UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Diseño e implementación de un robot móvil basado en microcontrolador

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Miguel Rodríguez Sorní

Tutor: Alberto José Pérez Jiménez

2020-2021

Agradecimientos

Don Alberto José Pérez Jiménez por su implicación y dirección del proyecto.

La universidad politécnica de Valencia por brindarme la oportunidad de desarrollar este proyecto.

Mis familiares y amigos por el apoyo recibido durante el desarrollo del mismo.

Resumen

Las tecnologías de la información están hoy presentes en muchos de los ámbitos domésticos y empresariales. En este caso, están surgiendo nuevos conceptos, como el *big data* o el internet de las cosas, en los que los sensores juegan un papel muy importante y es que los sensores se encargan de definir el entorno y obtener diversos datos sobre él.

Para el uso de estos datos es necesario su análisis ya que sino no serían de mayor utilidad. Los microcontroladores son usados para transformarlos en información útil y conocimiento para la toma de decisiones. De esta forma, mediante la interconexión de los sensores y comunicación entre dispositivos, es posible automatizar tareas en función de los datos de entrada.

En este trabajo de fin de grado se ha desarrollado un robot móvil con varias funcionalidades mediante el uso del microcontrolador Arduino UNO, que le permitirá ser controlado remotamente y que actúe de forma autónoma, interactuando con su entorno. Además, se estudian las posibilidades actuales en el mercado para la ampliación o el desarrollo de robot, visto que las oportunidades de crecimiento de este son enormemente amplias.

Para el diseño del robot se ha fabricado un chasis capaz de albergar todos los componentes, entre ellos el microcontrolador Arduino UNO, que funcionará como “cerebro” dotando al robot de sus características de cómputo, y la placa bluetooth la cual le otorga cierta conectividad.

Palabras clave: robot, autónomo, microcontrolador, Arduino, bluetooth, sensores, remoto.

Abstract

Information technology is now present in many areas of the home and business. In this case, new concepts are emerging, such as big data or the Internet of things, in which sensors play a very important role. Sensors are responsible for defining the environment and obtaining various data about it.

In order to use these data, it is necessary to analyse them, otherwise they would be of no use. Microcontrollers are used to transform them into useful information and knowledge for decision making. In this way, through the interconnection of sensors and communication between devices, it is possible to automate tasks based on the input data.

In this end-of-degree project, a mobile robot has been developed with several functionalities through the use of the Arduino UNO microcontroller, which will allow it to be controlled remotely and to act autonomously, interacting with its environment. In addition, the current possibilities in the market for the expansion or development of the robot are being studied, given that the opportunities for growth are enormous.

For the design of the robot, a chassis has been manufactured that is capable of housing all the components, among them the Arduino UNO microcontroller, which will function as the “brain” providing the robot with its computing characteristics, and the bluetooth board which gives it certain connectivity.

Keywords: robot, autonomous, microcontroller, Arduino, bluetooth, sensors, remote.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

1.1 Antecedentes

Ilustración 1- Robot aspirador R676

En la actualidad existen multitud de robots capaces de interactuar con el entorno mediante sensores con el fin de automatizar tareas simples y complejas. Uno de los más famosos actualmente a nivel doméstico es el robot aspirador, capaz de aspirar el suelo de la casa de forma autónoma y programada, para posteriormente volver a su base de carga.

Como aproximación a nuestro proyecto existe el robot aspirador R676 [1] de la empresa iRobot, el cual contiene las funcionabilidades anteriormente contempladas y añadiendo la monitorización del robot mediante un dispositivo móvil.

1.2 Contexto y justificación del trabajo

Actualmente existen infinidad de sensores capaces de medir casi la totalidad de las magnitudes físicas y químicas que definen su entorno. Estas mediciones pueden usarse para alimentar con una gran cantidad de información sistemas ad-hoc o de propósito general. El sistema puede definir el entorno mediante variables para decidir su forma de actuar según el valor que tomen estas.

En las ramas de la ingeniería que cada vez aprovechan más el uso de estas funciones, en el caso de la ingeniería informática pueden ser usados para monitorizar temperaturas en un ordenador, uso de CPU, memoria secundaria y principal y un largo etcétera, a su vez, en la rama de la robótica como trataremos en este trabajo, son usados para toma autónoma de decisiones por parte del robot, detectar objetos mediante infrarrojos para no chocar, tomar mediciones de CO2 para cuando sobrepasen un valor establecido avisar…

Por otra parte, otro movimiento que ha facilitado mejoras en este ámbito y en otros muchos relacionados con la informática ha sido el “Do it Yourself” que ha promovido la expansión del desarrollo de software libre con una comunidad que le da apoyo y soporte a iniciativas de este tipo. Esto ha proporcionado mejoras y que exista mucha información sobre algunos proyectos en

concreto que puedan servir para desarrollar proyectos nuevos que a su vez retroalimenten a otros. Un claro exponente de este tipo de ideas y uno de los sistemas más usados ha sido Arduino.

Por tanto, con todo esto, se pretende construir un robot tanto autónomo como manual mediante la placa Arduino UNO, capaz de ser controlado mediante bluetooth a través de una aplicación Android, pudiendo elegir en cualquier momento la opción que haga el robot autónomo para poder desplazarse por el espacio evitando cochar con los elementos o que siga una línea dibujada en el suelo. Este robot tan solo será un prototipo que puede mejorarse para el desarrollo de tareas más complejas incluyendo más inputs y outputs para que pueda realizar nuevas funciones.

1.3 Objetivos del trabajo

Los objetivos principales del proyecto son:

- Diseñar y construir un robot Arduino capaz de desplazarse.

Implementar una aplicación Android capaz de controlar los movimientos del robot y sus distintas funciones.

Además, como objetivos secundarios:

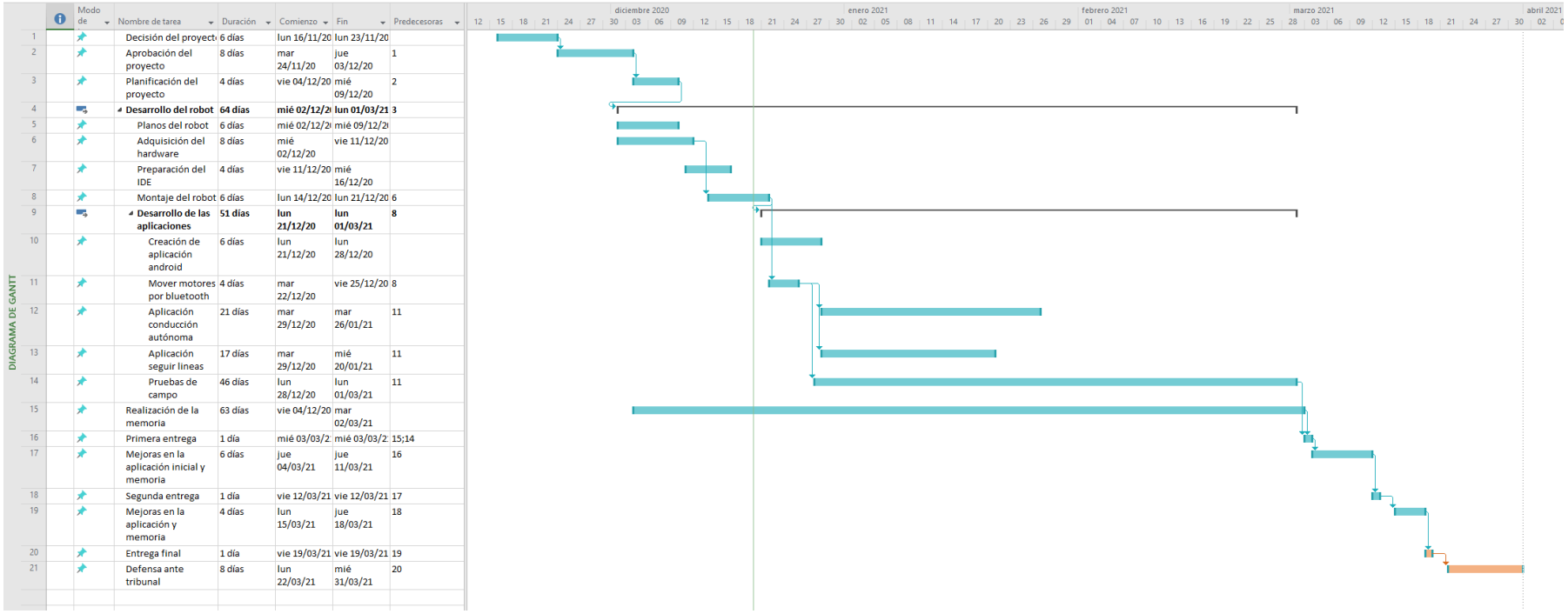
- El robot será accesible mediante la tecnología inalámbrica Bluetooth y estará dotado de varios sensores diferentes.
- Se desarrollará una aplicación Arduino capaz de hacer funcional y autónomo el robot definiendo salidas en función de las variables de entrada proporcionadas por los sensores.
- El robot debe ser ampliable, capaz de adaptarse fácilmente a nuevos entornos, por tanto, será un sistema abierto bien documentado para que pueda incluir en un futuro nuevas versiones o modificaciones.

1.4 Expectativas

Las expectativas durante este proyecto serán mejorar conocimientos personales acerca de programación Arduino y Android, obteniendo también habilidades acerca de la robótica general y circuitos poniendo en práctica todos los conocimientos adquiridos durante el grado de Ingeniería Informática con el objetivo de aportar mediante la memoria nuevas experiencias y conocimientos acerca del tema.

1.5 Planificación del trabajo

Se presenta mediante el diagrama de Gantt la planificación del Trabajo de Fin de Grado:



	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		Decisión del proyecto	6 días	lun 16/11/20	lun 23/11/20	
2		Aprobación del proyecto	8 días	mar 24/11/20	jue 03/12/20	1
3		Planificación del proyecto	4 días	vie 04/12/20	mié 09/12/20	2
4		**Desarrollo del robot**	**64 días**	**mié 02/12/2**	**lun 01/03/21**	3
5		Planos del robot	6 días	mié 02/12/2	mié 09/12/2	
6		Adquisición del hardware	8 días	mié 02/12/20	vie 11/12/20	
7		Preparación del IDE	4 días	vie 11/12/20	mié 16/12/20	
8		Montaje del robot	6 días	lun 14/12/20	lun 21/12/20	6
9		**Desarrollo de las aplicaciones**	**51 días**	**lun 21/12/20**	**lun 01/03/21**	**8**
10		Creación de aplicación android	6 días	lun 21/12/20	lun 28/12/20	
11		Mover motores por bluetooth	4 días	mar 22/12/20	vie 25/12/20	8
12		Aplicación conducción autónoma	21 días	mar 29/12/20	mar 26/01/21	11
13		Aplicación seguir lineas	17 días	mar 29/12/20	mié 20/01/21	11
14		Pruebas de campo	46 días	lun 28/12/20	lun 01/03/21	11
15		Realización de la memoria	63 días	vie 04/12/20	mar 02/03/21	
16		Primera entrega	1 día	mié 03/03/2	mié 03/03/2	15;14
17		Mejoras en la aplicación inicial y memoria	6 días	jue 04/03/21	jue 11/03/21	16
18		Segunda entrega	1 día	vie 12/03/21	vie 12/03/21	17
19		Mejoras en la aplicación y memoria	4 días	lun 15/03/21	jue 18/03/21	18
20		Entrega final	1 día	vie 19/03/21	vie 19/03/21	19
21		Defensa ante tribunal	8 días	lun 22/03/21	mié 31/03/21	20

Ilustración 2 – Planificación del proyecto de fin de grado

1.6 Breve descripción de próximos apartados de la memoria

En los siguientes apartados se detallará el proceso de este trabajo:

En el apartado de introducción se ha realizado un repaso de los antecedentes al proyecto, el contexto actual de las tecnologías a utilizar, los objetivos del trabajo y la planificación de este.
En el segundo apartado se realizará un análisis del problema y de la solución propuesta para el mismo.
En el tercer apartado se detallarán los aspectos del microprocesador que se va a utilizar (Arduino) comparándolo con el microcontrolador PICAXE.
En el cuarto apartado nos centraremos en el robot, detallando los sensores, su conectividad y las capacidades de expansión que podría tener el mismo.
En el quinto capítulo se mostrará y se explicará el diseño futuro del robot y la organización de sus componentes, así como su diseño eléctrico.
En el sexto capítulo se mostrará cómo se ha realizado el chasis y cómo han quedado los diseños del apartado anterior.
En el séptimo capítulo se mostrará el entorno de programación, se preparará y se comprobará el correcto funcionamiento de los sensores.
En el octavo capítulo se implementará el código Arduino de cada uno de los subprogramas y del programa final.
En el noveno capítulo se explicará cómo se ha realizado la aplicación Android para el control del robot.
En el décimo capítulo se mostrará el presupuesto empleado para la realización del proyecto.
En el undécimo capítulo, se reflejarán las conclusiones que se han sacado mediante la realización del proyecto.
Por último, los apartados duodécimo, decimotercero y decimocuarto se usarán para la consulta de referencia al glosario, bibliografía y anexos.

2. Análisis del problema

Hoy en día, cuando un cliente adquiere un robot, este generalmente sirve para realizar un tipo de tarea en concreto. Cuando los clientes necesitan que los robots efectúen otro tipo de tareas diferentes, tienen la necesidad de adquirir otro robot, ya que el adquirido anteriormente, casi con toda probabilidad no sea capaz de realizar o adaptarse a la nueva tarea por haber sido programado para realizar una tarea en concreto.

El objetivo de este nuevo robot es conseguir que este sea modular y capaz de realizar cualquier tarea según las necesidades del cliente, sin necesidad de que este adquiera otro robot totalmente nuevo, simplemente adquiriendo nuevas piezas para ampliar el que ya tiene.

Para definir el modelo de negocio para el producto se ha realizado la matriz CREA.

Tabla 1 - Matriz CREA

Eliminar	**Aumentar**
El concepto de robot tradicional La compra de diversos robots para cada una de las tareas	Productividad de las empresas Rentabilidad de las empresas
Reducir	**Crear**
Costes de mantenimiento Costes de adquisición de la empresa	Producto único y novedoso

Una vez definido el modelo de negocio para el producto se ha hecho uso del análisis FODA, también conocido como análisis DAFO, para realizar un estudio de la situación del proyecto analizando las características internas y su situación externa.

Tabla 2 - Matriz FODA/DAFO

	De origen interno	**De origen externo**
Puntos débiles	**Debilidades** - Diseño poco atractivo - Prototipo.	**Amenazas** - Crisis económica actual que afecta al sector. - Inestabilidad política en el país.
Puntos fuertes	**Fortalezas** - Idea innovadora. - Producto único actualmente en el mercado.	**Oportunidades** - Mejorar el prototipo para que adquiera más habilidades o resulte más modular.

3. Arduino vs PICAXE

Una de las primeras decisiones a tomar en la realización de este proyecto es el microcontrolador que se va a usar para gobernar el robot.

Para esta decisión he creado una tabla donde compararé dos microcontroladores famosos en el mercado como son Arduino y PICAXE [2].

Tabla 3 - Comparativa Arduino UNO REV 3 y AX408

	Arduino Uno REV 3	AXE408
Tipo de MCU	Atmega 8 bits	Microchip 8 bits
Modelos utilizados	ATmega328 (5V)	PIC18F14K22, PIC16F88, PIC16F684, PIC12F683, entre otros
Arquitectura	Advanced RISC	RISC
Apto para principiantes	Depende del dominio de lenguajes de programación	Si
Lenguaje de programación	C	Basic/diagramas de flujo
Documentación	Si, en la página web de Arduino y en su foro.	Sí, dispone de 3 manuales y soporte en el foro
Simulación con software IDE	no	Si
Hardware de programación	Puerto USB	Puerto serie
Programación IN CIRCUIT	Si	Si
Firmware	Bootloader	Bootstrap
Empaquetado	20 pines (14 digitales de los cuales 6 PWM y 6 analógicos)	14 pines (6 analógicos inputs y 8 digitales outputs)
Open Hardware	si	No
Frecuencia de trabajo	16 Mhz	4 Mhz
Oscilador	Externo	Interno y externo
Instrucciones por ciclo de reloj	1 ciclo de reloj	Más de 1 ciclo de reloj
EEPROM disponible	ATmega328 1024 bytes	256 bytes compartidos con el programa
Precio	20€ (arduino.com)	27,58€ (picaxe.com)

Una vez analizada la tabla, Arduino Uno REV 3 a parte de resultar más potente que el microprocesador AXE408 otorga ventajas respecto al mismo como el Open Hardware y su lenguaje de programación en C que considero más versátil para programadores, dejando a un lado la programación por bloques y diagramas de flujo de AXE408, además Arduino Uno REV 3 ofrece más pines que AXE408, en concreto 6 pines más, que serán de especial ayuda para la conexión de los diferentes sensores y a la posible expansión del robot en un futuro, lo que lo hace más modular.

4. Robot y sus componentes

La Real Academia Española (RAE), define un robot como “Máquina o ingenio electrónico programable que es capaz de manipular objetos y realizar diversas operaciones.”

A continuación, se muestra una imagen del robot Arduino que construiremos a lo largo de este proyecto.

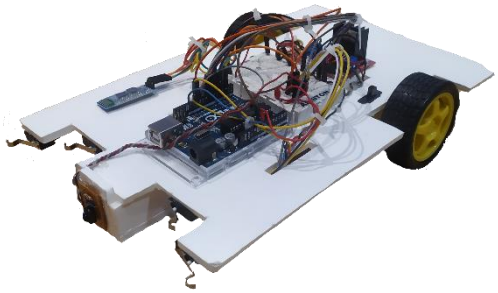


Ilustración 3 – Robot Arduino UNO.

Como se puede apreciar el robot esta alimentado por la placa Arduino Uno REV 3 introducida en el anterior apartado, además se puede apreciar el módulo Bluetooth HC-06, una placa de pines ARISTON, 2 interruptores, 2 ruedas y módulo controlador de motores L298N, el resto de los componentes como las pilas que alimentaran al robot y resto de sensores están en la parte inferior del mismo.

3.1. Cartón pluma

Para la creación del chasis del robot hemos utilizado el material cartón pluma, una plancha de poliestireno expandido recubierto por las dos caras de cartón. El cartón pluma, también es conocido bajo el nombre FOAM, que se toma de uno de los cartones pluma más conocidos de todo el mundo.

El cartón pluma se caracteriza por su ligereza, rigidez y su fácil manipulación, además de su bajo coste, por lo que es ideal para este tipo de proyectos, con el fin de reducir pesos en el robot y conseguir mayor movilidad.

Ilustración 4 - Papel pluma.

Para decidir el diseño del chasis primero se ha realizado un diseño 2D del robot mediante la herramienta de libreCad añadiendo todos los componentes del cual se compondrá para poder distribuirlos sobre la base de la forma más eficiente posible, compensando pesos y teniendo en cuenta un buen agarre por parte de todos los componentes al chasis.

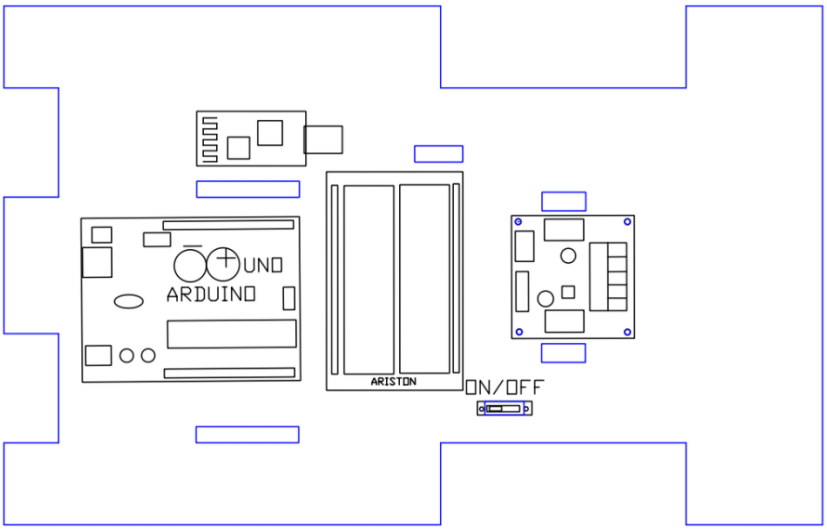


Ilustración 5 - Vista superior del boceto del robot en Librecad.

3.2 Arduino UNO

Para la programación del robot, como se ha explicado anteriormente, se ha optado por el uso del microcontrolador Arduino UNO, una placa basada en un microcontrolador ATMEL.

Los microcontroladores son circuitos integrados en los que se pueden grabar instrucciones, en este caso mediante la herramienta Arduino IDE, dando la posibilidad a la creación de programas que interactúen con los circuitos de la placa.

Ilustración 6 - Microcontrolador Arduino UNO

3.3 Módulo controlador de motores L298N

Para la conexión del microcontrolador Arduino UNO con los motores de corriente continua se ha utilizado un módulo controlador de motores, en concreto el L298N. Este módulo permite controlar la dirección y potencia de dos motores de corriente continua gracias a los dos H-bridge que monta.

H-bridge es un componente formado por 4 transistores que nos permite invertir el sentido de la corriente pudiendo invertir el sentido de giro del motor.

El rango de tensiones en el que trabaja el módulo va desde los 3 voltios hasta los 35V. Además, el módulo cuenta con un regulador de tensión que nos permite obtener una tensión de 5V de salida adecuada para alimentar el Arduino. En este caso no se hará uso de esta funcionalidad ya que se ha considerado más apropiado que los motores sean alimentados por una fuente de alimentación independiente.

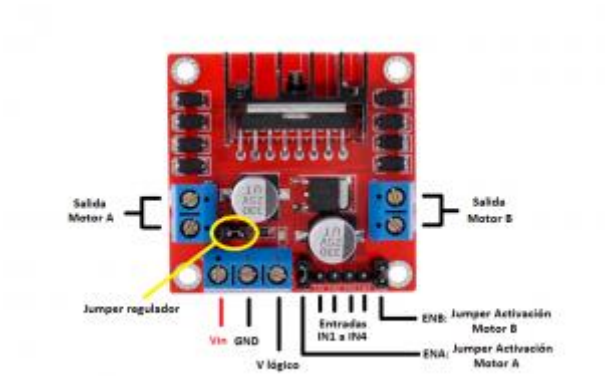


Ilustración 7 - Modulo controlador de motores L298N

3.4 Motores

Para el movimiento del robot he optado por dos motores de corriente continua a 5 Voltios conectados a unas ruedas de goma, con el fin de controlar su movimiento cambiando la polaridad de cada uno en función de la dirección deseada.

Ilustración 8 - Rueda con motor de 5V.

3.5 Placa de conexiones

Con el fin de hacer el montaje más modular y ordenado, se ha decidido utilizar una placa de conexiones ARISTON para conectar algunos de los sensores, los positivos y las toma a tierra, de esta forma se ha conseguido un diseño ordenado y modular del robot.

Las placas de conexiones ARISTON son placas diseñadas para la creación de circuitos electrónicos mediante la conexión de cables, resistencias, leds y otro tipo de componentes.

Ilustración 9 - Placa de conexiones ARISTON.

3.6 Modulo bluetooth HC-06

Para conseguir mayor conectividad con el robot se ha decidido hacer uso del módulo bluetooth HC-06, el cual se comporta como esclavo, esperando peticiones de conexión las cuales transmitirá a la placa Arduino y viceversa.

La elección del módulo HC-06 ante otros como puede ser el HC-05, ha sido que el módulo HC-06 funciona únicamente como Slave mientras otros módulos como el HC-05 también pueden funcionar como Master, como en este caso el único uso que se le dará será de Slave, no se ha considerado conveniente comprar otro con más funcionabilidades ya que no serían utilizadas.

Este módulo bluetooth será la clave para la conexión entre la placa Arduino UNO y la aplicación Android destinada para el manejo del robot.

Ilustración 10 - Modulo bluetooth HC-06

3.7 Finales de carrera

Para detectar el contacto con paredes y objetos que pueden pasar desapercibidos por el detector de IR frontal se ha hecho uso de cuatro interruptores final de carrera, estos interruptores funcionan de forma que cuando algún objeto ejerce presión sobre ellos, cierran el circuito dejando pasar la corriente y por lo tanto, mandado una pulsación a uno de los pines de la placa Arduino UNO para que esta tome las decisiones oportunas.

Se han ubicado cuatro finales de carrera en la parte frontal del robot.

Ilustración 11 - Final de carrera

3.8 Sensores Infrarrojos

Para la detección de objetos sin necesidad de que el robot establezca un contacto, dado que los contactos continuados pueden provocar futuros fallos en el robot, se emplean tres detectores de infrarrojos, configurados para detectar objetos cercanos a una distancia aproximada de 4 centímetros.

Uno de estos sensores ha sido situado en la parte frontal del robot para detectar obstáculos en su camino mientras avanza, mandando una pulsación al microcontrolador Arduino UNO cada vez que se acerque a uno de ellos, accionando los motores para que estos corrijan su trayectoria evitando una colisión.

Los otros dos sensores infrarrojos han sido situados en la parte inferior del dispositivo con el fin de detectar el color negro o el final del suelo, de esta forma cuando los sensores detecten un color negro mandaran una pulsación a la placa Arduino y de la misma forma cuando no detenten un objeto cercano, en este caso el suelo. Estos sensores también pueden ser aprovechados para hacer que el robot siga un circuito constituido por una línea negra ya que detectaran la línea y ayudaran al robot a corregir la trayectoria de forma que el robot siga la misma.

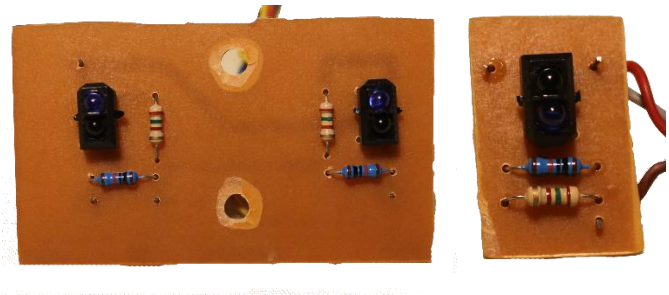


Ilustración 12 - Sensores IR inferior y frontal

3.9 Pila de 9 Voltios

Para la alimentación de los motores se ha optado por el uso de una pila de 9 voltios.

Ilustración 13 - Pila de 9 voltios.

3.10 Porta pilas de 6 V

Para la alimentación de la placa Arduino y del módulo bluetooth HC-06 se ha optado por un porta-pilas de 6 Voltios que incluirá cuatro pilas de tipo AA.

Un porta-pilas es el dispositivo contenedor de pilas, generalmente fabricado en plástico, el cual resguarda a la pila y permite una conexión segura.

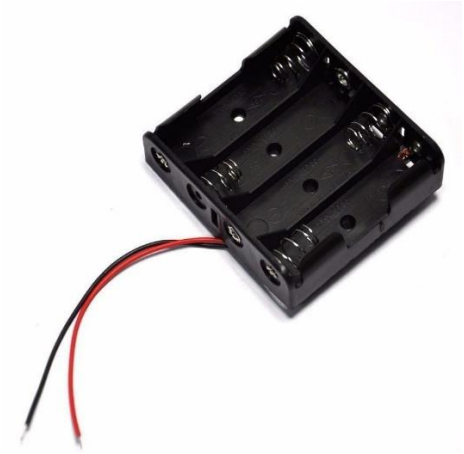


Ilustración 14 - Portapilas de 4 pilas AA 6 voltios

3.11 Interruptores

Para la detección de obstáculos en el entorno se ha hecho uso de cuatro interruptores con el fin de detectar los posibles choques que pueda sufrir el dispositivo para posteriormente corregir su trayectoria.

Un interruptor es un dispositivo para abrir o cerrar el paso de corriente eléctrica en un circuito.

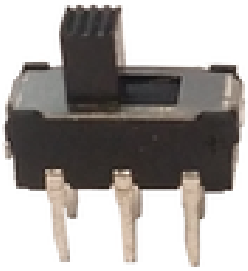


Ilustración 15 - Interruptor negro

3.12 Rueda unidireccional

Para el apoyo de la parte frontal del vehículo con el suelo se ha hecho uso de una rueda de carril unidireccional, con el fin de facilitar la movilidad del robot.

Ilustración 16 - Rueda unidireccional

3.13 Dispositivo Android

Hoy en día es común ser propietario de un Smartphone. La mayoría de estos Smartphones actualmente usan el sistema operativo Android desarrollado por Android Inc. financiada por GOOGLE.

Los Smartphones son computadores con una potencia de cálculo notable, conteniendo varios de estos varios núcleos en su procesador y una gran capacidad en la memoria principal. A la vez, estos dispositivos cuentan con un gran número de sensores como el GPS, el acelerómetro, giroscopio y otros muchos, además de ofrecer una gran conectividad gracias a su conexión WIFI, Bluetooth e Infrarrojos en algunos.

Es por todo ello que se ha optado por usar el sistema operativo Android para manejar el robot a distancia mediante su conectividad bluetooth.

Ilustración 17 - Dispositivo android

3.14 Arduino IDE 1.8.13

Para la programación se ha hecho uso de la aplicación Arduino IDE, un programa de software libre programada en Java por la empresa Arduino LLC.

Para su programación se ha hecho uso del lenguaje C++ con una adaptación proveniente de avr-libc que provee una librería para usar C++ con GCC.

Ilustración 18 - Arduino IDE

3.15 Android APP

Para el control del robot, como se ha explicado anteriormente, se hará uso de una aplicación programada para dispositivos Android mediante el uso de la herramienta online de MIT App Inventor [3].

La aplicación contendrá las funcionalidades que soporta el robot, se conectará al mismo mediante bluetooth y podremos desconectarlo en cualquier momento, en caso de que la conexión falle, la aplicación nos notificará del error. Una vez establecida la conexión las posibilidades de movimiento del robot son movimiento manual del mismo, selección del modo autónomo y selección del modo seguir líneas, pudiendo pausar estos dos últimos modos en cualquier momento.

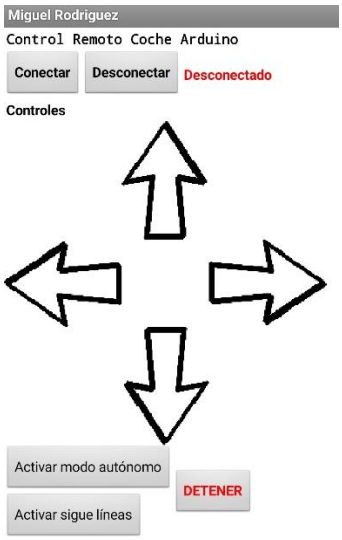


Ilustración 19 - Captura APP Android

3.16 LibreCad

Para la realizar de los planos del robot, los cuales se entregan en el anexo del trabajo, se ha hecho uso de la herramienta LibreCad [4], una herramienta de código libre de diseño asistido para diseño 2D.

Con esta herramienta hemos podido realizar los planos del futuro robot en vistas de alzado y planta, indicando el diseño del chasis, la posición de cada componente en el mismo y el acotado del diseño. Gracias a estas fases previas de planificación y diseño, pudimos realizar el chasis del robot sin cometer fallos estructurales.

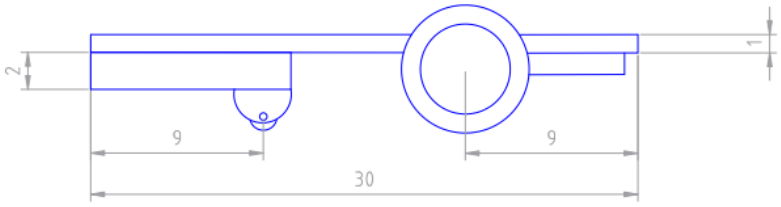


Ilustración 20 - Alzado robot en Librecad

3.17 Fritzing

Mediante el programa Fritzing [5], un programa de software libre de automatización de diseño electrónico se ha realizado el diseño del conexionado eléctrico de los componentes electrónicos del robot, diseñando el circuito y facilitando posteriormente su montaje.

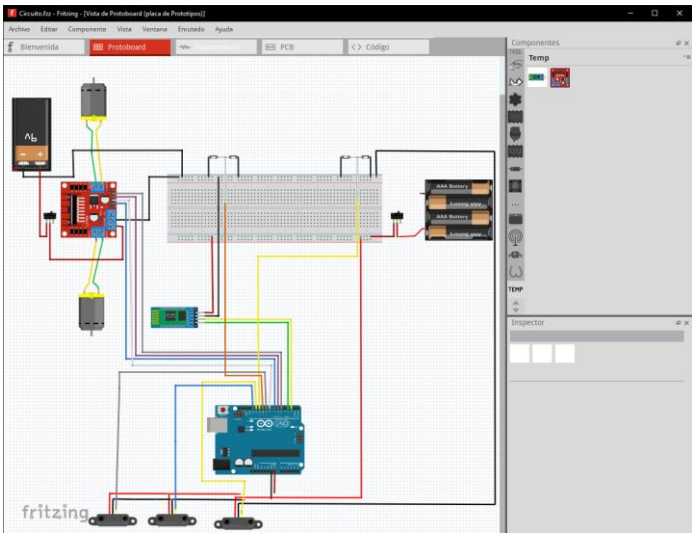


Ilustración 21- Captura del programa Fritzing

5. Diseño del robot

En este apartado se tratará de explicar los pasos seguidos para el diseño del robot, se explicará cómo será el esquema eléctrico, el diseño del chasis según el esquema eléctrico y como se distribuirán los componentes de este para que pueda funcionar correctamente.

El apartado de diseño, como se ha comentado anteriormente, se ha realizado en dos fases, primero se ha creado un esquema eléctrico con el objetivo de saber qué componentes serían necesarios para el correcto funcionamiento del robot, para posteriormente realizar el diseño del chasis distribuyendo los componentes necesarios con el objetivo de organizar los cables de la forma más limpia y óptima posible siendo el resultado del circuito eléctrico mostrado en el anexo de esta memoria.

5.1 Diseño del esquema eléctrico

El esquema eléctrico se ha realizado mediante la herramienta de Fritzing, un programa libre de automatización de diseño electrónico. Esta herramienta incluye nativamente componentes que incorporará el robot, los cuales son la placa microcontroladora Arduino UNO, motores de corriente continua, interruptores, placas Ariston, porta-pilas de 6V, pilas de 9V y sensores de infrarrojos, pero también ha sido necesario introducir el módulo controlador de motores L298N y el módulo bluetooth HC-06 mediante plugins.

5.1.1 Alimentación

Durante el diseño eléctrico se optó por usar pilas independientes para los motores y el resto de los componentes, dado que los motores necesitan más potencia que el resto de los componentes del robot serán alimentados mediante una pila de 9 voltios, dividiendo la alimentación de esta forma, se consigue que el rendimiento de los componentes del robot no se vea afectado por el comportamiento de los motores, estos otros componentes se verán alimentados por un porta-pilas de 6 voltios, que constará de 4 pilas triple AAA que aportan 1,5V cada una e irán conectadas al interruptor que les dará acceso a la placa Ariston donde se conectaran los sensores, el microcontrolador Arduino y el módulo bluetooth para ser alimentados cuando este interruptor se encuentre encendido.

También se ha optado por incluir otro interruptor con el fin de controlar el paso de tensión eléctrica al controlador de motores L298N, que es el encargado de suministrar energía a los dos motores de corriente continua que incluye el robot.

5.1.2 Módulo controlador de motores L298N

El módulo controlador de motores L298N alimentado por la pila de 9 voltios, será el encargado de suministrar energía a los motores para que estos funcionen según las necesidades del sistema, pudiendo también invertir el giro de cada uno de ellos.

Para ello es alimentado desde su conexión Vin, mientras la conexión GND está conectada a la toma tierra que se situará en la placa Ariston.

Con los jumpers de activación ENB y ENA y el jumper regulador activados, las entradas IN1, IN2, IN3 e IN4 irán conectadas a los pines digitales Arduino 7, 8, 6 y 5 respectivamente, desde

los cuales se seleccionará mediante impulsos de 5V el motor a funcionar y su dirección de giro activando las salidas OUT1, OUT2, OUT3 u OUT4 respectivamente, las cuales están conectadas a cada uno de las entradas de alimentación de los motores de corriente continua, OUT1 y OUT2 correspondientes a cada uno de los sentidos de giro del motor de la derecha del robot y OUT3 y OUT4 a los del motor de la izquierda. De esta forma se dotará el robot de movimiento según las necesidades de este.

5.1.3 Módulo Bluetooth HC-06

El módulo bluetooth HC-06 que dotará al robot de conectividad bluetooth para su conexión, en este caso, con la futura aplicación desarrollada posteriormente en Android, consta de 4 pines, correspondientes a VCC, GND, TXD y RXD.

El pin VCC del módulo se conectará a la placa Ariston, junto al positivo del porta-pilas de 6 voltios para que le doté de la energía necesaria para funcionar, mientras GND se conectará como anteriormente el módulo controlador de motores L298N y las dos pilas, a toma tierra también en la placa Ariston. Por último, los pines RXD y TXD, en un principio fueron conectados en los pines 0 y 1 de la Arduino para la configuración del módulo, de lo cual se hablará más adelante, para posteriormente ser situados en los pines 3 y 2 respectivamente para que la placa Arduino pueda recibir los distintos impulsos que genere la aplicación de Android para así que pueda ser controlado remotamente.

5.1.4 Sensores Infrarrojos

Los sensores infrarrojos situados tanto en la parte inferior del dispositivo como en la parte frontal constan de 3 pines cada uno de ellos, uno correspondiente al VCC que se conectará al positivo de la placa Ariston, otro GND que se conectará a la toma tierra de la toma Ariston y por último el encargado de enviar una pulsación cada vez que se detecte un objeto próximo o el color negro.

En el pin 12 de la placa Arduino se situará el sensor inferior derecho, mientras que en el pin 13 se alojará el detector inferior izquierdo para en el pin 9 ser conectado el infrarrojo delantero.

5.1.5 Finales de carrera

Los finales de carrera únicamente contienen 2 pines, uno destinado al GND que irá conectado a la toma tierra en la placa Ariston y otro correspondiente a enviar una pulsación al cerrar el circuito al chocar con algún objeto del entorno del robot.

Este último pin irá conectado según su posición en el robot. Se ha optado por usar en paralelo los dos finales de carrera situados en el mismo lado del robot, esto se hará posible mediante la conexión de estos a la misma columna de la placa Ariston, añadiendo un cable adicional en esta columna que irá conectado a uno de los pines de la placa Arduino con el fin de transmitir el impulso, en caso de los finales de carrera de la parte derecha la pulsación ira destinada al pin número 10 de la placa y en caso de los finales de carrera de la parte izquierda este estará conectado al pin 11 de la placa.

El objetivo de ponerlos en paralelo dos a dos es que, en caso de colisión por la parte izquierda del robot, este retroceda y posteriormente corrija su trayectoria hacia su derecha, evitando así el obstáculo y viceversa.

5.1.6 Conexiones Arduino UNO

La placa Arduino UNO, alimentada mediante su pin Vin que irá conectado a los positivos de la placa de conexiones Ariston y el GND a la toma tierra de la misma placa, estará conectada con el resto de los sensores y componentes siguiendo la siguiente tabla:

Tabla 4 - Distribución de los pines Arduino UNO

Componente	Función	Pin en Arduino
Bluetooh HC-06	TXD	2
	RXD	3
Controlador de motores L298N	Salida motor IN4	5
	Salida motor IN3	6
	Salida motor IN1	7
	Salida motor IN2	8
Sensor IR frontal	Entrada de sensor	9
Finales de carrera derecha	Entrada de sensor	10
Finales de carrera izquierda	Entrada de sensor	11
Sensor infrarrojo inferior derecha	Entrada de sensor	12
Sensor infrarrojo inferior izquierda	Entrada de sensor	13

5.2 Diseño del chasis

Lo primero a tener en cuenta a la hora de diseñar el chasis del robot ha sido el esquema eléctrico creado anteriormente, de esta forma se ha considerado distribuir los componentes de forma que quede un diseño limpio, con los cables lo mejor ordenados posibles, teniendo también en cuenta la distribución de los pesos de cada uno de los componentes para distribuir el peso por la totalidad de la base del robot, con el fin de no crear un desequilibrio y conseguir buena tracción de las ruedas con el suelo, teniendo también en cuenta el material con el que se construirá, en este caso, con una lámina de cartón pluma.

Con todo esto en consideración, se realizará el diseño mediante la herramienta de software libre para el diseño asistido por computadora para dibujo 2D y modelado 3D llamada Librecad, se realizará el chasis del robot que constará de 30cm de largo por 19cm de ancho, dejando huecos de 9cm de largo y 3 de ancho en cada uno de los lados y a 16 cm del frontal para situar las ruedas. Además, se realizarán huecos de distinta longitud y de 1cm de ancho para poder subir los cables de los componentes situados en la parte inferior de este.

Respecto a la distribución de los componentes a lo largo del chasis, se ha optado por situar en la parte inferior los sensores, las dos fuentes de alimentación y las ruedas (incluida la rueda unidireccional que se situará en la parte delantera y central del robot). Los finales de carrera irán situados en la parte frontal con el fin de detectar posibles colisiones del robot, en medio de ellos se situará el sensor infrarrojo frontal, con el fin de evitar los contactos con los objetos pudiendo detectarlos antes para corregir la trayectoria del robot. Respecto a la alimentación se situará la pila de 9 voltios en la parte central y el porta-pilas de 6 voltios en la parte trasera de forma que el peso será distribuido en la parte trasera del robot. Las dos ruedas serán situadas también en la parte inferior a cada lado del robot.

Siguiendo con la parte superior donde se situarán, la placa Arduino posicionada en la parte delantera, con los puertos de conexión apuntando hacia adelante para facilitar la conexión de los cables para la programación del robot. Justo en la parte derecha de la Arduino se situará el módulo bluetooth para que los cables que conectan el módulo y la placa Arduino recorran la menor distancia posible. Seguido de la placa Arduino se encuentra la placa Ariston situada en la parte central del robot seguido del módulo controlador de motores L298N. Por último, los dos interruptores se situarán a los dos lados de la placa Ariston, siendo el situado a la parte izquierda el correspondiente a los motores.

6. Construcción del robot

6.1 Construcción del chasis

Para comenzar, se dibujará sobre el cartón pluma el diseño elaborado anteriormente, posteriormente se procederá a recortarlo mediante el uso de un cúter, al ser el cartón pluma un material ligero y moldeable.

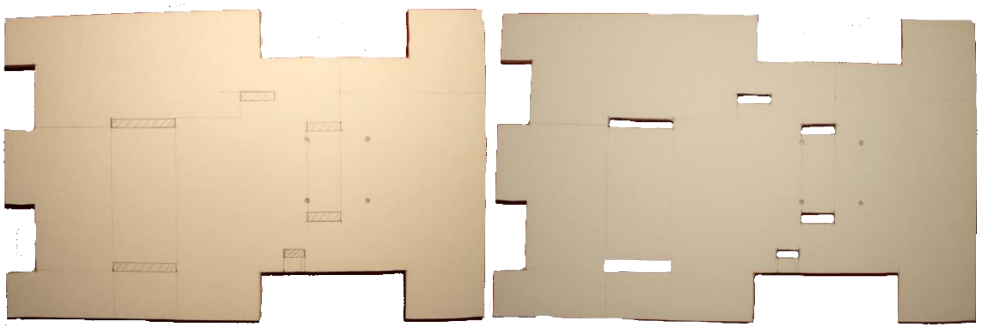


Ilustración 22 - Base del chasis

6.2 Incorporación de los componentes y su conexión

Una vez obtenido el chasis, el siguiente paso será incorporar los distintos componentes y conectarlos según el esquema eléctrico diseñado en pasos anteriores.

Para ello se comenzó integrando los sensores frontales correspondientes a los cuatro finales de carrera y el sensor infrarrojo frontal, se siguió incorporando las tres ruedas, las dos ruedas motrices y la rueda unidireccional que servirá como apoyo. Una vez incorporados estos componentes gracias al uso del termofusible, se prosiguió incorporando los interruptores y los portapilas, para finalizar introduciendo los componentes de la parte superior correspondientes a la placa Arduino, la controladora de motores L298N, el módulo bluetooth y la placa de conexiones Ariston, haciendo ya posible la conexión de todos estos componentes según el esquema eléctrico.

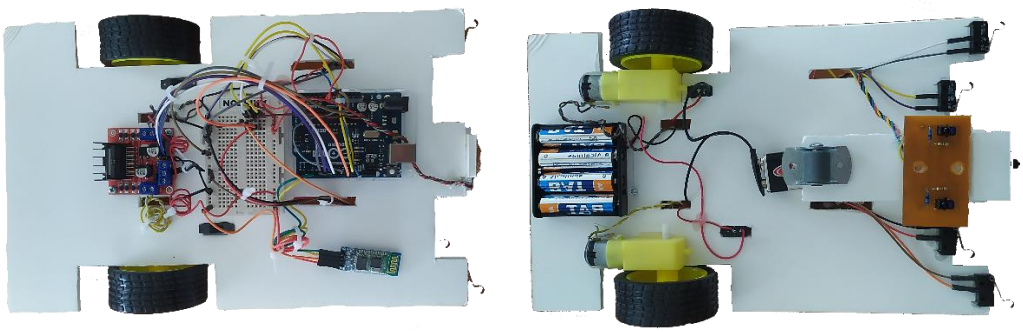


Ilustración 23 - Foto del robot construido

7. Preparación del entorno y comprobación de los sensores

7.1. Preparación del entorno ARDUINO

Lo primero que se debe hacer antes de programar es preparar el entorno de programación, en este caso el IDE de ARDUINO [6] para que sea compatible con la placa ARDUINO UNO, para ello será necesario dirigirse dentro del IDE de ARDUINO al apartado de Herramientas, Placa: y seleccionar el modelo de la placa, en este caso, Arduino Uno.

Una vez realizado este paso ya podremos comenzar a programar el proyecto.

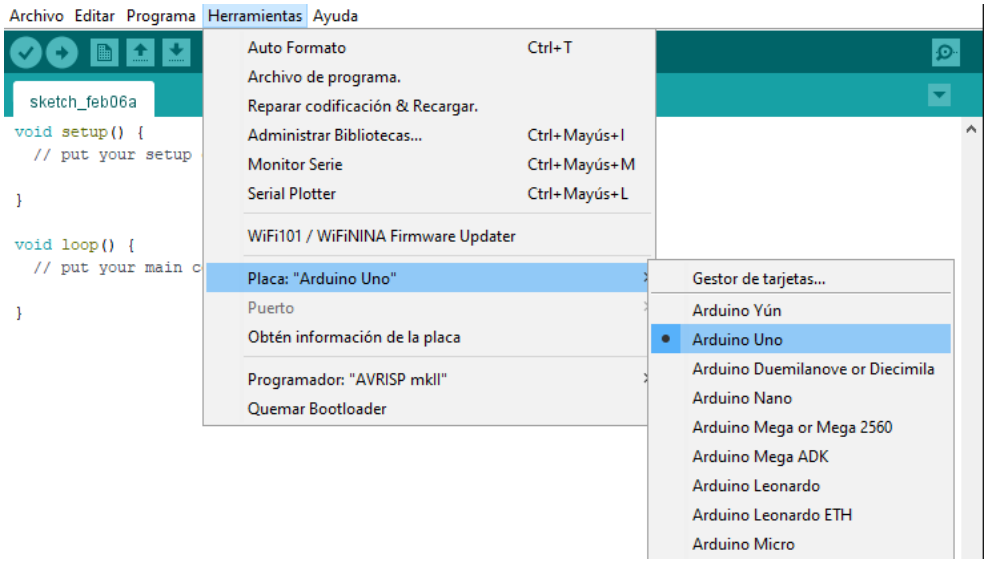


Ilustración 24 - Captura de la preparación del entorno Arduino

7.2. Comprobación del correcto funcionamiento de los sensores

Antes de comenzar a programar el proyecto, es conveniente comprobar que todos los sensores funcionan de forma correcta y según lo esperado, de esta forma comprobaremos que no sean defectuosos y estén correctamente conectados, para la realización de estos códigos y los posteriores se ha hecho uso del manual Arduino para obtener documentación sobre la programación de este microcontrolador [7].

Para ello se crearán un conjunto de programas, uno para cada sensor, que darán una respuesta como salida, si esta salida es la esperada significará que el sensor funciona correctamente, en caso contrario, se deberán hacer correcciones.

7.2.1. Comprobación de finales de carrera

Para comprobar el correcto funcionamiento de los finales de carrera se ha efectuado un programa de forma que cuando se detecte que el final de carrera (en caso del ejemplo el conectado al pin número 10 de la placa Arduino) ha sido pulsado se encenderá un led que estará conectado en el pin número 1 de la placa.

Para ello se han inicializado las variables *led* y *FC* correspondientes al número del pin de la placa que ocupan cada uno de los componentes. En el método *setup* se ha declarado que el modo del pin del led es de tipo *OUTPUT* (de salida) y el modo del pin del final de carrera de tipo *INPUT_PULLUP* (de tipo entrada de pulsador) esto ha sido posible gracias al método *pinMode*, posteriormente se ha inicializado el led en *LOW* mediante *digitalWrite*, lo que supondrá que el led al inicio del programa se encontrará apagado.

Durante el método *loop* el cual se encontrará en bucle, como su nombre indica, a lo largo de la ejecución del programa, se registrará el valor digital (0 o 1) del final de carrera en la variable inicializada con el nombre *valor* para posteriormente cargar la variable mediante la función de *digitalWrite* en el pin del led, por lo que si este valor es *HIGH* (el final de carrera esta pulsado) el led recibirá una entrada *HIGH* (5 voltios) y se encenderá, en caso contrario, su entrada será *LOW* (el final de carrera no se encuentra pulsado) y el led no recibirá voltaje por lo que no se encenderá.

```
int led = 1;
int FC = 10;

void setup() {
  pinMode(led,OUTPUT);
  pinMode(FC, INPUT_PULLUP);
  digitalWrite(led,LOW);
}

void loop() {
  int valor = digitalRead(FC);
  digitalWrite(led, valor);
}
```

Ilustración 25 - Código de comprobación de los finales de carrera

7.2.1. Comprobación de los sensores infrarrojos

Una vez comprobados los 4 finales de carrera, se ha procedido a comprobar los 3 sensores infrarrojos instalados en el robot, el IR frontal y los dos inferiores. Para ello se ha hecho uso del mismo programa, modificando los pines de los finales de carrera por los de los sensores IR (en este caso se comprueba el IR situado en el pin número 9) y el tipo de pin del sensor que esta vez pasa a ser de tipo *INPUT*.

```
int led= 12;
int IR = 9;

void setup() {
  pinMode(led,OUTPUT);
  pinMode(IR, INPUT);
  digitalWrite(led,LOW);
}
```

```
void loop() {
  int valor = digitalRead(IR);
  digitalWrite(led, valor);
}
```

Ilustración 26 - Código de comprobación de los sensores infrarrojos

7.2.3. Comprobación de los motores

Para la comprobación de los motores es necesario que las dos comprobaciones anteriores hayan salido correctamente, pues se hará uso de 2 de los 4 finales de carrera instalados y del sensor IR frontal, de esta forma se podrá comprobar si cuando el robot colisiona por su parte derecha corrige su trayectoria hacia la izquierda (retrocediendo un poco previamente) y viceversa. En caso de ser el detector IR frontal el que detectase un objeto, el robot debería retroceder y corregir la trayectoria hacia la izquierda. Los tiempos de corrección serán aleatorios mediante el uso del método *millis()*.

En el código se han declarado variables para los cuatro motores, las cuales contendrán cada una el número de pin en el que esté conectado el motor y la dirección los que hacen referencia, los tres sensores con sus correspondientes pines y cuatro variables extra donde se almacenaran los valores de cada uno de los sensores y el tiempo de corrección en cada movimiento del robot.

En la inicialización del robot al inicio del programa, se han declarado los cuatro motores como pines de modo *OUTPUT* mediante el uso de *pinMode*, los dos finales de carrera como *INPUT_PULLUP* como se había visto anteriormente y el sensor IR como *INPUT*.

Durante el bucle de ejecución la placa Arduino UNO recopilará los valores de cada uno de los sensores y los almacenará en la variable destinada para cada uno de ellos, en caso de que uno de los sensores se encuentre activo porque ha interactuado con el entorno, el robot efectuará una marcha atrás durante un tiempo entre los 200 y los 2000 milisegundos, esto es posible gracias al método *millis()* que devuelve el tiempo total de la ejecución del programa actual, a este tiempo le sumamos 200 milisegundos y los multiplicamos mediante el método *random(10)* por un número comprendido entre el 1 y el 10. Tras obtener el tiempo ejecutamos un método while que indicará que hasta que el tiempo de ejecución del programa sea inferior al tiempo que acabamos de calcular, el robot seguirá ejecutando la marcha atrás. Para la marcha atrás, simplemente ha sido necesario activar los pines de salida de Arduino 6 y 7 que apuntan al motor de la derecha dirección hacia atrás y motor de la izquierda dirección hacia atrás respectivamente, mediante la función *digitalWrite*.

Una vez el robot ya ha ejecutado durante el tiempo esperado la marcha atrás, comprobará si ha sido el final de carrera de la izquierda el que ha sido activado, comprobado el valor de la variable donde se ha guardado previamente la lectura de este sensor, en caso de contener un 1, este sensor ha sido el que ha reportado un objeto por lo que se activaran los pines 8 y 6 de la placa Arduino obligando al robot a efectuar un giro hacia la derecha durante un periodo de tiempo aleatorio como hemos visto anteriormente. En caso contrario, de haber sido el objeto reportado por el sensor derecho o el sensor frontal IR, el robot efectuará un giro a la izquierda, también de tiempo aleatorio.

Si el robot no detecta ningún objeto, este sigue una trayectoria recta en todo momento.

```
int motorderatras = 6; //motor derecha hacia atras pin 5
int motorderadel = 5; //motor derecha hacia delante pin 6
int motorizqdel = 8; //motor izquierda hacia delante pin 7
int motorizqdet = 7; //motor izquierda hacia atras pin 8

int sensorIR = 9; //sensor IR frontal pin 9
int finCizq = 11; //final de carrera izquierdo 10
int finCder = 10; //final de carrera derecho 11

int valueIR;
int valueizq;
int valueder;
long time;

void setup() {
  pinMode(motorderatras, OUTPUT);
  pinMode(motorderadel, OUTPUT);
  pinMode(motorizqdet, OUTPUT);
  pinMode(motorizqdel, OUTPUT);
  pinMode(finCizq, INPUT_PULLUP);
  pinMode(finCder, INPUT_PULLUP);
  pinMode(sensorIR, INPUT);
}

void loop() {
        valueIR = digitalRead(sensorIR);
        valueizq = digitalRead(finCizq);
        valueder = digitalRead(finCder);
        if (valueizq == 1 || valueder == 1 ||valueIR == 0) {
         time = millis() + random(10)*200;
         while (millis() > time){
           digitalWrite(motorderatras, HIGH);
           digitalWrite(motorizqdet, HIGH);
           digitalWrite(motorizqdel, LOW);
           digitalWrite(motorderadel, LOW);
           }

          if (valueizq==1) {
           time = millis() + random(10)*200;
           while(millis() > time){
             digitalWrite(motorizqdet, LOW);
             digitalWrite(motorderadel, LOW);
             digitalWrite(motorizqdel, HIGH);
             digitalWrite(motorderatras, HIGH);
             }
           } else {
             time = millis() + random(10)*200;
             while(millis() > time){
               digitalWrite(motorizqdel, LOW);
               digitalWrite(motorderatras, LOW);
               digitalWrite(motorderadel, HIGH);
               digitalWrite(motorizqdet, HIGH);
             }
           }
```

```
        } else {
         //mientras no detecte objetos hacia delante
         digitalWrite(motorizqdel, HIGH);
         digitalWrite(motorderadel, HIGH);
         digitalWrite(motorderatras, LOW);
         digitalWrite(motorizqdet, LOW);
        }
}
```

Ilustración 27 - Código de comprobación de los motores

7.2.4. Configuración del módulo Bluetooth HC-06

Una vez comprobado el correcto funcionamiento de todos los sensores y los motores, se procederá a configurar el módulo bluetooth HC-06 [8], que será el encargado de comunicar el robot con nuestro APP para Smartphones con sistema Android que crearemos en los siguientes apartados.

Para la configuración del módulo, primero este se conectará en los pines 0 y 1 de la placa Arduino que corresponderán a los pines RX y TX, respectivamente, del módulo HC-06. Estos pines de conexión serán indicados tras incluir la librería SoftwareSerial mediante la línea de código SoftwareSeria BT(0,1). Después, crearemos dos variables, una llamada led que guardará el número del pin donde se encuentra el led en el Arduino, en este caso, el pin número 2 y la variable BTWR que indicará el pin en el que estará conectado la patilla VCC que alimenta el módulo bluetooth HC-06.

A continuación, será necesario indicarle su nombre, su rango de funcionamiento y la contraseña para su emparejamiento con otros dispositivos, para ello se crean las variables nombreBT[10] que contendrá el nombre que adquirirá el módulo, en este caso "HC-06", velocidad que adquirirá el valor del rango de funcionamiento del módulo, en este caso se le pasa el char 4 que corresponde a una velocidad de 9600 bits por segundo, y por último en la variable pin[4] se almacenará la contraseña del módulo para su emparejamiento, la cual será 1234.

En el método *setup()* se inicializarán los dos pines, el correspondiente al módulo y otro correspondiente al led, que será el encargado de darnos una salida visual que indique que el programa se ha ejecutado correctamente. Seguidamente, se iniciará el Led en LOW por lo que se encontrará apagado al inicio del programa y el módulo HC-06 en HIGH por lo que se encontrará encendido al inicio del programa. A continuación, se inician el serial y el bluetooth en el puerto número 9600 para que puedan establecer una comunicación, el serial comenzará la comunicación mandado el comando AT y efectuará una espera de 1 segundo, de esta forma, el módulo entenderá que se trata de un solo comando, ya que, en caso de mandar dos comandos sin un timeout, la placa los entendería como solo 1 y nos devolvería un error. Si el módulo nos devuelve un mensaje con un OK como respuesta, significará que nos está escuchando por lo que podremos proseguir con la configuración (esta respuesta es visible en el monitor serie, que puede abrirse en el apartado herramientas del IDE de Arduino o mediante el atajo *Control+Mayúsculas+M*). Seguidamente se le mandará a través del serial el comando *AT+NAME*, este comando le indica al módulo que la cadena de texto que recibirá a continuación es el nombre que le queremos dar, en este caso, el contenido en la variable nombreBT que contiene el char HC-06. Posteriormente, después del timeout de 1 segundo indicando que vamos a mandar otro comando distinto, le mandaremos el comando *AT+BAUD* que le indicará que a continuación le vamos a facilitar la cantidad de bits por segundos en el que queremos que opere, en este caso 4, que hace referencia a 9600 bits por

segundo, que es la cantidad de bits por default a la que funciona el sistema bluetooth en un smartphone con sistema operativo Android. Esperaremos otro segundo y le indicaremos el pin de acceso al módulo mediante el comando *AT+PIN* seguido de la variable inicializada anteriormente pin, que contiene un *char* con la contraseña 1234. Una vez finalizados los comandos de configuración el serial imprimirá un "Listo" por pantalla para que sepamos que la configuración ha finalizado y encenderá durante 5 segundos el led situado en el pin 2 de la placa Arduino.

Durante el método loop indicamos que el serial de la placa Arduino en caso de estar disponible puede leer del módulo bluetooth y viceversa, por si necesitamos indicar alguna instrucción adicional que en este caso no será necesaria.

```
#include <SoftwareSerial.h>

SoftwareSerial BTPWR(0,1);

const int BTPWR = 6;
const int led = 2;

char nombreBT[10] = "HC-06";
char velocidad = '4'; //9600
char pin [4]= "1234";

void setup() {

   pinMode(BTPWR, OUTPUT);
   pinMode(led, OUTPUT);

   digitalWrite(led, LOW);
   digitalWrite(BTPWR, HIGH);

   Serial.begin(9600);
   BTPWR.begin(9600);

   Serial.print("AT");
   delay(1000);

   Serial.print("AT+NAME");
   Serial.print(nombreBT);
   delay(1000);

   Serial.print("AT+BAUD");
   Serial.print(velocidad);
   delay(1000);

   Serial.print("AT+PIN");
   Serial.print(pin);
   delay(1000);

   Serial.println("Listo");
   digitalWrite(led, HIGH);
   delay(5000);
   digitalWrite(led, LOW);
}
```

```
void loop() {
  if (BTPWR.available())
  Serial.write(BTPWR.read());
  if (Serial.available())
  BTPWR.write(Serial.read());

}
```

Ilustración 28 - Código para la configuración del módulo HC-06

8. Implementación del código Arduino

Este apartado del proyecto se centrará en el análisis y programación del código que se cargará en la placa Arduino UNO con el fin de dotar de funcionalidades al robot, dado que ya tenemos todos los componentes configurados y en funcionamiento.

Dado que el robot va a contener 3 subprogramas en 1, se ha decidido hacer estos subprogramas por separado para después juntarlos en una aplicación que contenga los 3. Estos tres subprogramas son un seguidor de líneas, un subprograma para el control manual del robot mediante la aplicación y un programa de autonomía de forma que dote al robot de comportamiento propio interactuando con su entorno para evitar caídas y choques con los diferentes objetos.

8.1 Subprograma robot autónomo

El primer subprograma que se abordará es el que se encargará de dotar de autonomía al robot, haciéndolo interactuar con su entorno mediante el uso de todos los sensores que lleva integrados. Este programa se encargará de detectar los posibles objetos que rodeen al robot y el suelo, de forma que pueda permitirle evitar obstáculos en su trayecto y caídas, pudiendo corregir su trayectoria al detectar cualquiera de estos obstáculos.

Para ello se ha comenzado declarando variables por cada componente conectado en la placa Arduino del motor, teniendo en cuenta que los dos finales de carrera de cada uno de los lados están conectados en paralelo, es decir, los dos estarán conectados al mismo pin con la ayuda de la placa de conexiones Ariston mediante el modelo eléctrico elaborado al principio de la memoria.

Posteriormente, durante el método *setup()*, donde se inicializará el programa, se declarará el tipo de pin correspondiente a cada uno de los componentes, siendo los motores de tipo *OUTPUT*, los dos pines correspondientes a los finales de carrera, como se ha visto en el apartado anterior, serán de tipo *INPUT_PULLUP* y los sensores infrarrojos de tipo *INPUT*.

Durante el método *loop()*, que será el encargado de ejecutar el programa en un constante bucle, se efectuará una lectura de cada uno de los sensores del robot para poder posteriormente realizar el análisis de cada una de estas respuestas pertenecientes a los sensores infrarrojos y los finales de carrera, los únicos sensores destinados a la interacción con el entorno del robot. Una vez obtenidos estos datos se realizará una comprobación de que estos datos son los esperados para que el robot pueda seguir con su trayectoria rectilínea, en caso de que uno de estos sensores haya detectado un elemento con el que pueda establecer una colisión o pueda provocar una caída, el código entraría dentro de la consulta *if* donde se procederá a corregir la trayectoria para que el robot evite el obstáculo dependiendo del sensor activo y su ubicación.

Dentro de la comprobación *if*, lo primero que se efectuará la activación de los motores marcha atrás durante un tiempo aleatorio, que comprenderá entre los 200 y 800 milisegundos y volverá a crear otro tiempo aleatorio entre 200 y 2000 milisegundos que dependiendo del lado que haya reportado el objeto este girará durante ese tiempo hacia un lado u otro con el fin de evitarlo, en caso del objeto encontrarse a su izquierda, este girará a la derecha y en caso de encontrarse a la derecha, este girará a la izquierda, activando los dos motores en el sentido correspondiente a cada uno de los giros, en caso de haber sido reportado el objeto con el sensor infrarrojos central

únicamente, el robot decidirá aleatoriamente mediante la variable *alea* el sentido de giro para evitarlo, de esta forma se consigue dotarle de más autonomía y poder de decisión.

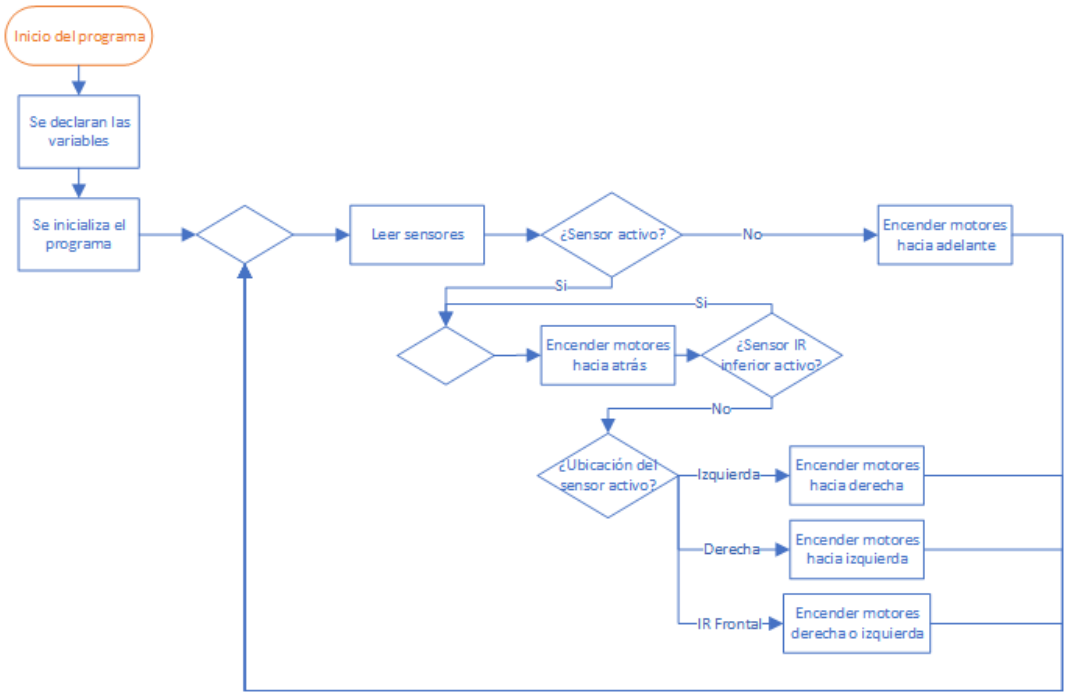


Ilustración 29 - Código arduino del subprograma autónomo

8.2 Subprograma sigue líneas

El segundo subprograma que contendrá el robot será el que le otorgue de la cualidad para seguir una línea, en este caso negra, dibujada en el suelo sin salirse de ella corrigiendo su trayectoria, esta función es muy útil para almacenes o cuando el robot va a seguir las mismas rutas para llegar de un punto A a un punto B.

Para este programa necesitaremos inicializar todas las variables que apunten a cada uno de los pines en los que están conectados en la placa Arduino los componentes y sensores, en un principio se inicializan las variables correspondientes a los dos motores con sus dos posibles direcciones, seguidos por los sensores tanto infrarrojos como los finales de carrera y por último las variables del programa donde recogeremos los valores de cada uno de los sensores y la variable *alea* donde guardaremos un valor generado automáticamente durante la ejecución.

Una vez estas variables apuntan ya al pin de cada uno de los componentes del sistema durante el método setup() se procederá a indicar el modo de cada uno de los pins, en caso de los motores como se ha visto anteriormente serán de tipo *OUTPUT*, en caso de los finales de carrera derechos e izquierdos de tipo *INPUT_PULLUP* y por último los sensores de tipo *INPUT*.

Una vez preparado todo el sistema, entramos en el método *loop()* el método que se ejecutará en constante bucle durante la ejecución del programa, en él, primeramente se recogen los valores de cada uno de los sensores frontales que pueden detectar objetos en el camino del robot (finales de carrera e infrarrojos frontal), en caso de que uno de estos sensores este detectando un objeto en su trayectoria, el robot de detendrá para evitar chocar con él hasta que este desaparezca de su camino, en ese momento, el robot volverá a retomar la marcha siguiendo la línea. Una vez comprobado que no hay objetos con los que pueda establecer una colisión, el robot comprobará

que ninguno de sus sensores infrarrojos inferiores está detectando la línea para poder seguir recto, en caso de que uno de estos sensores detecte la línea negra dibujada en el suelo, el robot corregirá su trayectoria para volver a dejar la línea entre los dos sensores activando los motores en la dirección correspondiente, en caso de detectar la línea con el sensor derecho el robot realizará un giro hacia la derecha y en caso de detectar la línea con el sensor izquierdo el robot realizará un giro hacia la izquierda, durante el tiempo que este sensor siga detectando la línea, para que, una vez haya dejado de detectarla, poder seguir en línea recta.

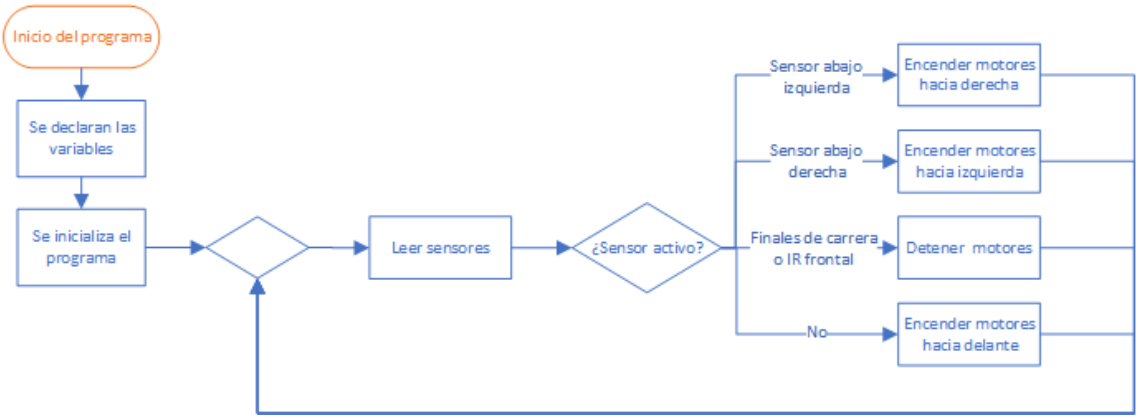


Ilustración 30 - Código arduino del subprograma sigue lineas

8.3 Subprograma manual

Por último, se implementará el programa de control manual del robot por parte del usuario mediante el dispositivo móvil.

Para realizar este subprograma necesitaremos recibir señales bluetooth como novedad respecto a los subprogramas anteriormente comentados. Primero será necesario incluir la librería *SoftwareSerial.h* en el proyecto para poder declarar y hacer uso del componente *HC-06*, luego se declarará el componente de tipo *SoftwareSerial* el cual recibirá el nombre de *miBT* y como argumentos de la conexión pin *RX* y *TX* del módulo *HC-06* en la placa *Arduino UNO*.

Una vez declarado el módulo bluetooth se procederá a la declaración de las variables para cada uno de los motores y sus direcciones como en los programas anteriores y se creará una variable *leer* de tipo *char*, cuyo uso será destinado a almacenar el *char* recibido por parte del módulo bluetooth para, posteriormente, dentro de un *switch* poder interpretar la acción que quiere realizar el usuario.

Una vez declaradas todas las variables se procede a la inicialización del programa, donde primeramente se inicializa el módulo bluetooth a una velocidad (baudios) de 9600 y posteriormente se indica que los pines destinados a cada motor y cada dirección serán de tipo *OUTPUT*.

Durante el método *loop* que se ejecutará de forma repetitiva a lo largo de la ejecución del programa comprobará que el módulo bluetooth se encuentra disponible y una vez el programa compruebe su disponibilidad esperará a leer de este algún dato. Una vez el usuario haya pulsado un botón de la aplicación, el módulo bluetooth retransmitirá a la placa Arduino dicho valor, el cual se filtrará mediante un *switch* y dependiendo del valor, el robot ejecutará una parte del código u otro, siendo el *char B* (*back*) activar los motores para la marcha atrás, *char F* (*forward*) activar

los motores dirección hacia delante, *char L* (*left*) activar motores hacia la derecha, *char R* (*right*) para activar los motores girando a la izquierda y *char S* (stop) para detener los motores.

Una vez realizada la acción volverá a esperar a recibir otro valor a través del módulo bluetooth.

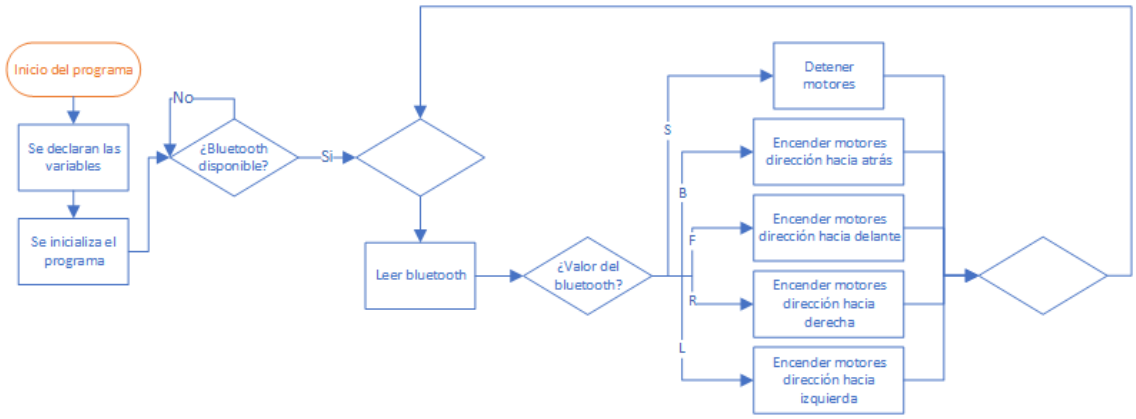


Ilustración 31 - Código arduino del subprograma manual

8.4 Código final

Una vez realizados todos los subprogramas que podrá ejecutar el robot basado en microcontrolador, se ha decidido que, para un uso mucho más práctico, se puedan ejecutar estos tres subprogramas desde un programa general que contenga los tres. De esta forma, no será necesario ir cargando los distintos subprogramas según los vayamos necesitando, sino que desde la app de Android podremos ir cambiándolos según nuestras necesidades, sin tener que reprogramar la placa Arduino.

Para ello se ha creado un nuevo programa que contendrá a los tres subprogramas anteriores, primeramente, se declaran las variables correspondientes a cada uno de los pines en los que se encuentran los distintos componentes del robot y las variables oportunas que necesitará posteriormente el programa. Se inicializarán los distintos pines según su tipo, el cual puede ser, *OUTPUT* en el caso de los motores, *INPUT_PULLUP* en caso de los finales de carrera y de tipo *INPUT* para los sensores infrarrojos instalados en la parte frontal e inferior del chasis.

Una vez inicializados los pins, entramos en el método *loop()* que se encontrará en bucle durante toda la ejecución del programa, en él comprobaremos que el módulo bluetooth se encuentra disponible y en ese caso se esperará a hacer lectura de algún valor que pueda recibir por parte de la aplicación Android y lo guardará en la variable *leer*, dependiendo del valor recibido, mediante un *switch(leer)* se buscará si el valor que ha recibido el módulo bluetooth corresponde con alguna acción del programa. Los valores que se contemplan dentro del *switch* son: *B, F, L, R, S, P* y *H*, los cinco primeros correspondientes al subprograma de control manual explicado anteriormente, donde B hará retroceder al robot, *F* avanzar, *L* girar a la izquierda, *R* girar a la derecha y *S* detenerse, en caso de que la variable leer adquiera el valor *P*, se ejecutará el programa autónomo que recorrerá una superficie sin caer ni cochar con ningún objeto y en caso de recibir el valor *H*, el robot ejecutará el subprograma sigue líneas. Para que estos dos últimos programas dejen de ejecutarse, de forma de que puedan permitir cambiar el subprograma a ejecutar por el robot, se ha creado la variable *leído*, que se encargará en cada iteración del subprograma de recoger si el módulo bluetooth ha recogido el valor *C* proseguido del valor *S* que hará que el programa se detenga y posteriormente se ejecute el *switch* con el valor *S* que detendrá

los motores y el robot quedará a la espera de que el usuario le indique la nueva instrucción a ejecutar.

Dentro del subprograma autónomo el robot efectuará una lectura de los sensores, en caso de no encontrarse activos este proseguirá en línea recta, en caso de haberse detectado alguno activo el robot retrocederá y comprobará en qué lado se sitúa la detección para efectuar cambios en su trayectoria. Una vez corregido el rumbo, el robot volverá a leer el bluetooth por si recibe el valor *C* indicando el fin del subprograma.

Durante el subprograma sigue líneas, el robot realizará una lectura de los sensores, en caso de no encontrarse ninguno activo, proseguirá en línea recta, en caso contrario, comprobará que sensor se encuentra activo, en caso de ser un final de carrera o el infrarrojos frontal, el robot se detendrá esperando a que desaparezca el obstáculo para poder continuar la marcha o se finalice el programa, en caso de ser un sensor infrarrojo de la parte inferior del robot, comprobará si se trata del de la parte izquierda o la derecha y dependiendo de esto hará un giro hacia el lado contrario tratando de dejar la línea en medio de los dos sensores. Una vez realizadas las acciones se volverá a realizar una lectura bluetooth por si recibe el valor *C* indicando el fin del subprograma, volviendo al *switch* inicial.

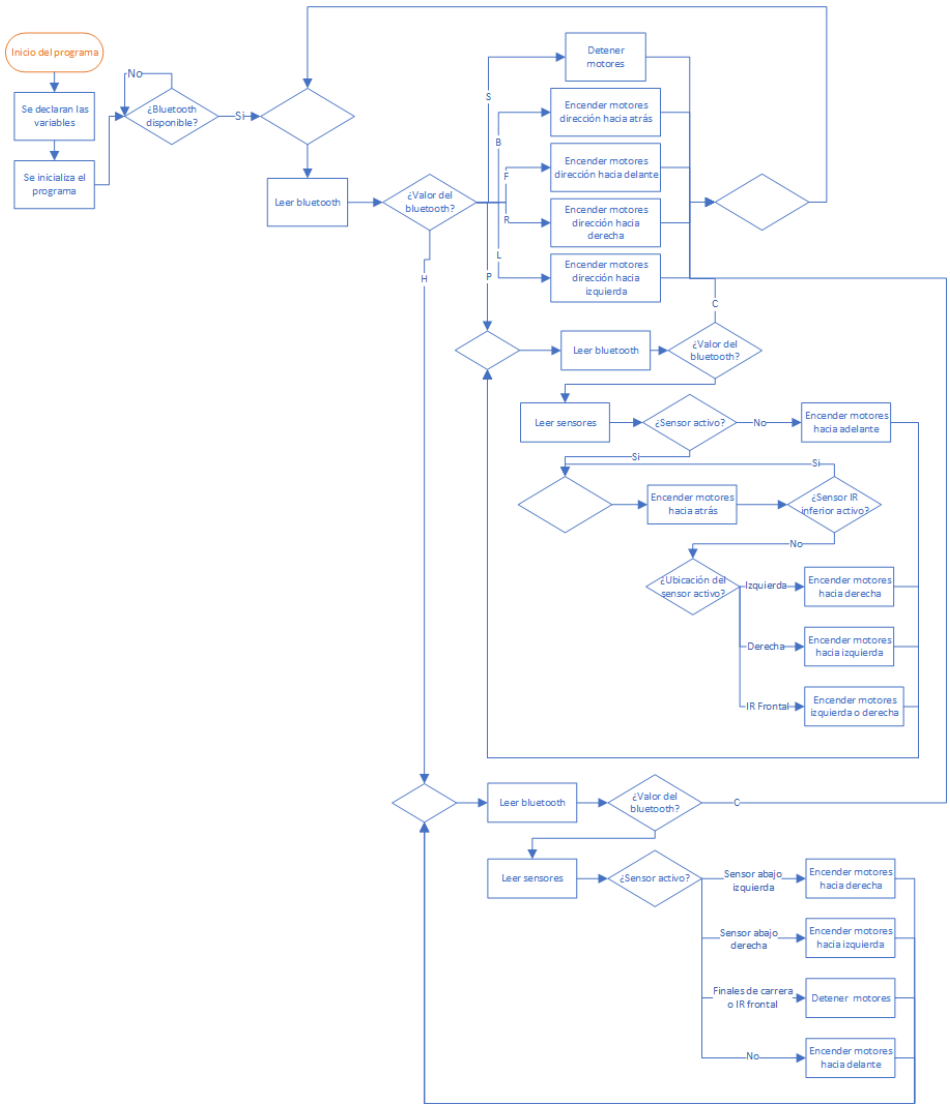


Ilustración 32 - Código final de Arduino Uno

9. Aplicación Android

Para el control del robot se ha decidido realizar una aplicación Android desde la cual, mediante conexión bluetooth, podamos mandar pulsaciones según la tarea que queramos que ejecute el robot.

Para realizar la aplicación se ha hecho uso de Mit App Inventor una plataforma web para realizar aplicaciones Android de forma sencilla y visual mediante la programación por bloques, en ella se ha realizado el siguiente diseño para la aplicación.

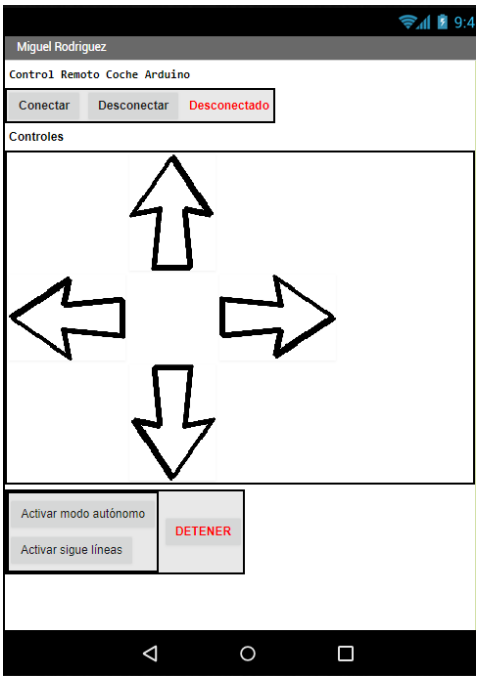


Ilustración 33 - Diseño aplicación android

Cabe destacar que esta es una versión beta pensada únicamente para mostrar que cumple con su funcionalidad, posteriormente la mejora dependerá de las necesidades y requisitos que pida el cliente que adquiera el robot, como de implementar el logo de su empresa, su nombre y hacer las modificaciones que se soliciten, como cambiar de tamaño o color los textos, reorganizar el contenido, etc.

Se ha optado por organizarla según los pasos que realizaremos al abrirla, es decir, primeramente tenemos los botones para establecer la conexión bluetooth con el robot o cortarla, de seguido encontramos las flechas de dirección que nos permitirán usar el robot de forma manual y por último los dos modos autónomos acompañados de un botón detener para parar su ejecución. Dependiendo de que función este activa algunos botones serán deshabilitados, por ejemplo, si se esta haciendo uso del modo autónomo, las flechas de dirección y el botón “Activar sigue líneas” estarán deshabilitados, únicamente se podrá desconectar el bluetooth o detener la ejecución del modo con el botón “detener”.

Para su programación, como se ha comentado anteriormente, se ha efectuado por bloques. Primero se ha programado la conexión bluetooth, indicando que al pulsar el botón "conectar" se listen los nombres y la dirección de los dispositivos bluetooth cercanos y tras haber elegido el dispositivo "HC-06", en este caso, se evalue si la conexión ha sido un éxito, en caso de haber sido emparejado correctamente se mostrará en un *label* llamado "estado" que el dispositivo se encuentra "Conectado" en color verde, en caso contrario se mostrará "Error de conexión" en color rojo.

Para la desconexión del bluetooth se hará una llamada a *Disconect* y se comprobará que la conexión se ha cerrado con éxito (comprobando que el dispositivo ya no se encuentra conectado), en caso de seguir conectado se mostrará en el *label* "estado" la frase "Error de conexión" en rojo, en caso de haber sido desconectado se mostrará "Desconectado" en rojo.

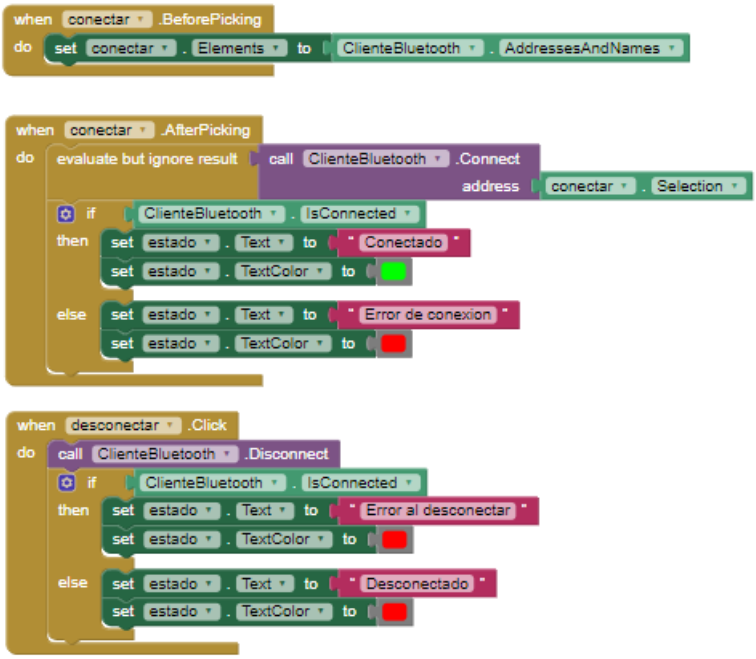


Ilustración 34 - Bloques conexión y desconexión de la App

Para el envío de los *chars* de forma que el robot pueda entender las acciones que quiera recibir el usuario se ha implementado un método para cada vez que sea pulsado cada uno de los botones de la aplicación que se encargaran de llamar al cliente bluetooth mandando el *char* correspondiente a cada acción que se quiera realizar, indicando que en el momento en el que el usuario deje de pulsar dicho botón se envíe el *char S* haciendo detenerse a los motores.

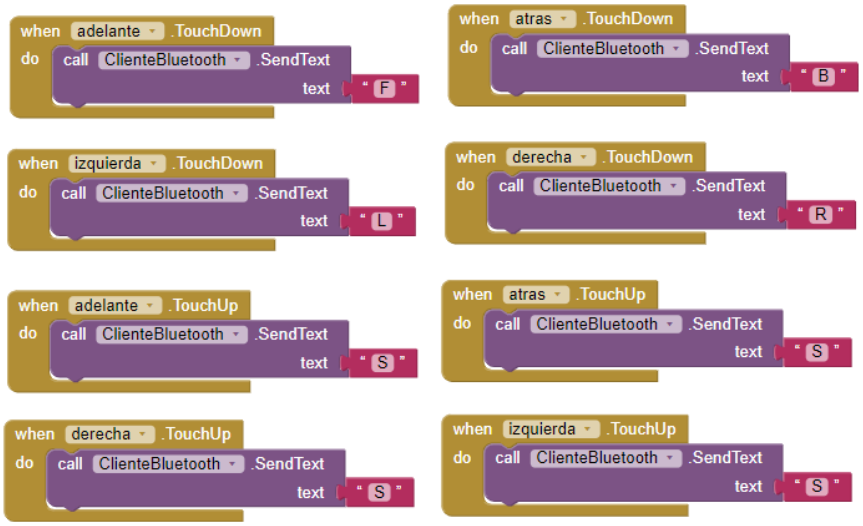


Ilustración 35 - Bloques de control manual de la app

Con los botones para la activación del modo autónomo y el sigue líneas se ha usado la misma lógica, solo que, se ha indicado que algunos de los botones deben deshabilitarse ya que sería incoherente pulsarlos durante la ejecución de estos programas como el reactivar alguno de los modos autónomos o pulsar los botones de dirección del modo manual, además se habilitará un texto que indicará en que modo de ejecución nos encontramos.

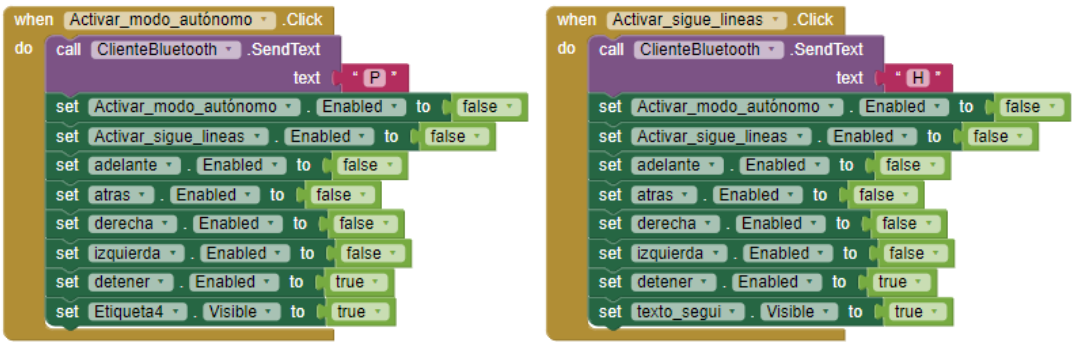


Ilustración 36 - Bloques de los modos autónomos de la App

Para detener los programas autónomos se ha habilitado un botón *detener* que únicamente se encontrará habilitado cuando estemos haciendo uso de uno de estos dos modos, una vez *detener*, sea pulsado, se mandará el *char C* al robot, que según el código Arduino quiere decir salir de esa selección del *switch* y posteriormente el *char S* que en el código Arduino detiene los motores, además se habilitaran de nuevo todos los botones deshabilitados durante la ejecución de estos subprogramas y se quitará el texto de ejecución de cada uno.

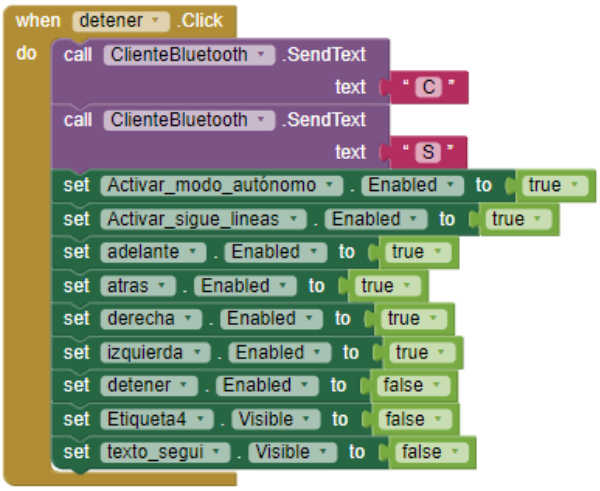


Ilustración 37 - Bloque del botón detener de la app

10. Presupuesto

Para la realización del proyecto se ha necesitado comprar varios componentes, así como la placa Arduino, los materiales y las herramientas para el montaje del chasis y la incorporación de los componentes al mismo. Aunque la finalidad de este TFG no es económica se refleja el presupuesto empleado en la siguiente tabla. [9]

Tabla 5 - Presupuesto del proyecto

Producto	PVP
Placa Arduino	24,29€
Cartón pluma	5,00€
Módulo L298N	5,99€
2x Ruedas y motores	8,29€
Placa Ariston	3,41€
Módulo HC-06	7,99€
4x Finales de carrera	5,64€
3x Sensores infrarrojos	3,00€
4x Pila de 9V	19,80€
Porta-pilas de 6V	5,80€
12x Pilas AA	9,77€
2x Interruptores	2,19€
Rueda unidireccional	2,00€
TOTAL	103,17€

11. Conclusiones y trabajos futuros

Con este trabajo se ha mostrado la infinidad de posibilidades que presenta la robótica y la automatización de las tareas. Hoy en día estas nuevas tecnologías siguen mejorando de cara al futuro abriendo nuevos horizontes y posibilidades con la automatización de nuevas tareas con las herramientas disponibles, por lo que es un mundo con aún mucho recorrido.

El objetivo de crear un robot autónomo controlado con microprocesador el cual pudiese realizar diferentes tareas con el mismo programa y a la vez siendo modular para poder incorporar nuevas funcionalidades en un futuro, se ha conseguido satisfactoriamente. En internet pueden comprarse robots para montar, ya prefabricados y de los cuales se aporta el código en muchas otras webs, sin embargo, en este trabajo se ha optado por realizar uno de cero, con el fin de conocer mejor cada uno de sus componentes, eligiendo su distribución correcta dentro del robot y aprendiendo a programar cada uno de ellos, comprendiendo cómo funcionan.

Para trabajos futuros se puede mencionar la mejora del robot, incorporando nuevos sensores que le permitan realizar nuevas tareas, realizando una aplicación IOS para su control o incluso añadiendo dos nuevas rudas delanteras para que pueda funcionar en mayor ámbito de terrenos.

Las posibilidades que se abren con este proyecto son muchas dado su modularidad y la gran amplitud de usos de la placa microcontroladora Arduino UNO.

12. Referencia al glosario

2D: Dos dimensiones.

APP: Aplicación.

AT: Tipo de comandos que reciben su nombre por la palabra “Atención”.

CPU: Unidad central de procesamiento.

IDE: Entorno de desarrollo integrado.

IR: Infrarrojo, infrared en inglés.

LED: Light-Emitting Diode en inglés.

RX: Recepción.

TX: Transmisión.

USB: Universal Serial Bus.

13. Bibliografía

[1] iRobot Roomba R676: https://tienda.irobot.es/roomba-aspirador-robot-aspirador-roomba-676/R676040.html

[2] Tabla comparativa Arduino - PICAXE: https://www.todopic.com.ar/foros/index.php

[3] Mit App Inventor: http://ai2.appinventor.mit.edu

[4] LibreCAD: https://librecad.org

[5] Fritzing: https://fritzing.org

[6] Arduino IDE: https://www.arduino.cc/en/software

[7] Manual Arduino: https://www.zonamaker.com/descargas/Arduino/Manual-Arduino.pdf

[8] Configuración módulo bluetooth HC-06: http://www.playbyte.es/electronica/arduino/modulo-bluetooth-hc-06/

[9] Compra de los componentes:

- Placa Arduino: https://www.amazon.es/Arduino-UNO-A000066-microcontrolador-ATmega328/dp/B008GRTSV6/

- Ruedas y motores de continua: https://www.amazon.es/XCSOURCE®-engranaje-neumático-Proyectos-TE696/dp/B06XPCZ78W/

- Modulo L298N: https://www.amazon.es/DollaTek-Controlador-módulo-Puente-Arduino/dp/B07DK6Q8F9

- Placa de conexiones Ariston: https://www.ondaradio.es/Catalogo-Detalle/1856/modulos-board-ariston-varios/24744/rohs-modulos-board-blanco-75-x-50

- Módulo bluetooth HC-06: https://www.amazon.es/Neuftech®-Comunicación-Inalámbrica-Bluetooth-transceptor/dp/B00PJXG9NA/

- Finales de carrera: https://www.amazon.es/Micro-Pulsador-Final-carrera-Palanca/dp/B06XRMVTV8/

- Pilas de 9V: https://www.amazon.es/Duracell-Plus-Power-Alcalina-paquete/dp/B00L6ZBJQC/

- Porta pilas de 6V: https://www.amazon.es/Portapilas-Pilas-Caja-Batería-para/dp/B07FDYT296/

- Interruptores: https://www.ebay.es/itm/5x-mini-interruptor-deslizante-ON-OFF-Raspberry-aereo-panel-PIC-tornillo-REF1239/264394688560

Plantilla, guía y recomendaciones para la realización del proyecto: https://www.inf.upv.es/www/etsinf/es/plantilla-tfg/

14. Anexos

Código 1: Subprograma de robot autónomo

Código 2: Subprograma sigue líneas

Código 3: Subprograma manual

Código 4: Programa final

Plano 1: Alzado y planta superior

Plano 2: Vista inferior

Plano 3: Plataforma, alzado y planta

Plano 4: Esquema eléctrico

Plano 5: Esquema Arduino UNO

Código 1 - Subprograma robot autónomo

```
int motorderatras = 6; //motor derecha hacia atras pin 5
int motorderadel = 5; //motor derecha hacia delante pin 6
int motorizqdel = 8; //motor izquierda hacia delante pin 7
int motorizqdet = 7; //motor izquierda hacia atras pin 8

int sensorIR = 9; //sensor IR frontal pin 9
int finCder = 10; //final de carrera derecho 10
int finCizq = 11; //final de carrera izquierdo 11
int sueloder = 12; //sensor IR del suelo parte derecha
int sueloizq = 13; //Sensor IR del suelo parte izquierda

int valueIR;
int valueizq;
int valueder;
int valuesuder;
int valuesuizq;
int alea;
long time;

void setup() {
  pinMode(motorderatras, OUTPUT);
  pinMode(motorderadel, OUTPUT);
  pinMode(motorizqdet, OUTPUT);
  pinMode(motorizqdel, OUTPUT);
  pinMode(led, OUTPUT);
  pinMode(finCizq, INPUT_PULLUP);
  pinMode(finCder, INPUT_PULLUP);
  pinMode(sensorIR, INPUT);
  pinMode(sueloder, INPUT);
  pinMode(sueloizq, INPUT);
}

void loop() {
          valueIR = digitalRead(sensorIR);
          valueizq = digitalRead(finCizq);
          valueder = digitalRead(finCder);
          valuesuder = digitalRead(sueloder);
          valuesuizq = digitalRead(sueloizq);
          if (valueizq == 0 || valueder == 0 ||valueIR == 1 ||valuesuder == 0 || valuesuizq == 0) {
           time = millis() + random(4)*200;
           while (millis() < time){
                digitalWrite(motorizqdet, HIGH);
                digitalWrite(motorderadel, LOW);
                digitalWrite(motorizqdel, LOW);
                digitalWrite(motorderatras, HIGH);
           }

           valuesuder = digitalRead(sueloder);
           valuesuizq = digitalRead(sueloizq);
           while (valuesuder == 0 || valuesuizq == 0){
                digitalWrite(motorizqdet, HIGH);
                digitalWrite(motorderadel, LOW);
                digitalWrite(motorizqdel, LOW);
                digitalWrite(motorderatras, HIGH);
                if (valuesuizq==0){
                  digitalWrite(motorizqdet, HIGH);
                  digitalWrite(motorderadel, HIGH);
                  digitalWrite(motorizqdel, LOW);
                  digitalWrite(motorderatras, LOW);
```

```
                } else {
                  digitalWrite(motorizqdet, LOW);
                  digitalWrite(motorderadel, LOW);
                  digitalWrite(motorizqdel, HIGH);
                  digitalWrite(motorderatras, HIGH);
                }
                valuesuder = digitalRead(sueloder);
                valuesuizq = digitalRead(sueloizq);
              }
              if (valueizq==0 ||valuesuizq == 0 ){
                time = millis() + random(10) * 200;
                while (millis() < time){
                  digitalWrite(motorizqdet, HIGH);
                  digitalWrite(motorderadel, HIGH);
                  digitalWrite(motorizqdel, LOW);
                  digitalWrite(motorderatras, LOW);
                }
              } else if (valueder == 0 || valuesuder == 0 ) {
                time = millis() + random(10) * 200;
                while (millis() < time){
                  digitalWrite(motorizqdet, LOW);
                  digitalWrite(motorderadel, LOW);
                  digitalWrite(motorizqdel, HIGH);
                  digitalWrite(motorderatras, HIGH);
                }
              } else {
                alea = random(2);
                if (alea == 1) {
                 time = millis() + random(10) * 200;
                  while (millis() < time){
                    digitalWrite(motorizqdet, HIGH);
                    digitalWrite(motorderadel, HIGH);
                    digitalWrite(motorizqdel, LOW);
                    digitalWrite(motorderatras, LOW);
                  }
                } else {
                  time = millis() + random(10) * 200;
                  while (millis() < time){
                    digitalWrite(motorizqdet, HIGH);
                    digitalWrite(motorderadel, HIGH);
                    digitalWrite(motorizqdel, LOW);
                    digitalWrite(motorderatras, LOW);
                  }
                }
              }
             } else {
              //mientras no detecte objetos hacia delante
              digitalWrite(motorizqdel, HIGH);
              digitalWrite(motorderadel, HIGH);
              digitalWrite(motorderatras, LOW);
              digitalWrite(motorizqdet, LOW);
            }
}
```

Código 2 - Subprograma sigue líneas

```
int motorderatras = 6; //motor derecha hacia atras pin 6
int motorderadel = 5; //motor derecha hacia delante pin 5
int motorizqdel = 8; //motor izquierda hacia delante pin 8
int motorizqdet = 7; //motor izquierda hacia atras pin 7

int sensorIR = 9; //sensor IR frontal pin 9
int finCder = 10; //final de carrera derecho 10
int finCizq = 11; //final de carrera izquierdo 11
int sueloder = 12; //sensor IR del suelo parte derecha
int sueloizq = 13; //Sensor IR del suelo parte izquierda

int valueIR;
int valueizq;
int valueder;
int valuesuder;
int valuesuizq;
int alea;

void setup() {
  pinMode(motorderatras, OUTPUT);
  pinMode(motorderadel, OUTPUT);
  pinMode(motorizqdet, OUTPUT);
  pinMode(motorizqdel, OUTPUT);
  pinMode(finCizq, INPUT_PULLUP);
  pinMode(finCder, INPUT_PULLUP);
  pinMode(sensorIR, INPUT);
  pinMode(sueloder, INPUT);
  pinMode(sueloizq, INPUT);
}

void loop() {
  valueIR = digitalRead(sensorIR);
  valueizq = digitalRead(finCizq);
  valueder = digitalRead(finCder);
  while (valueizq == 0 || valueder == 0 ||valueIR == 1) {
    digitalWrite(motorizqdet, LOW);
    digitalWrite(motorderadel, LOW);
    digitalWrite(motorizqdel, LOW);
    digitalWrite(motorderatras, LOW);
    valueIR = digitalRead(sensorIR);
    valueizq = digitalRead(finCizq);
    valueder = digitalRead(finCder);
   }
   valuesuder = digitalRead(sueloder);
   while (valuesuder == 0) {
    digitalWrite(motorizqdet, LOW);
    digitalWrite(motorderadel, LOW);
    digitalWrite(motorizqdel, HIGH);
    digitalWrite(motorderatras, HIGH);
    valuesuder = digitalRead(sueloder);
    }
   valuesuizq = digitalRead(sueloizq);
   while (valuesuizq == 0) {
    digitalWrite(motorizqdet, HIGH);
    digitalWrite(motorderadel, HIGH);
    digitalWrite(motorizqdel, LOW);
    digitalWrite(motorderatras, LOW);
    valuesuizq = digitalRead(sueloizq);
    }
```

```
    digitalWrite(motorizqdel, HIGH);
    digitalWrite(motorderadel, HIGH);
    digitalWrite(motorderatras, LOW);
    digitalWrite(motorizqdet, LOW);
}
```

Código 3 - Subprograma manual

```
#include <SoftwareSerial.h>

SoftwareSerial miBT(2,3);
int motorderatras = 6;
int motorderadel = 5;
int motorizqdel = 8;
int motorizqdet = 7;

char leer = 0;

void setup() {

  miBT.begin(9600);


  pinMode(motorderatras, OUTPUT);
  pinMode(motorderadel, OUTPUT);
  pinMode(motorizqdet, OUTPUT);
  pinMode(motorizqdel, OUTPUT);
}

void loop() {

  if (miBT.available()>0){
    leer = miBT.read();
    switch (leer) {
      case 'B':
        digitalWrite(motorderatras, HIGH);
        digitalWrite(motorizqdet, HIGH);
        digitalWrite(motorizqdel, LOW);
        digitalWrite(motorderadel, LOW);
        break;
      case 'F':
        digitalWrite(motorizqdel, HIGH);
        digitalWrite(motorderadel, HIGH);
        digitalWrite(motorderatras, LOW);
        digitalWrite(motorizqdet, LOW);
        break;
      case 'L':
        digitalWrite(motorizqdet, LOW);
        digitalWrite(motorderadel, LOW);
        digitalWrite(motorizqdel, HIGH);
        digitalWrite(motorderatras, HIGH);
        break;
      case 'R':
        digitalWrite(motorizqdel, LOW);
        digitalWrite(motorderatras, LOW);
        digitalWrite(motorderadel, HIGH);
        digitalWrite(motorizqdet, HIGH);
        break;
      case 'S':
        digitalWrite(motorizqdel, LOW);
        digitalWrite(motorderatras, LOW);
        digitalWrite(motorizqdet, LOW);
        digitalWrite(motorderadel, LOW);
        break;
     }
  }
}
```

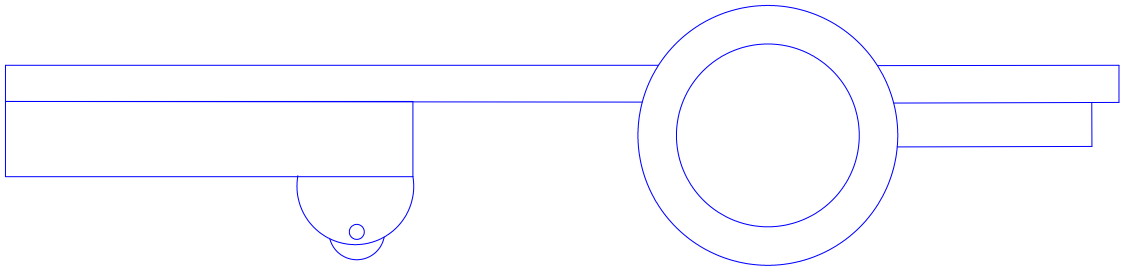
Código 4 - Programa final

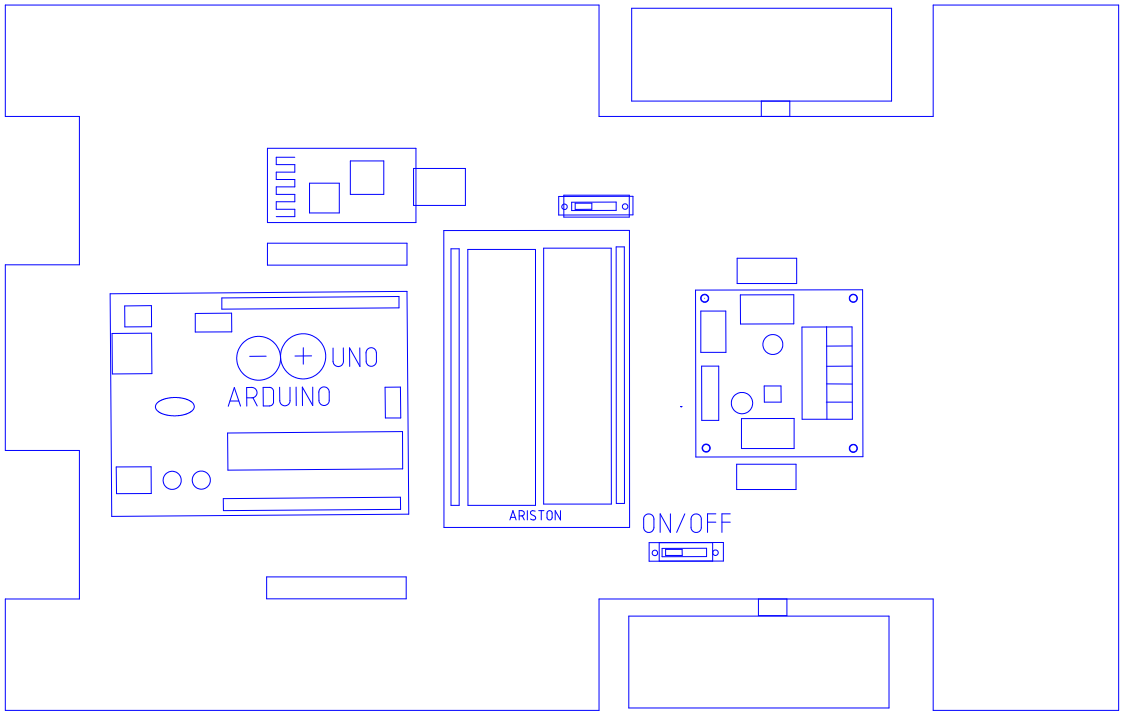
```
#include <SoftwareSerial.h>

SoftwareSerial miBT(2,3);
int motorderatras = 6;
int motorderadel = 5;
int motorizqdel = 8;
int motorizqdet = 7;

const int sensorIR = 9;
const int finCder = 10;
const int finCizq = 11;
int sueloder = 12;
int sueloizq = 13;

int valueIR;
int valueizq;
int valueder;
int valuesuder;
int valuesuizq;
int alea;
long time;

char leer = 0;
char leido = 0;

void setup() {
  Serial.begin(9600);
  miBT.begin(9600);

  pinMode(motorderatras, OUTPUT);
  pinMode(motorderadel, OUTPUT);
  pinMode(motorizqdet, OUTPUT);
  pinMode(motorizqdel, OUTPUT);

  pinMode(finCizq, INPUT_PULLUP);
  pinMode(finCder, INPUT_PULLUP);
  pinMode(sensorIR, INPUT);
  pinMode(sueloder, INPUT);
  pinMode(sueloizq, INPUT);
}

void loop() {
  if(miBT.available()>0){

    leer = miBT.read();

    switch(leer) {

      case 'B':
        digitalWrite(motorderatras, HIGH);
        digitalWrite(motorizqdet, HIGH);
        digitalWrite(motorizqdel, LOW);
        digitalWrite(motorderadel, LOW);
        break;
      case 'F':
        digitalWrite(motorizqdel, HIGH);
        digitalWrite(motorderadel, HIGH);
        digitalWrite(motorderatras, LOW);
        digitalWrite(motorizqdet, LOW);
```

```
    break;

  case 'L':
    digitalWrite(motorizqdet, LOW);
    digitalWrite(motorderadel, LOW);
    digitalWrite(motorizqdel, HIGH);
    digitalWrite(motorderatras, HIGH);
  break;

  case 'R':
    digitalWrite(motorizqdel, LOW);
    digitalWrite(motorderatras, LOW);
    digitalWrite(motorderadel, HIGH);
    digitalWrite(motorizqdet, HIGH);
  break;

  case 'S':
    digitalWrite(motorizqdel, LOW);
    digitalWrite(motorderatras, LOW);
    digitalWrite(motorizqdet, LOW);
    digitalWrite(motorderadel, LOW);
  break;

  case 'P':
    leido = 'P';
    while (leido != 'C') {
     valueIR = digitalRead(sensorIR);
     valueizq = digitalRead(finCizq);
     valueder = digitalRead(finCder);
     valuesuder = digitalRead(sueloder);
     valuesuizq = digitalRead(sueloizq);
     if (valueizq == 0 || valueder == 0 ||valueIR == 1 ||valuesuder == 0 || valuesuizq == 0) {
      time = millis() + random(4)*200;
      while (millis() < time){
           digitalWrite(motorizqdet, HIGH);
           digitalWrite(motorderadel, LOW);
           digitalWrite(motorizqdel, LOW);
           digitalWrite(motorderatras, HIGH);
      }
      valuesuder = digitalRead(sueloder);
      valuesuizq = digitalRead(sueloizq);
      while (valuesuder == 0 || valuesuizq == 0){
           digitalWrite(motorizqdet, HIGH);
           digitalWrite(motorderadel, LOW);
           digitalWrite(motorizqdel, LOW);
           digitalWrite(motorderatras, HIGH);
           if (valuesuizq==0){
             digitalWrite(motorizqdet, HIGH);
             digitalWrite(motorderadel, HIGH);
             digitalWrite(motorizqdel, LOW);
             digitalWrite(motorderatras, LOW);
           } else {
             digitalWrite(motorizqdet, LOW);
             digitalWrite(motorderadel, LOW);
             digitalWrite(motorizqdel, HIGH);
             digitalWrite(motorderatras, HIGH);
           }
           valuesuder = digitalRead(sueloder);
           valuesuizq = digitalRead(sueloizq);
      }
      if (valueizq==0 ||valuesuizq == 0 ){
```

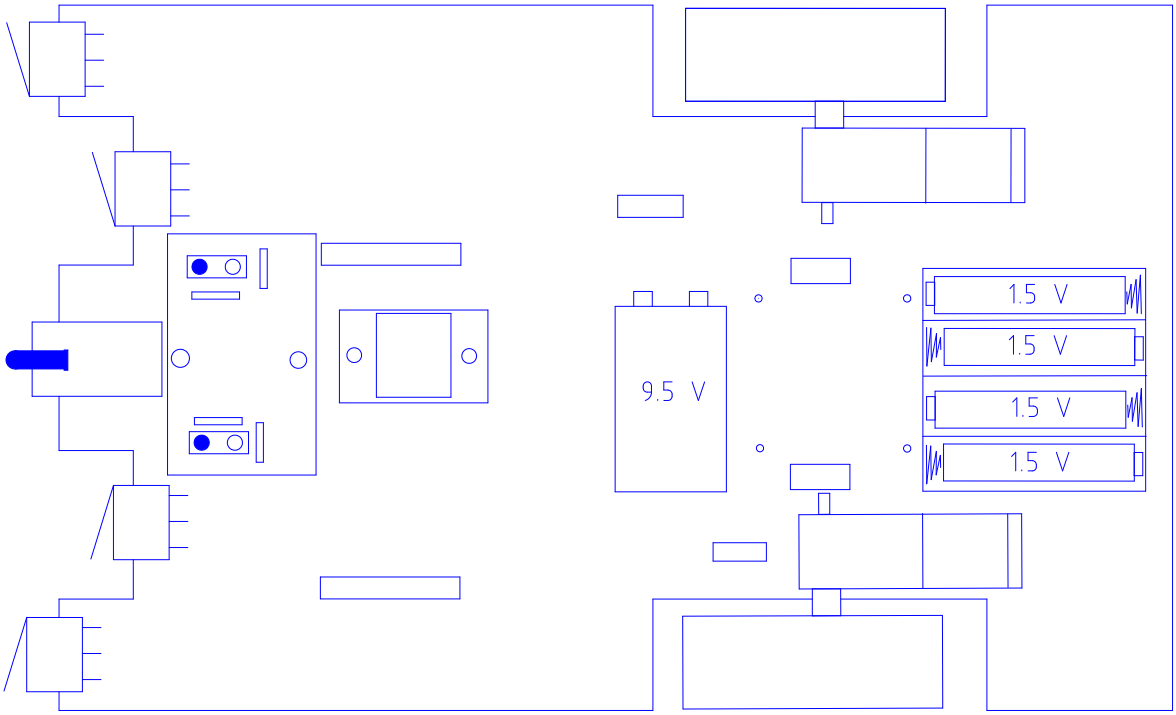
```
            time = millis() + random(10)*200;
            while (millis() < time){
              digitalWrite(motorizqdet, HIGH);
              digitalWrite(motorderadel, HIGH);
              digitalWrite(motorizqdel, LOW);
              digitalWrite(motorderatras, LOW);
            }
          } else if (valueder == 0 || valuesuder == 0 ) {
            time = millis() + random(10)*200;
            while (millis() < time){
              digitalWrite(motorizqdet, LOW);
              digitalWrite(motorderadel, LOW);
              digitalWrite(motorizqdel, HIGH);
              digitalWrite(motorderatras, HIGH);
            }
          } else {
            alea = random(2);
            if (alea == 1) {
             time = millis() + random(10)*200;
              while (millis() < time){
                digitalWrite(motorizqdet, HIGH);
                digitalWrite(motorderadel, HIGH);
                digitalWrite(motorizqdel, LOW);
                digitalWrite(motorderatras, LOW);
              }
            } else {
              time = millis() + random(10)*200;
              while(millis() < time){
                digitalWrite(motorizqdet, HIGH);
                digitalWrite(motorderadel, HIGH);
                digitalWrite(motorizqdel, LOW);
                digitalWrite(motorderatras, LOW);
              }
            }
          }
         } else {
          digitalWrite(motorizqdel, HIGH);
          digitalWrite(motorderadel, HIGH);
          digitalWrite(motorderatras, LOW);
          digitalWrite(motorizqdet, LOW);
        }
        leido = miBT.read();
        }
      break;

      case 'H':
        leido = 'P';
         while (leido != 'C') {
          valueIR = digitalRead(sensorIR);
          valueizq = digitalRead(finCizq);
          valueder = digitalRead(finCder);
          while (valueizq == 0 || valueder == 0 ||valueIR == 1) {
            digitalWrite(motorizqdet, LOW);
            digitalWrite(motorderadel, LOW);
            digitalWrite(motorizqdel, LOW);
            digitalWrite(motorderatras, LOW);
            valueIR = digitalRead(sensorIR);
            valueizq = digitalRead(finCizq);
            valueder = digitalRead(finCder);
          }
          valuesuder = digitalRead(sueloder);
```

```
            while (valuesuder == 0) {
             digitalWrite(motorizqdet, LOW);
             digitalWrite(motorderadel, LOW);
             digitalWrite(motorizqdel, HIGH);
             digitalWrite(motorderatras, HIGH);
             valuesuder = digitalRead(sueloder);
            }
            valuesuizq = digitalRead(sueloizq);
            while (valuesuizq == 0) {
             digitalWrite(motorizqdet, HIGH);
             digitalWrite(motorderadel, HIGH);
             digitalWrite(motorizqdel, LOW);
             digitalWrite(motorderatras, LOW);
             valuesuizq = digitalRead(sueloizq);
            }
            valuesuizq = digitalRead(sueloizq);
            valuesuder = digitalRead(sueloder);
            if (valuesuder == 1 && valuesuizq == 1){
             digitalWrite(motorizqdel, HIGH);
             digitalWrite(motorderadel, HIGH);
             digitalWrite(motorderatras, LOW);
             digitalWrite(motorizqdet, LOW);
            }
            leido = miBT.read();
            }
            break;
          }
    }
  }
```

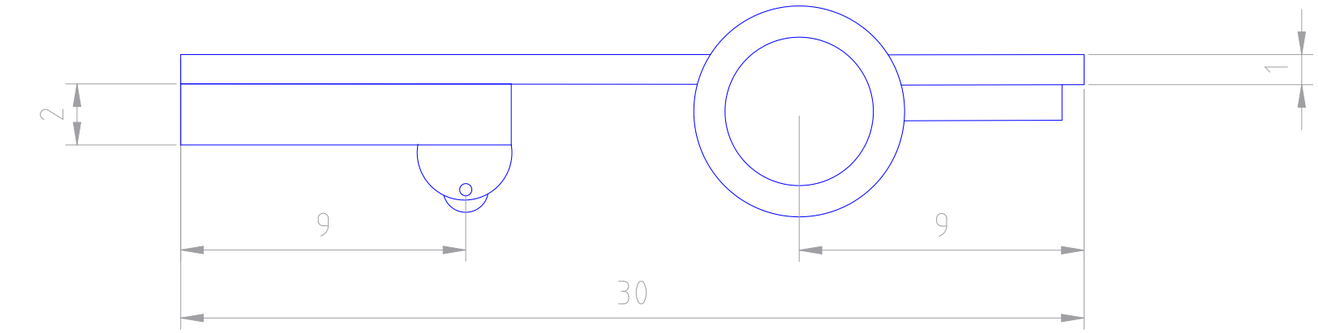


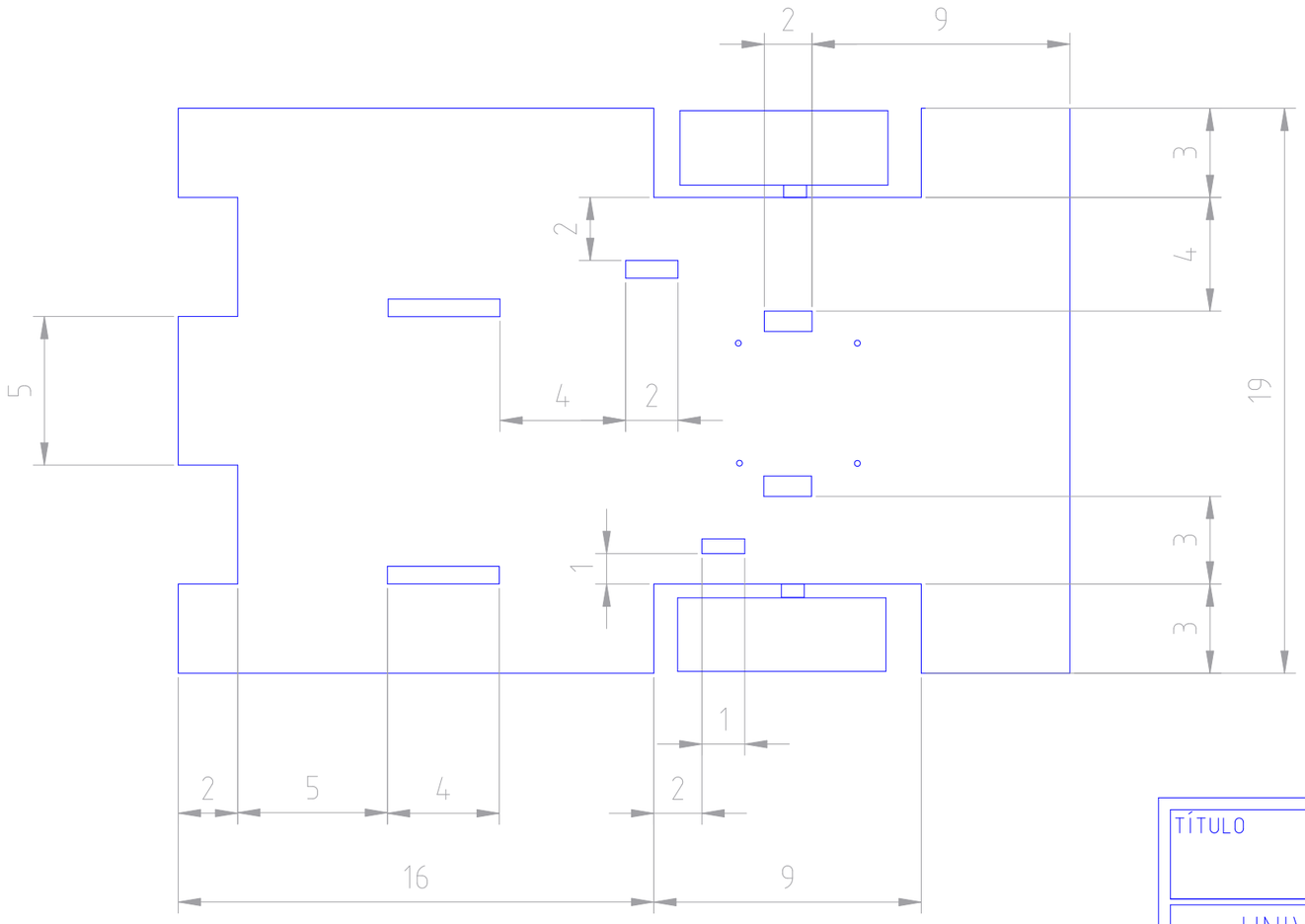


TÍTULO: PROYECTO FIN DE GRADO		PLANO Nº: 01
UNIVERSIDAD POLITÉCNICA DE VALÈNCIA		
AUTOR: MIGUEL RODRÍGUEZ SORNÍ		FIRMA
ESCALA: 1:2	PLANO: ALZADO Y PLANTA SUPERIOR	
CURSO: 2020-21		

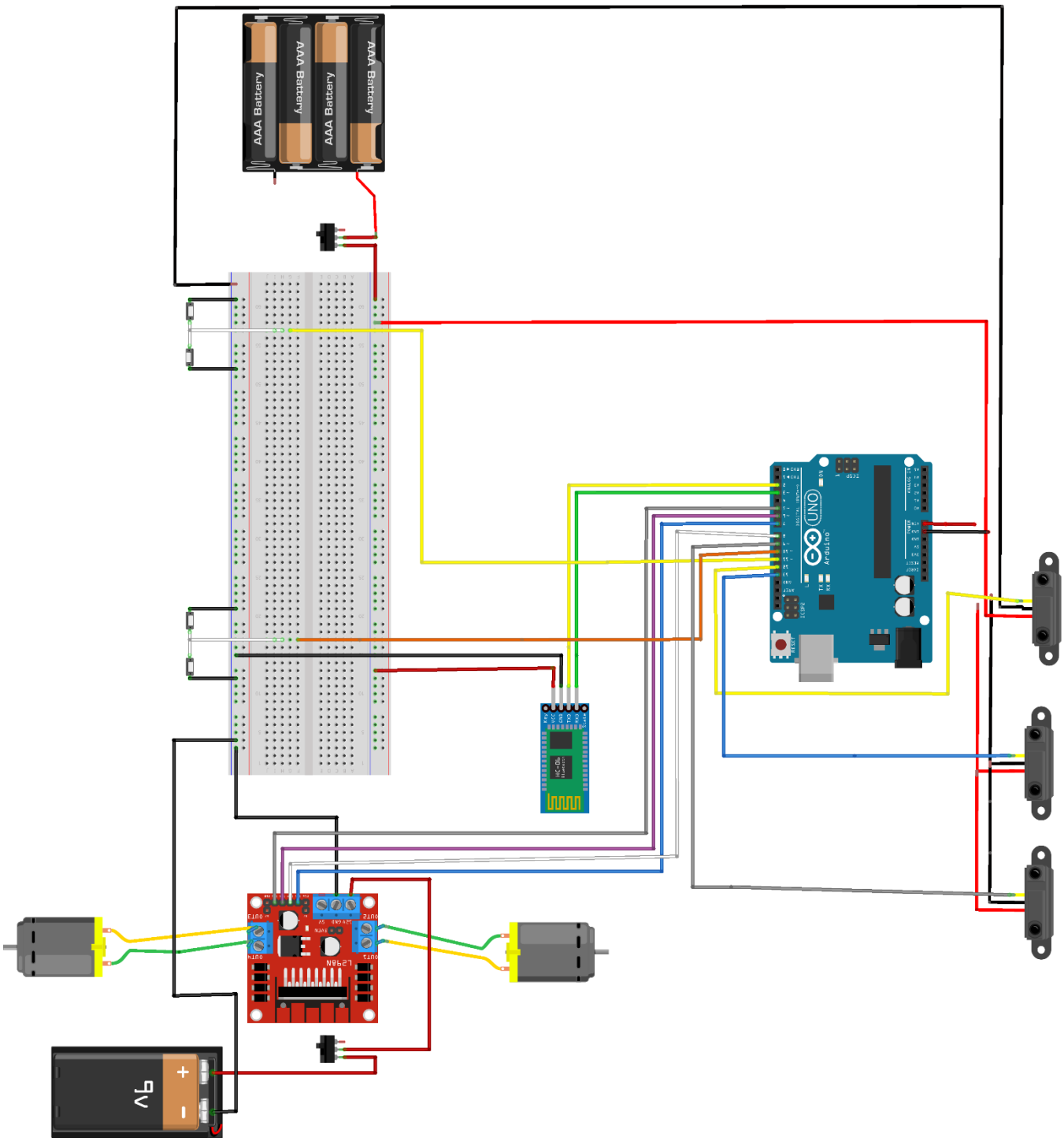


TÍTULO: PROYECTO FIN DE GRADO	PLANO Nº: 02
UNIVERSIDAD POLITÉCNICA DE VALÈNCIA	
AUTOR: MIGUEL RODRÍGUEZ SORNÍ	FIRMA:
ESCALA: 1:2	PLANO: VISTA INFERIOR
CURSO: 2020-21	



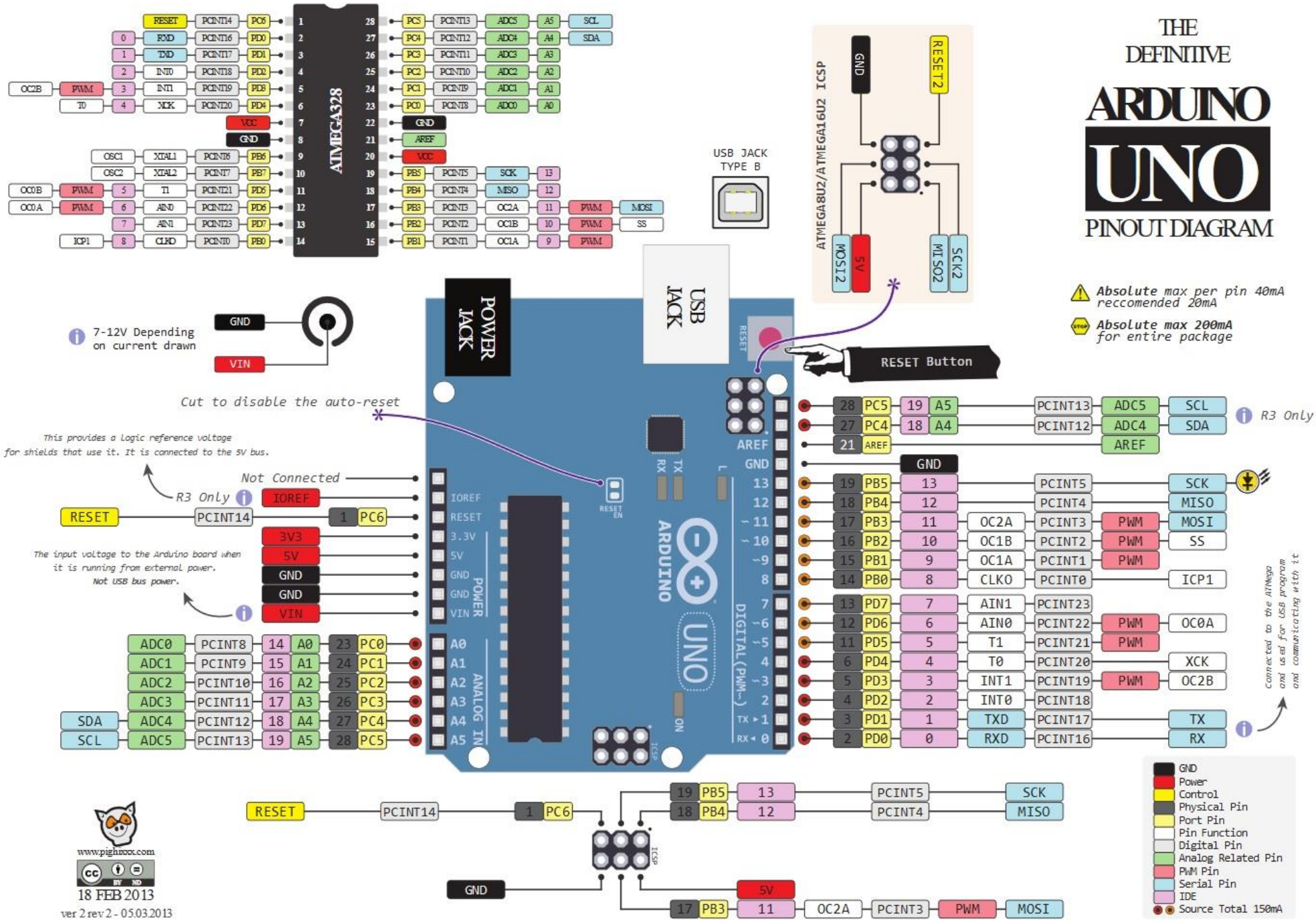


TÍTULO: PROYECTO FIN DE GRADO		PLANO Nº: 03
UNIVERSIDAD POLITÉCNICA DE VALÈNCIA		
AUTOR: MIGUEL RODRÍGUEZ SORNÍ		FIRMA
ESCALA: 1:2	PLANO: PLATAFORMA ALZADO Y PLANTA	
CURSO: 2020-21		



PLANO 4 – Esquema eléctrico

Miguel Rodríguez Sorní


THE DEFINITIVE
ARDUINO UNO
PINOUT DIAGRAM
ATMEGA328
ATMEGA8U2/ATMEGA16U2 ICSP
USB JACK TYPE B
POWER JACK
USB JACK
RESET Button
7-12V Depending on current drawn
Cut to disable the auto-reset
This provides a Logic reference voltage for shields that use it. It is connected to the 5V bus.
Not Connected
R3 Only
The input voltage to the Arduino board when it is running from external power. Not USB bus power.
Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package
Connected to the ATMega and used for USB program
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
Source Total 150mA
www.pighixxx.com
18 FEB 2013
ver 2 rev 2 - 05.03.2013