcionara o que el Arduino Nano no dejara programar.

Derivado de lo anterior he aprendido que no hay que centrarse demasiado en una cosa y que hay que intentar buscar siempre alternativas para solucionar los problemas e imprevistos. Esto lo he aprendido a base de equivocarme ya que se perdió muchísimo tiempo con el problema de configuración que daba el módulo ZigBee con el programa X-CTU y con el error que daba el Arduino Nano al intentar programarlo desde el ordenador. Con estos 2 problemas se perdieron semanas, pasando horas y horas en foros y páginas intentando solucionarlo y al final la solución fue utilizar Bluetooth en vez de ZigBee y comprar un Arduino nuevo.

También he aprendido que las cosas nunca salen a la primera ni a la segunda, se puede tener en mente que lo que has hecho está bien y está bien pensado y cuando lo pones en marcha hace todo lo contrario a lo que esperabas. He aprendido a

Pablo Ferro Laspidea

resolver los problemas por mi propia cuenta a base de buscar y buscar y buscar donde está el error e intentar solucionarlo.

Como se ha visto en el apartado de posibles mejoras, con tiempo y dedicación las posibilidades son casi infinitas, pero el tiempo es limitado y además el desarrollo del proyecto se ha realizado en paralelo con un semestre bastante complicado porque he tenido 2 asignaturas extras repetidas del 6º semestre y ha sido bastante estresante para mí.

Hay que destacar que en la realización del proyecto se han utilizado conceptos vistos en el grado relacionados con la programación de microcontroladores , el control automático, circuitos eléctricos, utilización de sensores analógicos, uso sistemas de comunicación, diseño de circuitos impresos, etc... Además se han usado programas vistos en el grado como el LabView o el DesignSpark.

Por todo esto el desarrollo del proyecto ha sido todo un desafío para mí en el que considero que he aprendido bastante, no solo a realizar un proyecto completo por mi cuenta sino también a cambiar un poco la mentalidad que había tenido hasta ahora en el grado pasando de un enfoque teórico a un enfoque más práctico, a aplicar los conocimientos adquiridos en un proyecto de ingeniería industrial.

Considero que la realización del proyecto de fin de grado ha sido bastante positiva y me ha ayudado en mi formación como ingeniero.

## 9. Bibliografía

---

### 9.1. Recursos electrónicos

<http://letsmakerobots.com/node/39972>

[http://www.madridbot.org/sensores\\_cny70.htm](http://www.madridbot.org/sensores_cny70.htm)

<http://letsmakerobots.com/content/new-beginner-robotics?page=1>

[http://es.wikipedia.org/wiki/Motor\\_el%C3%A9ctrico](http://es.wikipedia.org/wiki/Motor_el%C3%A9ctrico)

[http://es.wikipedia.org/wiki/Motor\\_de\\_corriente\\_continua](http://es.wikipedia.org/wiki/Motor_de_corriente_continua)

[http://es.wikipedia.org/wiki/Motor\\_de\\_corriente\\_alterna](http://es.wikipedia.org/wiki/Motor_de_corriente_alterna)

[http://es.wikipedia.org/wiki/Motor\\_paso\\_a\\_paso](http://es.wikipedia.org/wiki/Motor_paso_a_paso)

<http://es.wikipedia.org/wiki/Servomotor>

[http://es.wikipedia.org/wiki/Puente\\_H\\_\(electr%C3%B3nica\)](http://es.wikipedia.org/wiki/Puente_H_(electr%C3%B3nica))

<http://users.ece.utexas.edu/~valvano/Datasheets/L293d.pdf>

[http://es.wikipedia.org/wiki/Modulaci%C3%B3n\\_por\\_ancho\\_de\\_pulsos](http://es.wikipedia.org/wiki/Modulaci%C3%B3n_por_ancho_de_pulsos)

<https://www.google.es/imagenes>

[http://es.wikipedia.org/wiki/Circuito\\_impreso](http://es.wikipedia.org/wiki/Circuito_impreso)

[http://es.wikipedia.org/wiki/Proporcional\\_integral\\_derivativo](http://es.wikipedia.org/wiki/Proporcional_integral_derivativo)

<http://www.superrobotica.com/S320107.htm>

<http://tallerarduino.com/2011/12/06/modulo-bluetooth-hc-06-o-gp-gc021-y-arduino/>

<http://www.kivasystems.com/about-us-the-kiva-approach/history/>

### 9.2. Recursos bibliográficos

Se han utilizado apuntes de algunas asignaturas del grado de los que se ha extraído ideas y material.