ArduBot

Alejandro Almira Mollá (aam107@alu.ua.es)
Francisco Manuel García Botella (fmgb3@alu.ua.es)

26 de mayo de 2015

Resumen

Con este proyecto queremos demostrar la potencia del controlador Arduino frente a misiones específicas que nos ha realizado el profesorado de la asignatura. Debido a que nosotros no hemos utilizado los robots de *LEGO* debemos de explicar el funcionamiento básico de los distintos tipos de sensores utilizados en la práctica y las herramientas utilizadas para llevar a cabo este proyecto.

Índice

1. Estructuración del proyecto

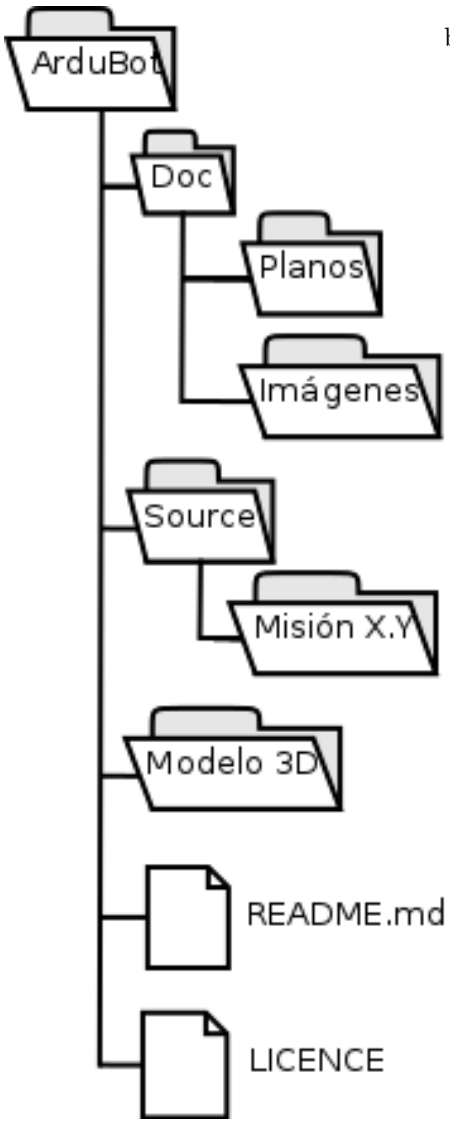


Figura 1: Estructura del proyecto

El árbol de directorios del proyecto se basará en la figura 1.

ArduBot: Directorio padre, correspondiendo con el nombre del proyecto.

- **Doc**: Se encuentra la memoria en formato .pdf y las imágenes y planos correspondientes al proyecto.
  - **Planos**: Para conocer con exactitud las medidas del robot y todos sus componentes.
  - **Imágenes**: Las utilizadas durante la creación de la memoria del proyecto.
- **Source**: El código fuente del proyecto.
  - **Misión X.Y.Z**: Corresponde al código generado en cada misión y su explicación. Donde 'X', 'Y' y 'Z' corresponderá con el número de misión pertinente.
  - **Documentación**: Corresponde al código generado para la creación de la documentación.
  - **Modelos 3D**: Código fuente de los modelos 3D empleados en el proyecto.
- **Modelos 3D**: Archivos .stl generados de los ficheros .scad del apartado Source.
- **README.md**: Archivo de documentación del código.
- **LICENCE**: Los términos de la licencia en la que se ha desarrollado el proyecto.

2. Introducción

El desarrollo de este proyecto ha sido a consecuencia de una práctica de *Tecnología y Arquitectura Robótica* de la *Universidad de Alicante*. Esta práctica consiste en la implementación y experimentación de una serie de misiones que deberemos de cumplir con el robot correspondiente. Estás misiones serán descritas y cumplidas en apartados posteriores.

Para esta práctica, el profesorado nos propuso realizar las misiones con las herramientas de: **LEGO® MINDSTORMS® Education**. Pero surgió la idea de implementarlas en la plataforma **Arduino** y debido a las ventajas y desventajas que suponía implementar para cada una de las plataformas, nos decantamos definitivamente por **Arduino**.

A lo largo de esta documentación veremos viendo los componentes electrónicos, como por ejemplo los sensores y actuadores, utilizados durante el desarrollo de cada misión. Además describiremos las herramientas empleadas para la construcción del robot, los escenarios de las misiones y las mejoras del robot que hemos creído pertinentes.

3. Componentes empleados

3.1. Placa controladora Bq ZUM BT-328

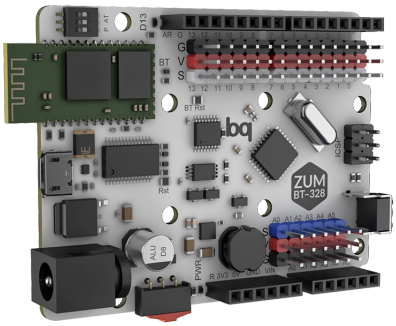


Figura 2: Bq ZUM BT-328

El cerebro de nuestro robot, sin esta placa no habría ningún movimiento ni ningún comportamiento, es por ello que es imprescindible tener una buena placa controladora. En nuestro caso, nos decantamos por la placa fabricada por **bq**, la **bq ZUM BT-328**. Elegimos ésta por el simple hecho de que viene incorporado la comunicación por Bluetooth y para cumplir una de las misiones es idónea. Sin embargo, también nos decantemos por ella por su facilidad de conectar servos sin necesidad de emplear una protoboard. Su funcionamiento es igual que el de cualquier otro Arduino y su comunicación por Bluetooth, la explicaremos más adelante.

Aquí tenemos una tabla comparativa entre el Arduino UNO y nuestro Arduino:

ZUM BT - 328	vs	Arduino UNO
6V - 17V	Voltaje de entrada	6.9 - 20 V
50 mA	Corriente a 3.3V	150 mA
3.2 contínuo	Corriente a 5V	0.8 contínuo - 1 A máx
14	Pines Digitales	14
6	Pines analógicos	6
Sí	Botón de ON/OFF	No
Programable por Bluetooth	Bluetooth	No
Micro-B	USB	Tipo B
Sí	Pines periféricos(GND,V, Señal)	No
8 (4 ArduinoTM compatible, 4 en cuadrado 20x20mm)	Montaje - Huecos	4
Botón de reinicio lateral programable por USB	Otras	Programable por USB

Cuadro 1: Tabla comparativa: UNO vs. bq ZUM BT-328

Su función en el proyecto se basa en la de controlar los valores que los sensores nos proporcionan y actuar en base a esos valores. Es por ello que deberemos de saber interpretar estos valores para un correcto funcionamiento.

Como veremos en el apartado 6, explicaremos detalladamente el circuito empleado. En todas las misiones se utiliza el mismo circuito salvo en la misión que deberemos de establecer un aparcamiento que viene dada con la adición del sensor de ultrasonidos que lo pondremos lateralmente en el robot.

Por último destacar que en los circuitos que aporta Fritzing, que explicaremos más adelante, hemos considerado que el Arduino que más se adapta al nuestro es el siguiente:

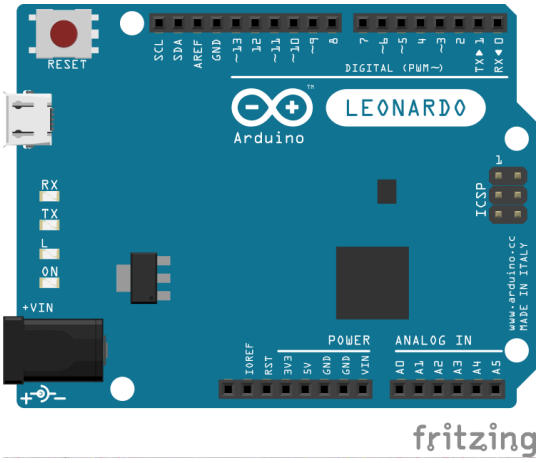


Figura 3: Fritzing: Bq ZUM BT-328

3.2. Servos de Rotación Continua 360º

Mucha gente no tiene conocimiento de este tipo de servos. Los popularmente conocidos son los servos digitales de 180º de rotación, que en este proyecto tenemos uno y lo explicaremos en 3.3. Su función en el proyecto es la utilización de su fuerza y de su rotación continua como la tracción del robot, para que éste pueda recorrer el escenario planificado en la misión. Como podemos ver en sus especificaciones como la siguiente imagen, es un servo de mayor tamaño comparado con los conocidos popularmente, pero es de este tamaño para desarrollar

mucha más fuerza que el que conocemos. Nosotros hemos optado por el Servo SM-S4303R.

Products specification								Technical parameters						
Size (mm)					Weight		Wire	4.8V			6V			Rotation angle
								Speed	Torque		Speed	Torque		
A	B	C	D	E	g	oz	cm	rpm	kg·cm	oz·in	rpm	kg·cm	oz·in	
41.3	20.7	40.2	50.3	10.0	41	1.45	30.0	43	3.3	45.91	54	5.1	70.95	360°

(Specifications are subjected to change without notice.)

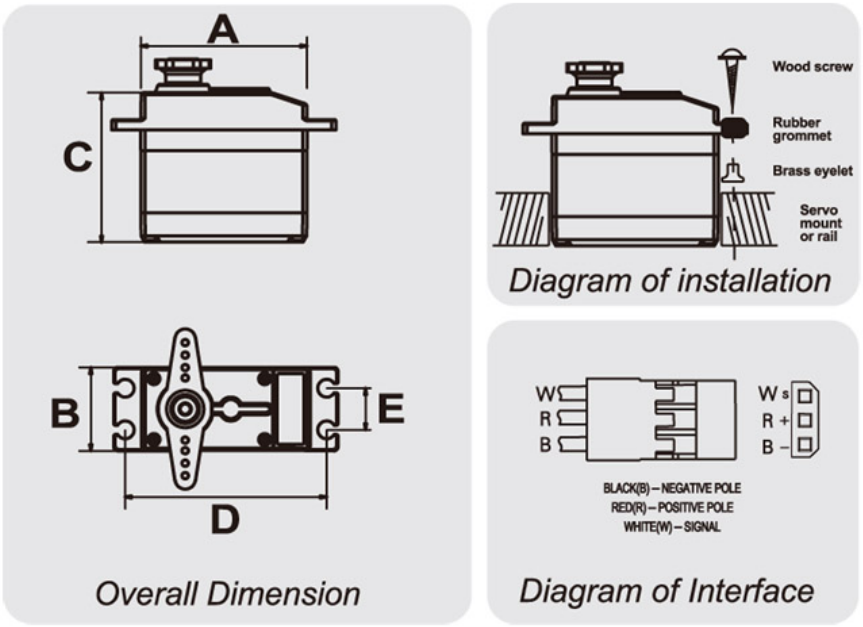


Figura 4: Especificaciones de los Servos utilizados

Su funcionamiento es bastante simple, bajo la librería de Arduino, **Servo.h**, lo emparejamos con el Arduino y para poder mover el servo le enviaremos una señal digital con un ancho de pulso determinado. Normalmente, los servos de rotación continua tienen un rango de pulsos, entre 2000 y 1000 μsegundos. La velocidad y la orientación de rotación se conocerá a través de esos pulsos. Si nosotros le enviamos una señal a través de la función que nos permite la librería, *writeMicroseconds(microseconds)*, su comportamiento será el siguiente:

- Si le enviamos 2000 μs el servo girará en el sentido de las agujas del reloj y a la máxima velocidad que permite el servo.
- Si le enviamos 1000 μs el servo girará a la máxima velocidad también, pero en este caso en el sentido contrario de las agujas del reloj.
- Si le enviamos 1500 μs el servo se detendrá en la posición en la que esté.
- Si le enviamos cualquier otra señal, ésta girará en el sentido de las agujas del reloj si es mayor de 1500 μs y si es menor, girará al contrario. La

velocidad es gradualmente regulable, por lo tanto, si le enviamos la señal $1750\mu s$ girará a la mitad de velocidad y en el sentido de las agujas del reloj.

3.3. Servos de Rotación Digital 180º

Como hemos explicado en la sección anterior, no me detendré mucho aquí. Este servo de rotación digital tiene la función de utilizarse como brazo del robot, permitiéndole golpear objetos con dicho brazo. Como hemos dicho anteriormente, nosotros le pasamos un pulso que es equivalente a un grado en el servo, de esta forma podremos golpear a los objetos que consideremos.

Figura 5: Servo de rotación 180º

3.4. Sensor de reflectancia

Este tipo de sensor detecta la reflectancia que emite la superficie a la que está enfocado. Como ya deberíamos de saber la reflectancia que radian los colores es distinta para cada color. Es el sensor que utilizan los sigue-líneas que hay en el mercado, ya que la función que desemplean éstos es la distinción entre superficies, pudiendo colocar, por ejemplo, una línea negra sobre una superficie blanca para que el sensor verifique si él está sobre la línea negra o sobre una superficie de otro color.

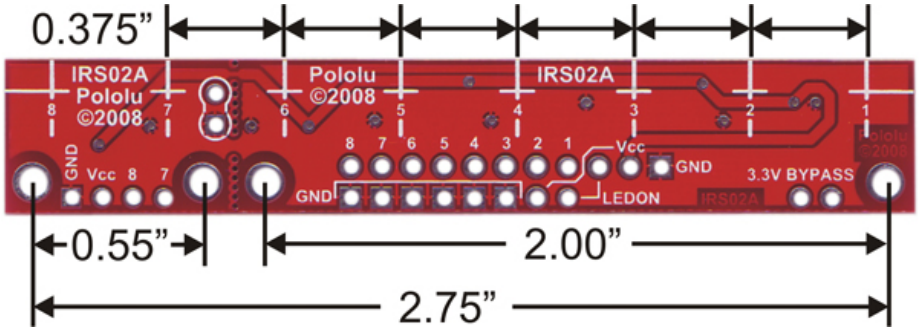


Figura 6: Sensor de reflectancia

Su composición es bastante básica ya que consta de un LED infrarrojo y un fototransistor. Y su funcionamiento es muy básico también, utiliza la reflectancia de los colores para determinar sobre que superficie está situado. Ya que su funcionamiento es emitir una luz con el LED infrarrojo hacia la superficie y esta luz se proyecta en la superficie con una determinada reflectancia hacia el

fototransistor y cuando fototransistor la detecta, a través de la longitud de onda determina si es un color u otro.

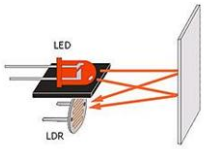


Figura 7: Funcionamiento del sensor de reflectancia

La función que desempeña en el proyecto es la detección de marcas en la superficie por la que se mueve el robot, determinando así si está en un lugar o en otro. Cabe mencionar que nosotros hemos optado por el sensor de Pololu QTR-8RC, el cual nos proporciona una array de 8 sensores de reflectancia digitales, capaz de localizar distintos colores en cada uno de ellos, aunque para las misiones sólo aprovechamos dos sensores, ya que no se necesitan más.

3.5. Sensor de contacto

Sensor de contacto o también llamado final de carrera. Éstos sensores se sitúan al final del recorrido o en nuestro caso en un elemento móvil, con el objetivo de que cuando choque el robot, éste lo detecte y pare los motores para no lastimar ninguna parte del robot. Nosotros hemos optado por un final de carrera de los chinos, funcionan igual o un poco peor que los convencionales, pero cumplen su función y son mucho más baratos.

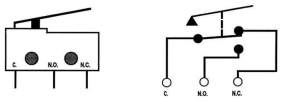


Figura 8: Sensor de contacto

Su funcionamiento es de los más básicos de la electrónica, su estado cambia es presionado o deja de presionar y para ello tiene de un muelle que ejerce una ligera presión sobre la patilla dejando pasar la corriente por ese circuito, pero cuando es presionado, la corriente dejará de pasar por el anterior y pasará por el nuevo camino encontrado. Cuando se encuentre en:

- **Reposo**: Su valor será 0 y se siente a la espera de ser presionado y cambiar la señal a 1.
- **Activo**: Su valor es 1 y dejará de serlo en cuanto se deje de presionar el final de carrera.

Sus ventajas residen en la facilidad de instalación, y que es insensible a estados transitorios, tensiones altas o magnéticas ya que su funcionamiento es totalmente física sin presencia de imanes. Sin embargo, podemos tener problemas con este sensor, puesto que no se caracteriza por su velocidad de detección y posiblemente haya un rebote si realizamos el choque con demasiada fuerza.

En nuestro robot se sitúa en la parte frontal, encima de los sensores de luz provocando que no llegue a colisionar con ningún otro componente del robot,

evitando que por no detectar una colisión perjudique a cualquier componente del robot.

3.6. Ultrasonidos

El sensor de ultrasonidos sirve para calcular la distancia a la que se encuentra un objeto del propio sensor. Este sensor es uno de los más fáciles de conseguir y adaptarlo a nuestro proyecto, puesto que es un sensor muy extendido y existen muchas librerías que nos facilitan el uso de éste.

Figura 9: Sensor ultrasonidos

Para nuestro proyecto, la función del sensor de ultrasonido consistirá en poder obtener la distancia de los objetos que albergan el entorno en el que está situado el robot, y una vez calculados realizar una serie de acciones dependiendo de la misión que queramos realizar. Cabe decir que nosotros hemos utilizado dos sensores de ultrasonidos, uno conectado en el frontal del robot y otro en el lateral. Esto se debe a que la realización de la misión de aparcamiento exigía tener un ultrasonido lateral para determinar cuando queda un hueco libre y poder aparcar ahí, y como nos faltaba espacio, decidimos pegarlo al brazo que contiene el robot.

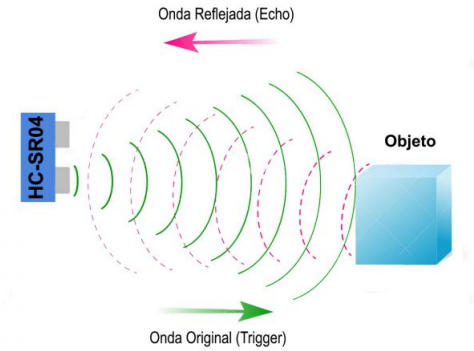


Figura 10: Porque una imagen vale más que mil palabras

El funcionamiento del sensor es muy sencillo como vemos en la figura 10 y es el siguiente:

- Genera una onda sónica(inaudible para el oído humano por su alta frecuencia) en el emisor o "trig"mediante el pulso que le enviemos desde el Arduino.
- Esta onda recorre el espacio hasta encontrarse con cualquier obstáculo y rebotará.
- Cuando la onda vuelve hacia el sensor, el receptor o .ecoregistra la onda que le ha llegado, traduciendo esa onda en un pulso. Por lo tanto, para saber el espacio que ha recorrido la onda simplemente con unas simples fórmulas matemáticas y unos valores determinados podemos resolver el problema.

Las fórmulas simples son las siguientes:

$$Velocidad = \frac{Espacio}{Tiempo} \rightarrow Espacio = Velocidad \cdot Tiempo$$

$$Velocidaddelsonido = 343m/s = 0{,}0343cm/\mu s$$

$$Espacio = 0{,}0343 \cdot Tiempo$$

Pero como la onda ha recorrido el espacio de ida y vuelta, nosotros lo deberemos de dividir entre dos para conocer la distancia real entre el sensor y el objeto, por lo tanto:

$$Espacio = 0{,}01715 \cdot Tiempo$$

3.7. Sensor de luz o fotoresistor

Este sensor lo hemos comentado antes, pero debemos de ponerlo puesto que su uso es algo diferente. Y es que la función de este sensor de luz, está determinado por el uso que deberíamos de haberle dado al micrófono, que puesto que se nos rompió mientras realizábamos pruebas, lo cambiemos por este sensor.

En lugar de realizar palmadas para la misión, deberemos de proyectarle con una linterna, o luz y que cuente como si fuesen palmadas. Cabe decir que este fotoresistor debe ir conectado a un pin analógico del Arduino.

Figura 11: Fotoresistor

3.8. Zumbador

Este componente es prescindible en el proyecto, puesto que nace como un comportamiento opcional, pero que dado su sencillez de implementación y conexión decidimos inyectarlo también.

Figura 12: Zumbador

3.9. Protoboard

La protoboard es una especie de tablero con orificios entre sí que nos facilita la conexión de componentes electrónicos. Su composición está formada por un aislante, normalmente plástico, y por un conductor que conecta los diversos orificios entre si utilizando algún tipo de patrón para la conexión.

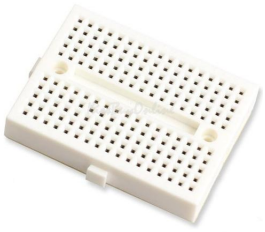


Figura 13: Protoboard

El funcionamiento en este proyecto es la posibilidad de conectar los diversos componentes sin tener que soldar ningún cable, añadiendo más simplicidad al proyecto y una mayor rapidez a la hora de montar y modificar su diseño. Nuestra protoboard está en la parte anterior del robot, aprovechando al máximo el espacio que nos ofrece este robot.

3.10. Porta pilas

Este complemento es el corazón de nuestro robot, el que le da la energía y la expande por todos los circuitos. Consta de 6 pilas del tipo .ªA"que tienen 1.5 V. Éstas están conectadas en serie para dar un voltaje de 9V y poder suministrar de esta forma el Arduino sin que tenga falta de voltaje. Debido a que los muelles son bastante caros y no hay ninguna tienda especializada en eso por aquí cerca, hemos optado por la opción de utilizar como bornes tornillos de 3mm para que realicen el contacto con la parte de la pila correspondiente y realicen el circuito de una forma completa y sin pérdidas.

Como bien ya deberíamos de saber, la corriente que circula por las pilas es corriente continua y por lo tanto no necesitamos ningún adaptador de corriente

para poder hacer funcionar nuestro Arduino sin problemas. La gran desventaja que tiene frente a otros dispositivos es que cuando se agotan las pilas, el renovarlas supone tener que desmontar casi todo el robot. Intentaremos desarrollar otro sistema para no desmontar el robot entero cada vez que se agoten y poder obtener una batería que sea recargable y de menos peso, agilizando así los movimientos del robot y nuestra comodidad.

3.11. Otros componentes

Como queremos detallar al máximo nuestro proyecto, el resto de componentes que forman el robot está formado por:

- **Partes imprimibles**: Partes que hemos impreso con una impresora 3D y que adjuntamos los planos y modelos de todos los objetos impresos.
- **Tornillos**: Como no, el uso de tornillos es fundamental. Para nuestro caso hemos utilizado de dos tamaños de diámetros:

  Servo: Estos tornillos son especiales para el servo, y cuando los compras vienen incluidos.

  3mm: Estos residen alrededor de todo el chasis. Empleamos un total de 18 tornillos de este tipo, repartidos en 12 para presionar las pilas del portapilas y los otros 6 para la sujeción de los servos y de la base del sensor de reflectancia.

  2mm: Utilizamos 14 tornillos. Y su utilización se basa en los componentes electrónicos:

  3: Para el sensor de reflectancia.

  2: Para cada sensor de ultrasonido.

  5: Para sujetar el portapilas y el Arduino.

  2: Para sujetar el final de carrera con el sensor de reflectancia y el sensor de ultrasonido.

- **Tuercas 3mm**: Aquí utilizamos un montón de este componente. Se distribuye en: 6 para la sujeción de los servos y la base del sensor de reflectancia y además el portapilas lleva consigo dos tuercas por cada pila, es por ello que necesitamos 24 para el portapilas, haciendo un total de 30 tuercas de 3mm.
- **Terminales**: Estos posibilitan una mejora a la hora de establecer contacto entre los tornillos que llevarán la carga de las pilas. Una mejora que le hemos introducido es colocarles una tuerca entre el portapilas y el terminal para hacer presión y evitamos que no haya ninguna desconexión fortuita, por el mero hecho de que haya un μs que no haya contacto en uno de los 12 terminales.
- **Junta de fontanería**: Utilizamos 2 juntas de fontanería para evitar el deslizamiento de las ruedas sobre la superficie, y de esta forma que pueda avanzar sin problemas el robot y sin pérdida de pasos. Un detalle a tener en cuenta es el tamaño de la junta, y es que ésta debe ser de un radio menor que la rueda puesto que sino no se mantendría. Si no encontramos una junta de ese tamaño y encontramos una junta bastante más pequeña,

lo que podemos realizar es bañar la junta en agua muy caliente durante un periodo de tiempo para que se dilate un poco y de esa forma poder estirarla mejor y quedar mucho más ajustada a la rueda en particular.

- **Placa aislante**: Esta placa está situada debajo del Arduino para aislarlo del posible contacto entre éste y las pilas. Esta placa consiste en un simple cubo de plástico que proporciona una altura ideal para que no contacten el Arduino y las pilas y evitar una posible catástrofe. Este componente es sólo una medida de seguridad para que no nos dé ningún susto la placa de Arduino.

- **Canica**: La parte más novedosa que añadimos a nuestro modelo, su funcionalidad es la establecerse como una bola loca capaz de girar hacia cualquier dirección sin oponer a penas resistencia y sirve de un tercer apoyo(cada rueda de los servos es un apoyo) para dar estabilidad al robot.

4. Herramientas utilizadas

4.1. Impresora 3D

Una característica de este proyecto es la utilización de técnicas en impresión 3D capaces de abaratar enormemente los costes de un modelo e incluso adaptarlo a nuestras necesidades o ideas según nos vayan surgiendo. Una impresora 3D consta de una máquina con tres grados de libertad en el que tiene un punto extrusor de plástico, llamado Hot-End, y gracias al movimiento de sus motores permite extruir el plástico por capas, cada capa de $0{,}2mm$ aproximadamente y cada capa de una forma determinada, creando así la forma el objeto. Para rellenar el interior del objeto se suele utilizar un tipo de mallado prediseñado que establece la robustez y la flexibilidad en el objeto. La diversidad de materiales está creciendo bastante, puesto que muchos expertos apuntan a que la impresión 3D será el futuro en la comercialización de objetos.

Puesto que el robot está compuesto por un 70 % aproximadamente de componentes no electrónicos y de ese 70 % el 90 % es plástico imprimido con nuestra impresora 3D. Es por ello que consideramos una potente herramienta y siempre recomendamos su utilización.

4.2. Varias herramientas

Como hemos dicho, la impresión 3D nos genera los objetos necesarios para la realización del chasis del robot y es por ello que necesitaremos de ciertas herramientas para su ensamblaje. Entre ellas están las siguientes:

Soldador: Esta herramienta la necesitaremos para estañar cables, soldar otros tantos y también para introducir tuercas entre el modelo, este paso es calentando la tuerca en el hueco hasta que entre. Y es por ello que es necesario tener también estaño, tubo termorectráctil(para que quede más curioso y no haya cortocircuitos) y si tenemos unas pinzas para sujetas los cables mientras los soldamos mucho mejor y más sencillo.

Taladro: Las piezas no son 100 % perfectas y siempre se crean alguna imperfección ya sea de las medidas tomadas para la realización del modelo o

imperfecciones del cálculo de la impresora, y es por ello que deberemos de repasar algún que otro agujero de la pieza que hemos impreso con una broca y si tenemos el taladro, será mucho más rápido. Si se utiliza el taladro para crear un nuevo agujero en alguna de las piezas recomiendo que sean con una broca destinada a la madera, puesto que están más afiladas y permiten una mejor perforación.

Pegamento fuerte: Este utensilio nos ha sacado de más de un apuro a cualquiera de nosotros. Debido a la reacción que provoca con el plástico impreso es fácilmente adhesivo y los resultados son muy buenos. Nosotros lo hemos utilizado para pegar el servo del brazo en la parte anterior del robot y para pegar pegar los dos sensores de Ultrasonido a las respectivas piezas en contacto, el brazo y la pieza que está arriba del final de carrera.

5. Software

5.1. Arduino Programming

El lenguaje Arduino es similar al popularmente conocido C++. Como requisito mínimo el programa deberá de tener dos métodos llamados *setUp()* y *loop()*.

El módulo ***setUp()*** es el encargado de ejecutarse sólo una vez, y esa ejecución viene dada al principio del programa. Por lo tanto, este módulo se utiliza para realizar acciones que con sólo una vez es suficiente, como por ejemplo inicializar variables, sensores, calibrarlos, y ponerlos a punto.

Sin embargo el módulo ***loop()*** es el encargado de estar ejecutándose siempre que el Arduino esté enchufado y mientras no ocurra ningún fallo, seguirá ejecutándose continuamente. En este módulo es donde se realizan las lecturas de los sensores y las acciones de los actuadores.

5.2. Arduino IDE

Este es el software oficial de Arduino, que bajo nuestro punto de vista es un pésimo editor, nosotros programamos con *Emacs* y realizamos las labores de Arduino bajo su IDE. A través de este IDE podremos:

- Compilar los sketches, es decir programas, que hayamos realizado con nuestro editor.
- Quemar el programa al Arduino, pero primero deberemos de seleccionar el tipo de Arduino al que le vamos a quemar el programa, y el puerto en el que se sitúa.
- Este IDE tiene ejemplos de programas simples para la realización de una tarea de ejemplo para los distintos tipos de sensores y de acciones que podemos hacer con nuestro Arduino.
- Si por algún casual tenemos la mala suerte de que se haya roto el *bootloader*, es decir que el programa que realiza la instalación del programa a quemar, nosotros a través del IDE de Arduino podremos repararlo.

- Y por último y no por ello menos importante, podremos incluir librerías fácilmente a través de unos pocos pasos.

Así pues, como hemos dicho anteriormente como IDE de desarrollo, tiene mucho que mejorar, sin embargo sus funcionalidades para realizar con nuestro Arduino, son las necesarias.

5.2.1. Librería Servo.h

Hablaré poquito sobre esta librería ya que tenemos la información completa en La pagina Oficial de Arduino. Lo que más nos interesa son los siguientes métodos:

Servo NOM_VARIABLE: En primer lugar deberemos de incluir un objeto Servo en nuestro programa. A través de este objeto podremos llamar a los métodos de la clase.

attach(PIN,PULSO_MIN,PULSO_MAX): Normalmente, en el *setUp()* vincularemos un servo con un puerto de nuestro Arduino para poder mandarle las señales. Como argumentos opcionales están PULSO_MIN y PULSO_MAX, que determinarán el pulso mínimo y el máximo que podrán alcanzar los servos, normalmente son de 1000 a 2000.

write(ÁNGULO): Este método y el siguiente realizan la misma acción pero con argumentos de entrada distintos. En este nosotros le enviamos el ángulo al que tiene que dirigirse el servo, y automáticamente Arduino realiza la conversión de ese grado a pulso y se lo envía al servo. Normalmente este método se utiliza para los servos de rotación de 180 º.

writeMicroseconds(PULSO): Lo mismo que el anterior pero nosotros le pasaremos como argumento el pulso directamente. El pulso estará dado, normalmente, entre 1000 μs y 2000 μs, ya hemos explicado en el apartado 3.2 las características de cada pulso. Este método suele emplearse para los servos de rotación continua.

5.2.2. Libreria QTRSensors

Esta librería es la determinada por el fabricante de nuestro sensor de reflectancia, véase el apartado 3.4

QTRSensorsRC Nombre_Sensor (ArrayPines [], NumSensors, TiempoEsperaMaximo,LedEmisor): Esta es la declaración de un objeto del sensor de reflectancia. Su constructor está compuesto por la entrada de una array, indicándoles los pines que controlará Arduino, el número de sensores que habrán conectados, el tiempo de espera máximo en esperar la señal y si hemos introducido un puerto que se encargue del emisor de LED.

calibrate(): Este módulo es para calibrar los sensores. En un principio, ejecutamos este módulo en el *setUp()* del programa mientras que el sensor está sobre la superficie en la que va a trabajar para que cuando lo pongamos en marcha realice un mejor contraste.

read(arrayValores): Este módulo se encarga de recibir un array, e introducir los valores que han leído los sensores en ese instante y después compruebas lo que han leído y determinará si está sobre un fondo de un color u de otro.

5.3. Overleaf

Overleaf es un servicio que nos brinda la posibilidad de trabajar colaborativamente para desarrollar nuestros proyectos en LaTeX, pero con la ventaja de que el documento se compilará en tiempo real y permite realizar cambios simultáneamente, al contrario de un repositorio, como habíamos empezado originalmente. Podríamos decir que es como el popularmente conocido Google Docs pero para LaTeX.

Una de las múltiples ventajas es que te evita la instalación de paquetes exclusivos de LaTeXen tu ordenador, ya que esta herramienta contiene todos los paquetes necesarios para compilar cualquier documento escrito en LaTeX. El único inconveniente es la compilación, ya que es una compilación lenta y además cada vez que modificas el fichero, tiene que subirlo al servidor, compilarlo y devolver el PDF resultante, resultando a veces hasta molesto.

Un ejemplo de proyecto en Overleaf lo podemos ver en la figura 14.

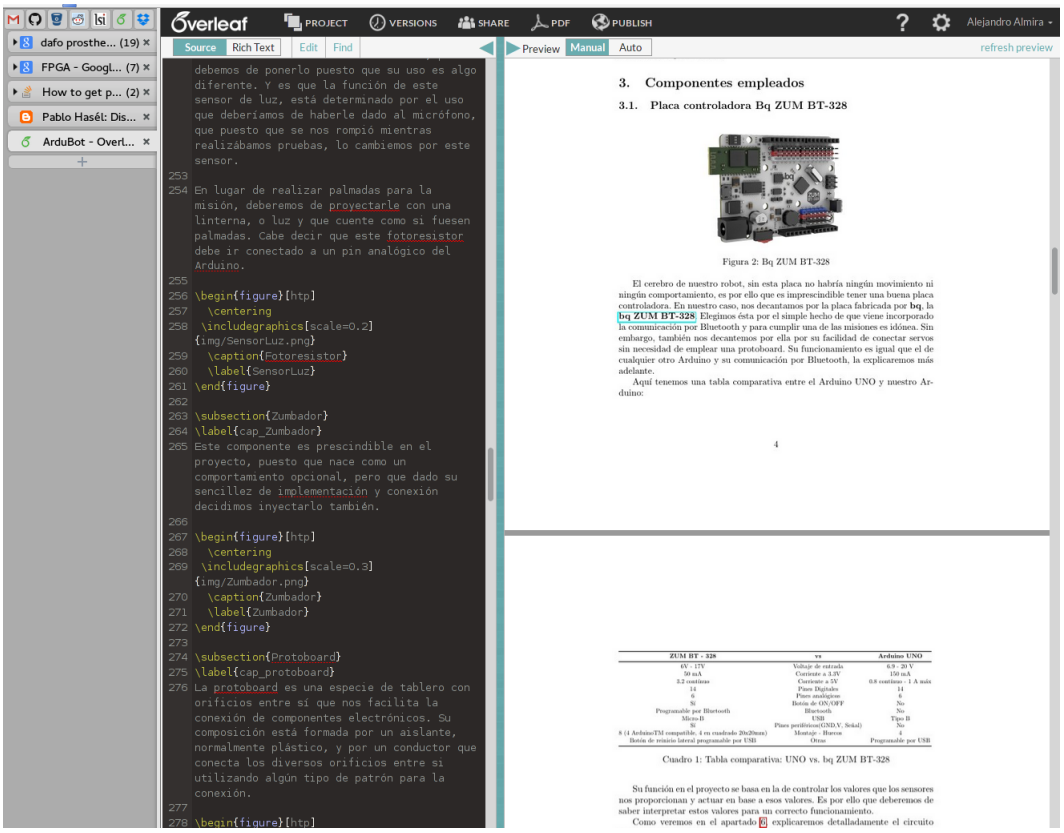


Figura 14: Ejemplo de proyecto de Overleaf

5.4. Fritzing

Este software tiene la funcionalidad de expresar gráficamente los circuitos electrónicos de una forma sencilla y clara. Fritzing puede realizar animaciones con los componentes electrónicos que le pongamos, pero eso es un nivel más avanzado.

Pensamos que es una herramienta muy potente puesto que incluye muchos componentes electrónicos prediseñados por defecto, y si no te sirve ninguno, puedes buscarlo a través de una sencilla búsqueda en internet o si no lo encuentras, puedes intentar realizarlo tú mismo.

La funcionalidad que le hemos dado es la de poder observar a simple vista el circuito que hemos integrado en este proyecto, ya que si no existieran herramientas de este tipo deberíamos de expresarlo con el esquema simbólico, el cual es más difícil de entender y de aclararse tanto para el lector como para el diseñador del circuito. Nosotros hemos realizado el esquema de nuestro circuito a través de este software, como puedes comprobar en la figura 24 . Y en la figura 15 podemos ver el programa con un proyecto cargado.

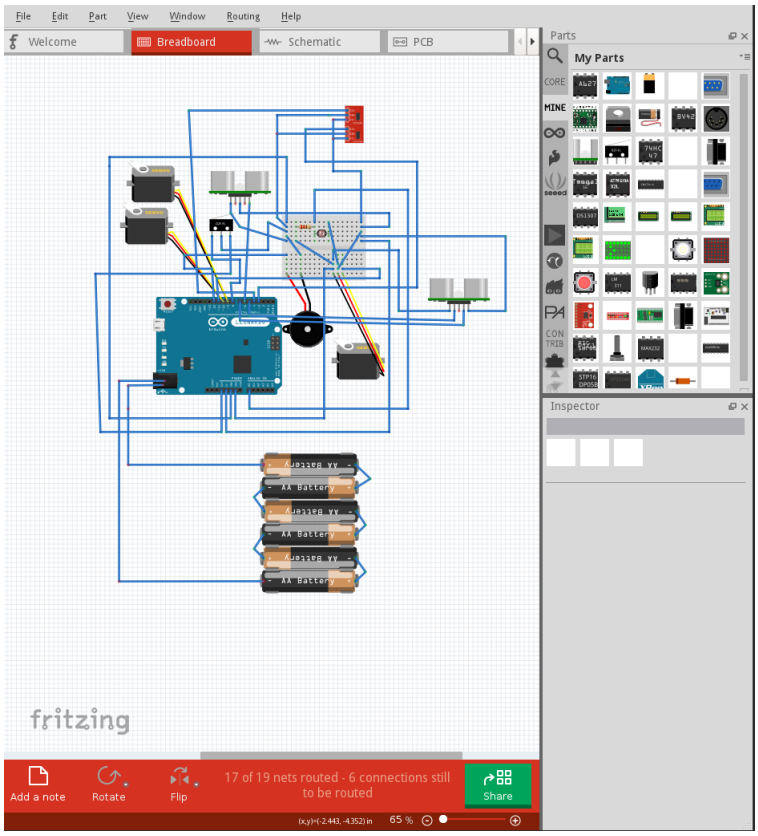


Figura 15: Ejemplo de proyecto de Fritzing

5.5. OpenSCAD

OpenSCAD es una aplicación de software libre, la cual sirve para crear objetos sólidos 3D CAD.

OpenSCAD no es como un programa interactivo para modelar, como por ejemplo, Blender, sino más bien un compilador 3D basado en un lenguaje de descripción textual, es decir, nosotros creamos un código fuente, como si fuese C++, pero digamos que tenemos una librería con la que poder crear cubos (**cube**), cilindros (**cylinder**), etc... OpenSCAD se basa en operaciones de conjuntos, es decir, uniones y diferencias.

Un ejemplo de creación de un cubo lo podemos ver en la figura 16.

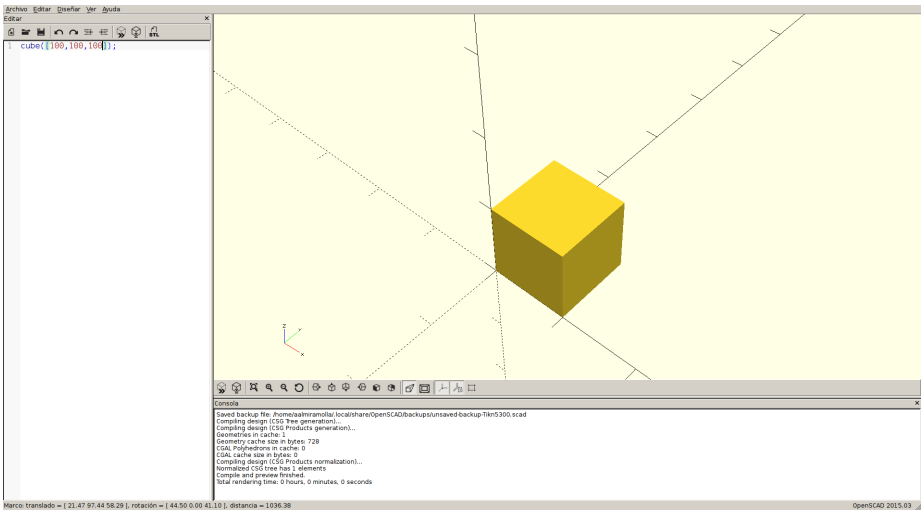


Figura 16: OpenSCAD

6. Montaje

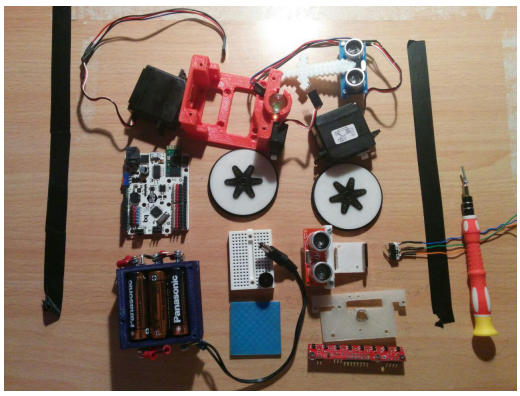


Figura 17: Todas las piezas del robot

Debido al número de piezas que el robot tiene como podemos comprobar en la figura 17 y su complejidad media que puede ocasionar su montaje, hemos decidido realizar un pequeño tutorial, el cual contiene los siguientes puntos:

1. **Las piezas para el chasis**: Las piezas para el chasis son la de la figura 18

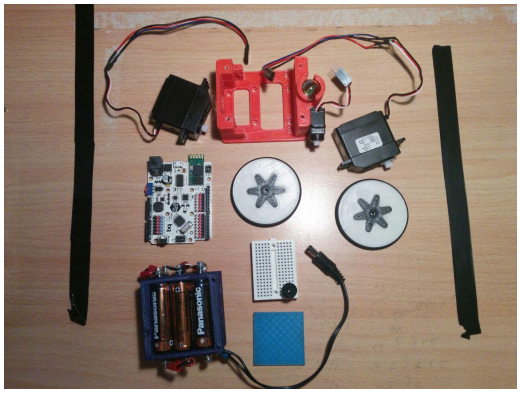


Figura 18: Las partes que necesitamos para el chasis principal

2. **Montaje del portapilas**: Como hemos dicho en el apartado 3.10, el primer paso para el montaje es el montaje del chasis principal con el portapilas, por cierto, no nos olvidemos de poner correctamente las pilas. Y pegamos directamente el servo en la parte anterior. 19
3. **Montaje Servos**: Una vez puesto el portapilas con las pilas(muy importante), vamos al montaje de los servos, muy sencillito, antes deberemos de introducir con el soldador las tuercas pertinentes a los anclajes de los servos para que los tornillos puedan hacer rosca.20
4. **Montaje Ruedas**: Este es el paso más sencillo, con los tornillos que incluyen los servos los introducimos en su acople unido a la rueda y lo enroscamos. Y colocamos la placa aislante encima de las pilas.21

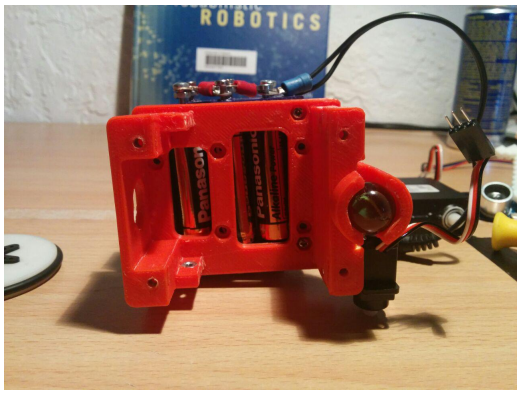


Figura 19: Montaje del portapilas

Figura 20: Montaje de los Servos

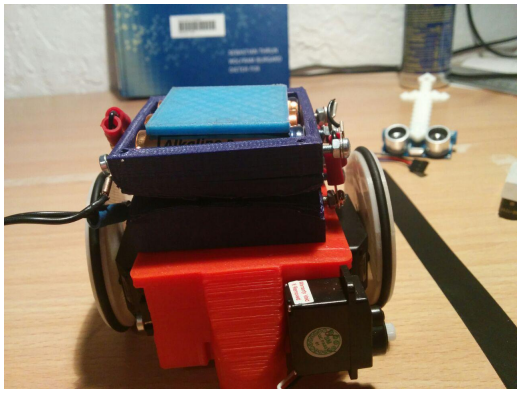


Figura 21: Montaje de las ruedas

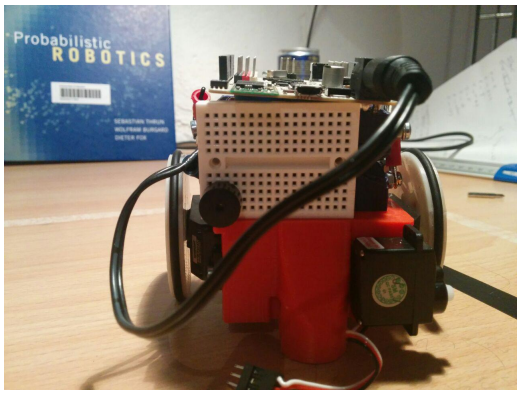


Figura 22: Montaje del Arduino

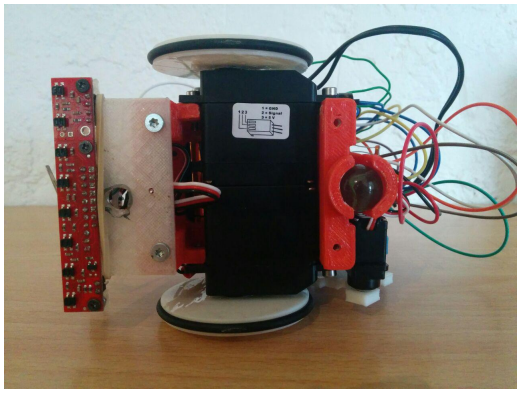


Figura 23: Montaje de los sensores frontales

5. **Montaje del Arduino y de la protoboard**: Colocamos el Arduino encima de la placa aislante y con tornillos de 2mm enroscamos en el portapilas.22

6. **Montaje de los sensores frontales**: Este el último paso para concluir el montaje del chasis. Simplemente debemos de colocar dos tornillos de 3mm en la parte inferior del robot con la plancha que contiene los sensores.

Es con ello, el montaje de las partes imprimibles viene dado por una serie de pasos que el lector puede interpretar debidamente gracias a las fotografías tomadas aclarando cualquier posible duda sobre el montaje.

Para la realización del montaje hemos considerado el circuito de la siguiente forma:

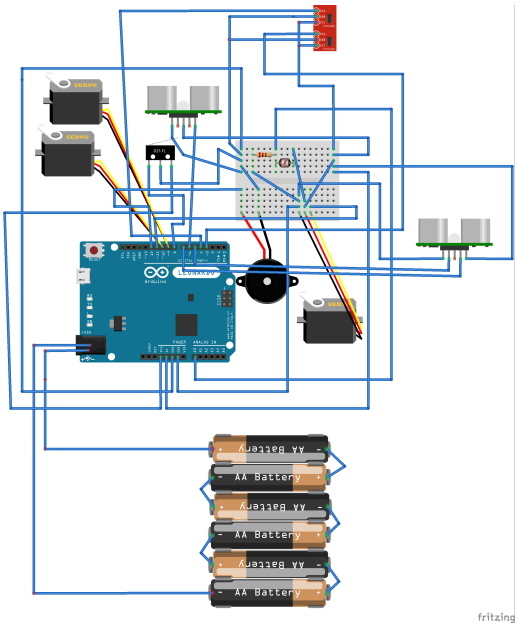


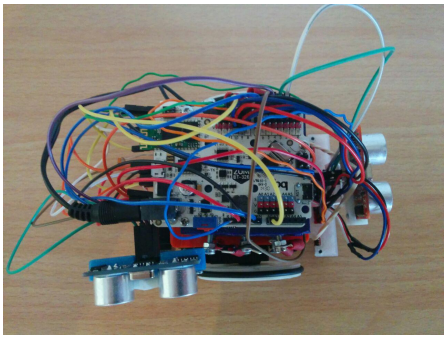
Figura 24: Circuito

.

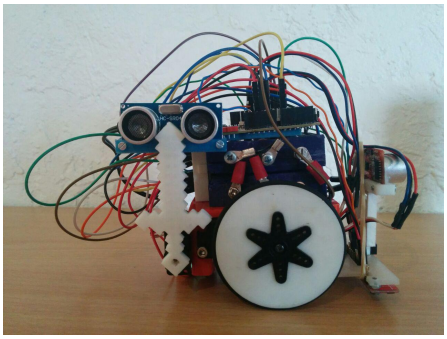
Y con esto terminamos el montaje del robot.

7. ArduBot - Versión Final

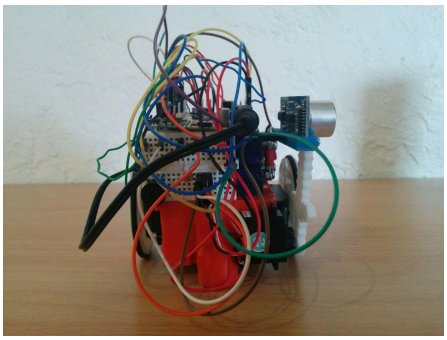
En esta sección mostramos la versión final de nuestro robot. Esta versión tendrá que hacerse cargo de cumplir las misiones que comentaremos posteriormente.



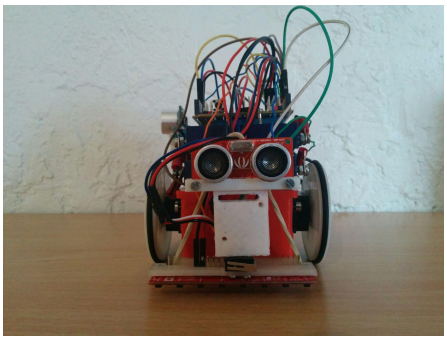
(a) Vista desde arriba



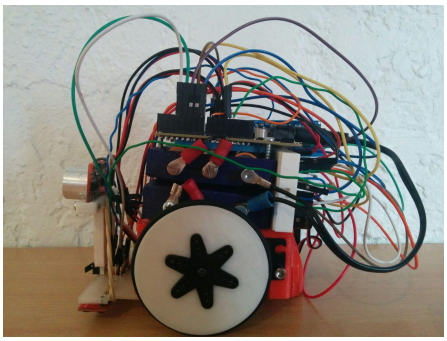
(b) Vista desde el lateral derecho



(c) Vista desde atrás



(d) Vista desde el frontal



(e) Vista desde el lateral izquierdo

Figura 25: Vistas del robot definitivo

8. Misiones realizadas

8.1. Misión 1.1

8.1.1. Enunciado

La primera misión de nuestro robot va a ser moverse siguiendo una trayectoria previamente establecida, por ejemplo, dentro de una fábrica. Esta ruta es la siguiente (figura 26):

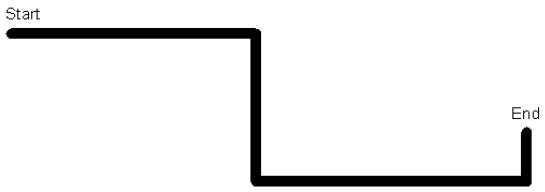


Figura 26: Ruta de la misión 1.1

El algoritmo a seguir, en pseudocódigo, es:

- Encender Motor B y Motor C, sentido hacia adelante
- Esperar 3 segundos
- Girar a la derecha
- Moverse hacia adelante durante 2 segundos
- Girar a la izquierda
- Moverse hacia adelante durante 3 segundos
- Girar a la izquierda
- Moverse hacia adelante durante 1 segundo
- Parar Motores B y C
- (Opcional) Emitir un sonido como finalización

Opcional: ¿Serías capaz de hacer que tu robot anduviera marcha atrás desde la posición final del recorrido hasta la inicial? ¿Y si gira 180o desde la posición final y anduviera de nuevo marcha adelante?

8.1.2. Como se ha realizado

Para la realización de esta misión hemos utilizado la librería **Servo.h**, especificada en la sección 5.2.1.

Hemos seguido el pseudocódigo al pie de la letra, hemos enchufado los motores en sentido hacia delante enviando el pulso correspondiente, hemos esperado 3 segundos con una función de Arduino, **delay(time)**, y así con todos los pasos, enviando los pulsos correspondientes y esperando el tiempo necesario.

8.1.3. Escenario Especifico

Para esta misión, en vez de ponerle un bolígrafo al robot, hemos puesto con cinta aislante el recorrido de la figura 26 para comprobar que el robot realizaba bien el recorrido.

8.2. Misión 1.2

8.2.1. Enunciado

Programa ahora una función en la que el robot dibuje un círculo, de tal manera que el radio del círculo (en cm.) sea un parámetro de entrada a determinar por el usuario.

8.2.2. Como se ha realizado

Nosotros hemos realizado la práctica basándonos en la imagen 27 en la cual nosotros le ponemos la máxima velocidad a la rueda derecha y la velocidad relativa será la de la rueda izquierda para poder realizar el círculo correctamente. Además la forma de pasarle el radio del círculo es a través del puerto serial, que si lo tenemos conectado por Bluetooth podremos enviarle el radio que queremos. Como una imagen vale más que mil palabras, hemos decidido generar la siguiente imagen:

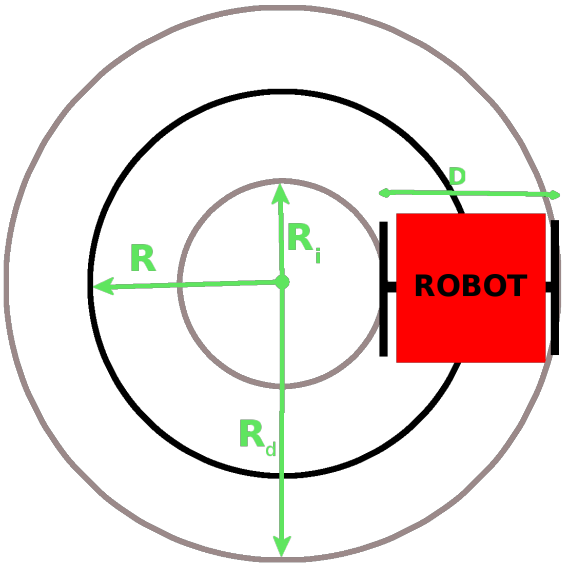


Figura 27: Realización de la misión

Para realizar la fórmula, deberemos de tener en cuenta los símbolos de la imagen 27. Tenemos una serie de pequeñas fórmulas:

$$R_i = R - D/2$$

$$R_d = R + D/2$$

$$Rel = R_d / R_i$$

Como resultado, nosotros tendremos la máxima velocidad en R_d, que es 2000, pero para hallar la velocidad de R_i deberemos de calcularla a través de:

$$Vel_i = R_d * Rel$$

8.3. Misión 1.3

8.3.1. Enunciado

A continuación, imagina que quieres controlar al robot mediante telepresencia. Para ello, programa una función en la que se realice este control de la siguiente forma:

- El movimiento hacia adelante, atrás, izquierda y derecha vendrá determinado por las teclas de los cursores.
- Si se aprieta la tecla 'espacio' y el robot está en movimiento, debe detenerse. Si el robot está parado, debe empezar a moverse hacia adelante.
- Las teclas '+' y '-' se usarán para aumentar y reducir la velocidad de los motores, respectivamente.
- La tecla 'u' y 'd' permitirán mover el brazo del robot hacia arriba o hacia abajo, respectivamente.

8.3.2. Como se ha realizado

Para realizar esta misión hemos programado una función para cada comando, es decir, ir hacia delante, hacia atrás, aumentar/disminuir velocidad, etc... y dependiendo de la tecla pulsada, llamamos a una función o a otra.

8.4. Misión 2.1

8.4.1. Enunciado

Realiza un programa en el que el robot se mueva indefinidamente hacia adelante hasta que detecte un choque.

8.4.2. Como se ha realizado

Para la realización de esta misión teníamos que leer el valor del sensor de choque, para ello hacemos uso de una función de Arduino, **digitalRead(pinChoque)**, la cual nos da el valor 0 o 1, como se específica en la sección 3.5, y cuando detecta que ha chocado paramos los motores.

8.4.3. Escenario Especifico

Hemos colocado un obstáculo para que el robot se choque.

8.5. Misión 2.2

8.5.1. Enunciado

Crea ahora un programa en el que el robot comience a moverse al pulsar el sensor de contacto y efectúe un giro según el número de colisiones continuadas detectadas en el sensor:

- 1 pulsación: giro hacia la izquierda de 45o y continuar en línea recta.
- 2 pulsaciones seguidas: giro de 90o en sentido contrario y continuar.
- 3 pulsaciones seguidas: giro de 180o y continuar.
- 4 pulsaciones seguidas: el robot debe detenerse y mover el brazo de arriba abajo 2 veces.

8.5.2. Como se ha realizado

Para la realización de esta misión teníamos que leer el valor del sensor de choque, para ello hacemos uso de una función de Arduino, **digitalRead(pinChoque)**, la cual nos da el valor 0 o 1, como se específica en la sección 3.5, y cuando detecta las pulsaciones obtenidas, realizamos la operación correspondiente enviando pulsos a los motores y esperando el tiempo necesario como en la misión 1[8.1].

8.6. Misión 3.1

8.6.1. Enunciado

Programa una función en la que el robot avance indefinidamente hasta que alcance una línea negra. Si encuentra algún obstáculo, debe parar y cambiar de trayectoria.

8.6.2. Como se ha realizado

Para realizar esta misión hacemos uso del método **read()** de la librería del sensor de reflectancia, especificada en 5.2.2. Leemos el valor inicial, es decir, el suelo normal, y vamos avanzando y leyendo a la vez, una vez que haya cambiado el valor del suelo, detenemos los motores.

8.6.3. Escenario Especifico

Hemos puesto con cinta aislante una linea en el suelo.

8.7. Misión 3.2

8.7.1. Enunciado

Programa una nueva función que realice lo siguiente:

- Empezando en inicio, ir hasta la línea 1
- Al llegar a la línea 1, girar en sentido contrario y volver al inicio.
- En inicio, girar de nuevo en sentido contrario e ir a la línea 2.

- Al llegar a la línea 2, girar en sentido contrario y volver al inicio.
- En inicio, girar de nuevo en sentido contrario e ir a la línea 3.
- Al llegar a la línea 3, girar en sentido contrario y volver al inicio.
- En inicio, parar el robot.

8.7.2. Como se ha realizado

Para la realización de esta misión, usamos la misma librería que la misión anterior.

Debemos establecer una variable para saber porque linea vamos para ello necesitaremos un contador, si la variable esta a 0, es que estamos empezando, y cuando el robot llega a la primera linea, para volver al inicio, debe de leer 0 lineas, una vez en el inicio, la variable aumenta en 1, por lo tanto avanza y debe leer e ignorar el valor de la variable, es decir, 1 linea, por lo tanto en la segunda linea pararía y volvería al inicio ignorando 1 linea, y así sucesivamente hasta que llegue al inicio después de leer la 4 linea.

8.7.3. Escenario Especifico

Hemos puesto con cinta aislante cuatro lineas en el suelo separadas entre ellas 30 cm.

8.8. Misión 4

8.8.1. Enunciado

Vamos a crear un programa en el robot intente localizar un objeto cercano con el sensor de ultrasonidos y se acerque a él, girando primero para apuntar al lugar en donde se encuentre el objeto y moviéndose luego hacia él hasta situarse a una distancia de unos 10 cm (esta distancia es aproximada y se debe validar experimentalmente). Al llegar a la distancia predeterminada, el robot debe mover el brazo y golpear el objeto.

8.8.2. Como se ha realizado

Nuestra implementación para esta práctica ha sido la de coger el valor inicial del sensor de ultrasonidos, y que el robot vaya girando poco a poco y en cuanto detecte un objeto con menor distancia, que se dirija hacia el y cuando el objeto esté a 2 cm, girar el robot 180º y bajar el brazo para que golpee el objeto.

8.8.3. Escenario Especifico

Hemos colocado un objeto para que lo encuentre y lo golpe.

8.9. Misión 5

8.9.1. Enunciado

Realiza un programa que cuente el número de veces que aplaudimos en un periodo de 10 segundos. El resultado debe mostrarse en la pantalla del PC, así

como una gráfica en la que se dibuje la intensidad del sonido que se ha ido obteniendo en esos 10 segundos.

8.9.2. Como se ha realizado

Debido a un problema con el micrófono, lo quemamos sin querer, realizamos dicha misión con un sensor de luz, ya que prácticamente es el mismo procedimiento. Nuestra misión consistía en leer durante 10 segundos los valores que nos generaba el sensor. Primero, obteníamos el valor de la luz ambiental para calibrar el sensor, y una vez realizado esto, empezamos a contar los incrementos de luz dentro de un umbral, y guardamos en un array todos los valores que ha generado el sensor para la futura gráfica. Para realizar los altos picos de luz, necesitamos una linterna o como hicimos nosotros con la linterna del smartphone. Con ello, cada vez que el sensor detectaba un valor de luz superior al de la luz ambiental más un umbral, contamos 1 al contador de encendidos. Y con ello detectaremos cuantas veces se ha encendido.

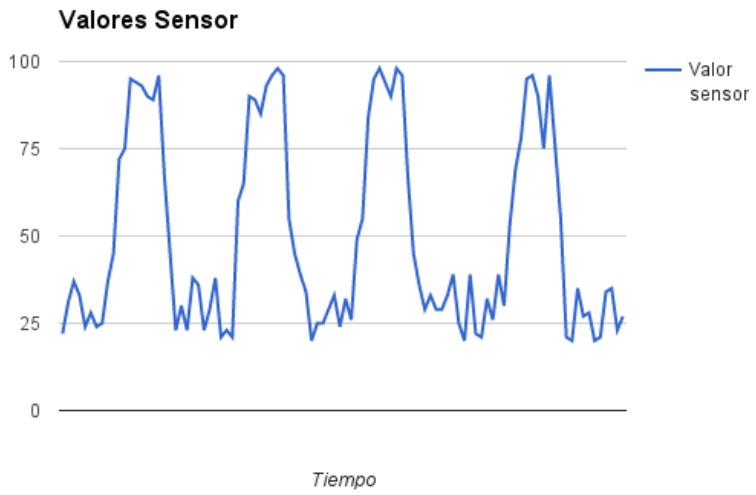


Figura 28: Gráfico generado por el sensor

8.9.3. Escenario Especifico

El sensor de luz como he explicado anteriormente, se sitúa en la parte anterior del robot.

8.10. Aparcamiento

8.10.1. Enunciado

Versión simplificada: El robot se mueve hacia adelante y, sin utilizar información de los sensores externos, llegado un punto "prefijado" realiza la maniobra de aparcamiento. En este caso, resultaría muy útil emplear los encoders que

tenemos en los motores y, si lo deseas, de sensores de contacto para asegurar que todo el proceso se realiza correctamente.

Versión completa: El robot se mueve hacia adelante y, cuando el "conductor" desea buscar aparcamiento, presiona un botón para comenzar dicha búsqueda (por ejemplo, con la barra espaciadora). En ese momento, el sensor de ultrasonidos comienza la búsqueda de una plaza de aparcamiento con tamaño suficiente para el robot. Al encontrarla, realiza la maniobra de aparcamiento. Al finalizar, el robot debe quedar en línea con el resto de "coches" y evitar a toda costa cualquier posible "roce".

8.10.2. Como se ha realizado

Simplificada: En esta versión leemos el sensor ultrasonido inicialmente y guardamos ese valor. Avanzamos hacia delante mientras vamos comprobando si los nuevos valores del sensor son mayores que el inicial más un pequeño umbral. Si es así significará que tenemos un hueco entre los obstáculos y, en cuanto obtengamos un nuevo valor como el inicial, realizamos la maniobra de aparcamiento. Que previamente hemos diseñado.

Avanzada: En este caso, lo que hemos realizado ha sido añadir a la versión simplificada un contador, para que cuente el tiempo que tarda en volver a leer un obstáculo, y si es mayor que el tiempo establecido por nosotros entonces el hueco será lo suficientemente ancho para establecer el aparcamiento debidamente.

8.10.3. Escenario Especifico

Para el aparcamiento simplificado hemos simulado un aparcamiento con dos cajas separadas entre si como si fuesen dos vehículos. Sin embargo para el avanzado hemos optado por poner tres cajas.

Referencias

[1] Placa controladora Bq ZUM BT328
`http://www.bq.com/es/placa-zum-bt`