Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA DE COMUNICACIONES

Trabajo Fin de Grado

DISEÑO DE INTERFACES ENTRE SENSORES Y ACTUADORES DE LEGO MINDSTORMS CON MICROCONTROLADORES COMERCIALES

Autor: Emilio Fernández Moreno

Tutor/es: D. Julio Pastor Mendoza

2017

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA DE COMUNICACIONES

Trabajo Fin de Grado

DISEÑO DE INTERFACES ENTRE SENSORES Y ACTUADORES DE LEGO MINDSTORMS CON MICROCONTROLADORES COMERCIALES

Autor: Emilio Fernández Moreno

Tutor: D. Julio Pastor Mendoza

TRIBUNAL:

Presidente: Pedro Alfonso Revenga de Toro

Vocal 1º: José Luis Martín Sánchez

Vocal 2º: Julio Pastor Mendoza

CALIFICACIÓN: ……..

FECHA: ……..

"Si una persona es perseverante, aunque sea dura de entendimiento,
se hará inteligente; y aunque sea débil se transformará en fuerte"

Leonardo Da Vinci

A mi tutor Julio:
Por querer realizar este TFG conmigo y ayudarme en ello.

A mi familia:
Por permitirme la oportunidad de realizar este Grado y apoyarme en todo momento.

A Nata:
Por estar siempre a mi lado y nunca dejar de confiar en mí.

Índice

Índice de figuras

Índice de tablas

I. Resumen

El presente Trabajo Fin de Grado, se basa en el estudio de dos plataformas robóticas muy extendidas dentro de la robótica educativa, y su posterior adaptación entre ambas. Para su realización, se han seleccionado la plataforma LEGO Mindstorms NXT y Arduino.

Se han desarrollado dos líneas de trabajo bien diferenciadas en las que se puede comprobar que ambas plataformas pueden llegar a ser compatibles. En la primera línea, LEGO Mindstorms controla sensores y actuadores de índole comercial, llegando hasta mantener una comunicación maestro-esclavo con Arduino. En la segunda línea, Arduino se encarga de controlar los periféricos de LEGO Mindstorms.

Palabras clave: LEGO Mindstorms, Arduino, sensores, actuadores, microcontrolador.

II. Abstract

This Bachelor Thesis is based on the study of two well-known robotic platforms in educational robotics and the adaptation between them. Two platforms have been selected to reach this goal: LEGO Mindstorms NXT and Arduino.

Two differentiated lines of study and research have been developed to demonstrate that both platforms can be compatible. In the first one, LEGO Mindstorms controls comercial sensors and actuators and gets to keep a master-slave communication with Arduino. In the second one, Arduino controls LEGO Mindstorms peripherals.

Key words: LEGO Mindstorms, Arduino, sensors, actuators, microcontroller.

III. Resumen extendido

La aparición de plataformas *open source* (código abierto), es decir, plataformas que pueden ser distribuidas y desarrolladas libremente, junto con la llegada al mercado de dispositivos y componentes electrónicos a un precio realmente asequible, han propiciado un auge en la robótica educativa, que se ha convertido en un sistema de enseñanza de carácter interdisciplinario con el que se pueden trabajar distintas materias como las ciencias, la tecnología, las matemáticas, etc.

Con la finalidad de crear herramientas que puedan ser útiles durante el aprendizaje en el ámbito de la robótica educativa, en este trabajo se ha realizado un estudio a fondo de dos plataformas robóticas que en gran medida han sido diseñadas con fines didácticos, LEGO Mindstorms y Arduino.

Este estudio ha desembocado en dos vertientes en las que ambas plataformas están interconectadas.

La primera vertiente ha consistido en utilizar el ladrillo inteligente (*brick*) NXT como núcleo de control de sensores y actuadores que se encuentran actualmente en el mercado y son de bajo coste. También se ha añadido al sistema una tarjeta Arduino Nano, que se comunica con el *brick* mediante un bus bajo la norma RS-485, permitiendo de esta manera ampliar las prestaciones que tiene el *brick* NXT. En la figura 1 se puede ver el diagrama de bloques del sistema.

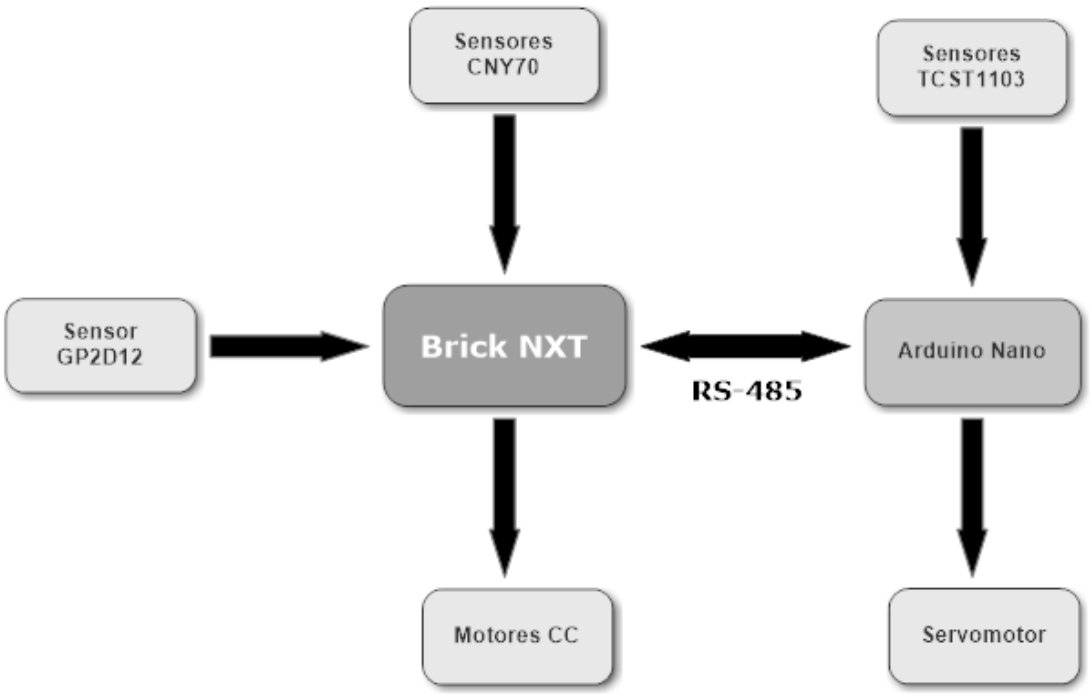


Figura 1. Diagrama de bloques del primer sistema

Inicialmente se ha realizado un estudio pormenorizado del kit LEGO Mindstorms NXT, con el que se han analizado las principales características de todos los dispositivos incluidos en el kit. Para esta primera vertiente se ha centrado el estudio en el *brick* inteligente, prestando especial atención a los puertos de entrada (puertos para sensores) y a los puertos de salida (puertos para actuadores), en los que se conectarán los sensores y actuadores seleccionados para suplir la funcionalidad de los periféricos que incluye el kit. También se ha analizado el puerto de comunicaciones a alta velocidad, a través del cual se habilitará la comunicación entre el *brick* NXT y la tarjeta Arduino Nano.

La segunda vertiente ha consistido en utilizar la tarjeta Arduino Nano como núcleo central de un sistema embebido diseñado para controlar los periféricos del kit. Este sistema incluye un driver de doble canal con capacidad para controlar dos servomotores del kit. La figura 2 muestra el esquema del sistema.

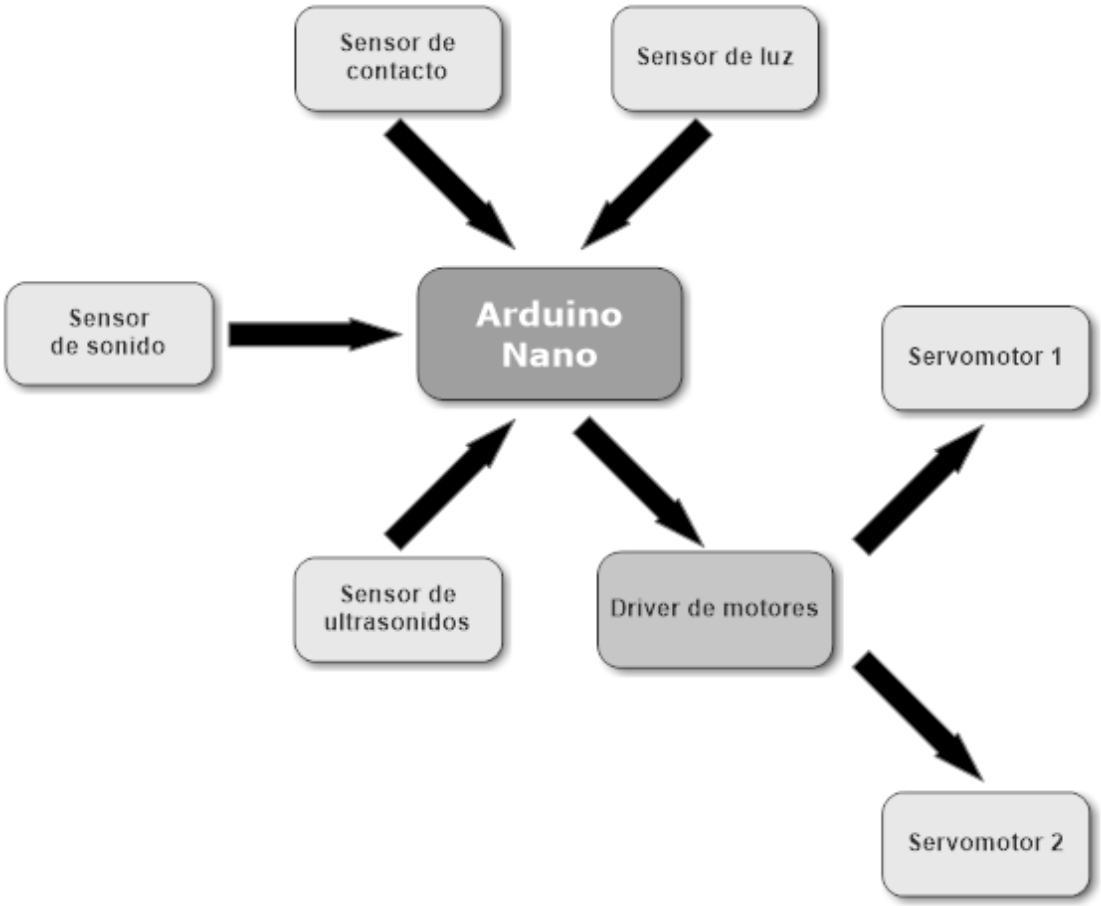


Figura 2. Diagrama de bloques del segundo sistema

Una vez realizado el estudio de los sensores y actuadores del kit LEGO Mindstorms NXT, se ha completado un estudio de la plataforma Arduino, centrado en la tarjeta Arduino Nano. Dicha placa será la encargada de sustituir al *brick* NXT siendo capaz de mantener las mismas prestaciones.

Finalizado el estudio de ambas plataformas, el objetivo perseguido ha sido interconectarlas para crear dos robots con características similares pero con tecnologías intercambiadas.

Como resultado del estudio previo, se detectó que los sensores utilizados para conectarse al *brick* inteligente, necesitan de una electrónica de acondicionamiento para poder ser funcionales, por tanto, ha sido necesario el diseño y fabricación de varias placas de circuito impreso (PCB). Para el sistema embebido también se ha diseñado y fabricado una PCB que hace más sencilla la conexión de los periféricos del kit LEGO Mindstorms NXT, liberando así de cableado al sistema.

Con todo el material necesario para la implementación de los dos robots preparado, se ha procedido a su montaje.

El robot A ha sido diseñado bajo un chasis de DM (densidad media), cortado mediante la técnica del corte por láser y estructurado en dos plantas para que pueda ser más compacto. Este robot está basado en el *brick* inteligente que se encargará de:

- Controlar los motores para obtener movimiento
- Controlar sensores reflexivos que le dotarán de "visión", ya sea para poder seguir una línea o detectar obstáculos
- Controlar sensores de transmisión que le permitirán tener un control preciso del movimiento
- Comunicarse con una placa Arduino para poder ampliar su número de puertos

El robot B por el contrario, ha sido diseñado bajo un chasis elaborado mediante impresión 3D y con un sistema embebido que permite controlar y conectar de una forma sencilla los periféricos del kit LEGO Mindstorms NXT.

En la figura 3 se puede observar el resultado final de ambos robots.

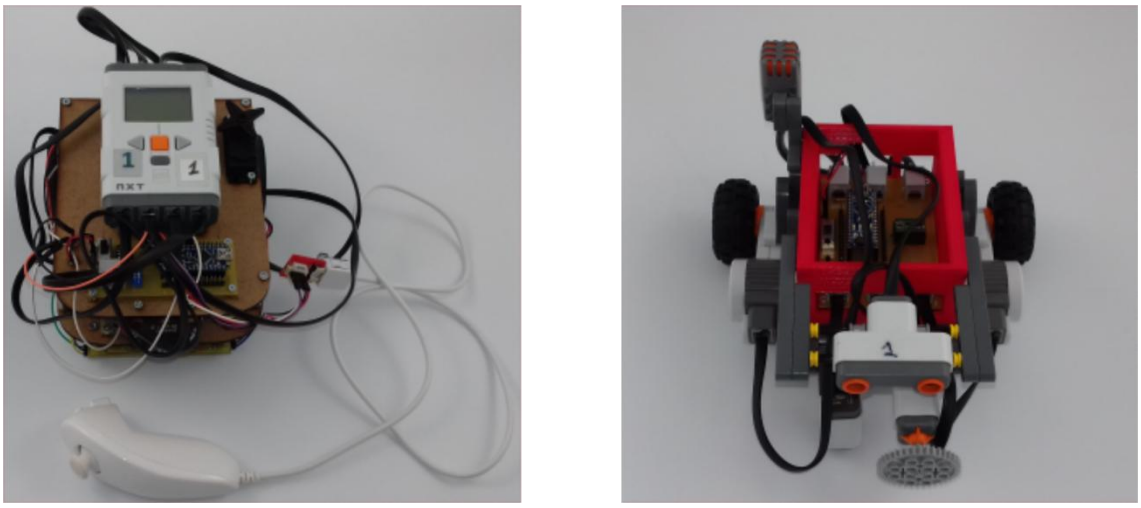


Figura 3. De izda. a dcha. robot A y robot B

IV. Memoria

1. Introducción

1.1. Robótica educativa

Desde principio de la década del 2000 se empezó a reflejar un descenso significativo del interés de los alumnos en el ámbito de las disciplinas científico-tecnológicas. Por este motivo, se ha producido un impulso notable de iniciativas STEAM, tanto por parte de los organismos educativos de los países más desarrollados, como de compañías líderes vinculadas al ámbito tecnológico [1].

El término STEM es un acrónimo en inglés de las iniciales de cuatro materias o disciplinas académicas: Science, Technology, Engineering, Maths. Actualmente se ha empezado a hablar del término STEAM incluyendo también la disciplina del arte. Las iniciativas englobadas bajo este acrónimo, pretenden aprovechar sus similitudes y puntos en común para desarrollar un enfoque interdisciplinario del proceso de enseñanza y aprendizaje, utilizando todas las herramientas tecnológicas necesarias [1].

Uno de los ámbitos donde pueden desarrollarse las competencias STEM, es la robótica. Esto ha sido posible gracias a la aparición de dispositivos y componentes electrónicos con precios asequibles, que han permitido desarrollar prototipos de una forma que antes sólo podría haberse hecho en ambientes profesionales. De esta manera nace la robótica educativa, una corriente de enseñanza que promueve el aprendizaje basado en la experimentación y el autodescubrimiento.

1.2. Plataformas robóticas

Actualmente existen multitud de tarjetas que se usan como controladores de plataformas robóticas y que pueden ser utilizadas como recurso educativo, Crumble [2], micro:bit [3], Raspberry Pi [4], etc**.** Sin embargo, este trabajo se centrará en dos de las plataformas robóticas más utilizadas, LEGO Mindstorms y Arduino.

1.2.1. *LEGO Mindstorms*

La plataforma LEGO Mindstorms surgió de la colaboración entre LEGO y el MIT (Massachusetts Institute of Technology) en el año 1998, con la que desarrollaron construcciones programables. En su inicio, debían estar constantemente conectadas al ordenador, lo que provocó que se plantease la posibilidad de crear un ladrillo inteligente (*brick*) que ejerciera como tal, liberando a las construcciones de la necesidad de estar conectadas mediante cables a un ordenador. Así dio comienzo el proyecto LEGO Mindstorms [5].

Hasta la fecha han sido desarrolladas tres generaciones de LEGO Mindstorms. En la siguiente tabla se muestran las principales características de cada una [6] [7].

Generación	**RCX (1ª)**	**NXT (2ª)**	**EV3 (3ª)**
Procesador	Hitachi H8/3292	Atmel ARM7	Texas Instruments ARM9
Frecuencia de trabajo	16 Mhz	48 Mhz	300 Mhz
Memoria	16 Kb ROM 32 Kb RAM	256 Kb FLASH 64 Kb RAM	16 Mb FLASH 64 Mb RAM
Puertos	3 para motores 3 para sensores	3 para motores 4 para sensores	4 para motores 4 para sensores
Comunicaciones	Infrarrojos	USB Bluetooth	USB Bluetooth Wi-Fi
			

Tabla 1. Comparación entre las tres generaciones de LEGO Mindstorms [8] [9] [10]

Para llevar a cabo este trabajo se ha elegido la segunda generación, el NXT.

A pesar de estar descatalogado por LEGO y no ser la generación más actual, es un material que rivaliza con muchos de los microcontroladores del mercado actual y ofrece la posibilidad de mejorar tanto su vida útil como sus prestaciones. Además, muchas instituciones educativas disponen del NXT y uno de los objetivos de este proyecto es proporcionarle grandes posibilidades de ampliación.

1.2.2. *Arduino*

Arduino surgió en el año 2005, ante la necesidad que tenían los estudiantes de un instituto de Italia de trabajar con un microcontrolador económico para una asignatura del grado de Bellas Artes, ya que los que había en el mercado eran muy costosos [11].

Se trata de una plataforma *open source* (código abierto), es decir, tanto su software como su hardware son accesibles para que cualquier usuario pueda verlo y modificarlo si fuera necesario.

Dentro del mundo Arduino hay diversos modelos de tarjetas con características diferentes, tanto a nivel físico como a nivel de hardware. Las más utilizadas son Arduino UNO, Arduino Nano y Arduino Mini (Figura 4).

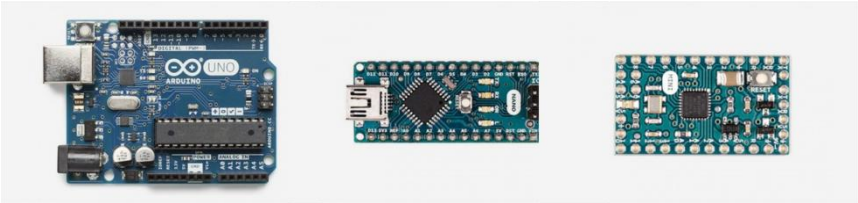


Figura 4. De izda. a dcha. tarjetas Arduino UNO, Arduino Nano y Arduino Mini [12]

Todas ellas poseen características muy similares. Están basadas en el microcontrolador ATMEGA328 de Atmel, con una frecuencia de trabajo de 16 Mhz y una memoria Flash para el almacenamiento de programas, de 32 Kbytes.
Tienen 14 puertos de entrada/salida digitales y, en el caso de la tarjeta Arduino UNO, 6 puertos de entrada analógicos (8 para las Arduino Nano y Arduino Mini).

Es importante destacar, que la carga de programas sobre la tarjeta se puede realizar mediante conexión USB en todas ellas, salvo en la tarjeta Arduino Mini, en la que es necesario utilizar un conversor externo [12].

Para la realización de este trabajo se ha elegido la tarjeta Arduino Nano debido a su reducido tamaño y a la simplicidad con la que se puede establecer la comunicación con el ordenador, ya que incluye un circuito integrado FT232RL adicional que tiene un interfaz USB.

1.3. Objetivos

Con la plataforma LEGO Mindstorms NXT como núcleo del trabajo, se pretende diseñar y montar un sistema en el que el *brick* NXT, sea capaz de trabajar de forma estable con sensores y actuadores ajenos al kit, mejorando de esta manera las prestaciones de la plataforma.

Por otra parte, se pretende diseñar un sistema embebido con una tarjeta Arduino Nano con el que se puedan monitorizar y controlar los sensores y actuadores del kit.

Mediante estas dos líneas de trabajo, se pretende crear una nueva vía para la enseñanza de la robótica en la que el alumno pueda desarrollar nuevos dispositivos, que puedan ser compatibles con la plataforma LEGO Mindstorms NXT disponible en muchos centros educativos.

1.4. Organización

El grueso de este trabajo se puede diferenciar en tres partes:

- Estudio de las plataformas LEGO Mindstorms NXT y Arduino
- Acondicionamiento de dichas plataformas
- Desarrollo de robots basados en las plataformas estudiadas

El estudio de las plataformas se realizará en los capítulos 5 y 6, donde se analizarán detenidamente las características de cada plataforma, además de seleccionar los sensores y actuadores comerciales más adecuados para conectarse al *brick* NXT.

El acondicionamiento estará incluido en los capítulos 7 y 8, en los que se explicará el diseño de los circuitos necesarios para compatibilizar una plataforma con la otra.

El desarrollo de los dos robots se realizará en los capítulos 9 y 10, donde se explicarán las características de cada robot y las aplicaciones que pueden tener en el dominio de la robótica.

Para finalizar el trabajo, en los capítulos restantes se encuentra un anexo con la técnica de fabricación de PCB utilizada, las conclusiones en las que se añade una línea de posibles mejoras que podrían ser realizadas en el futuro, el pliego de condiciones y el presupuesto que sería necesario para su realización.

2. Kit LEGO Mindstorms NXT

LEGO Mindstorms NXT es un kit de robótica compuesto por un *brick* inteligente en el que se pueden conectar sensores y actuadores. Los sensores que contiene el kit son de contacto, sonido, luz y ultrasonidos, existiendo más sensores en el mercado que no están incluidos en el kit. La parte de actuadores del kit la conforman los servomotores, que son los encargados de dotar de movimiento al robot [13].

Figura 5. *Brick*, sensores y actuadores del kit LEGO Mindstorms NXT [14]

2.1. Brick inteligente

Basado en un procesador ARM de 32 bits, con una memoria RAM de 64 KB, una memoria Flash de 256 KB y una frecuencia de trabajo de 48 MHz, se trata del cerebro del robot. Además del procesador mencionado (procesador principal), existe otro procesador AVR de 8 bits que se encarga de asistir al procesador principal para liberarle de carga de trabajo. Este coprocesador trabaja a 8 Mhz y se comunica periódicamente con el procesador principal mediante un bus I^2C operando a 380 Kbit/s, siendo el procesador principal el maestro de la comunicación [15].

El coprocesador se encarga de las siguientes tareas de bajo nivel:

- Controlar el encendido y apagado del *brick* a través del botón naranja central.
- Monitorizar el estado de la batería y enviar esa información al procesador principal.
- Generar las señales PWM (Pulse Width Modulation) para los tres puertos de salida a una frecuencia de 8 KHz con el ciclo de trabajo especificado por el procesador principal.

- Realizar las conversiones digitales de las señales analógicas de los puertos de entrada.
- Decodificar los botones del *brick* para que el procesador principal conozca qué botón ha sido presionado.

El *brick* tiene varios puertos de entrada/salida para conectar los distintos periféricos que incluye el kit. En su parte superior, se encuentran los tres puertos de salida identificados con las letras A, B y C, donde se conectarán los actuadores del robot. En la parte inferior, se encuentran los cuatro puertos de entrada identificados con los números 1, 2, 3 y 4, donde se conectarán los sensores del robot. En la parte central, están situados los cuatro botones con los que el usuario puede navegar por sus menús. Además, lleva incorporado una pantalla LCD monocroma con una resolución de 100x64 píxeles, que se comunica mediante bus SPI (Serial Peripherical Interface) con el procesador principal y posee un altavoz de un canal con 8 bits de resolución, capaz de emitir tonos en el rango de frecuencias de 2 a 16 kHz.

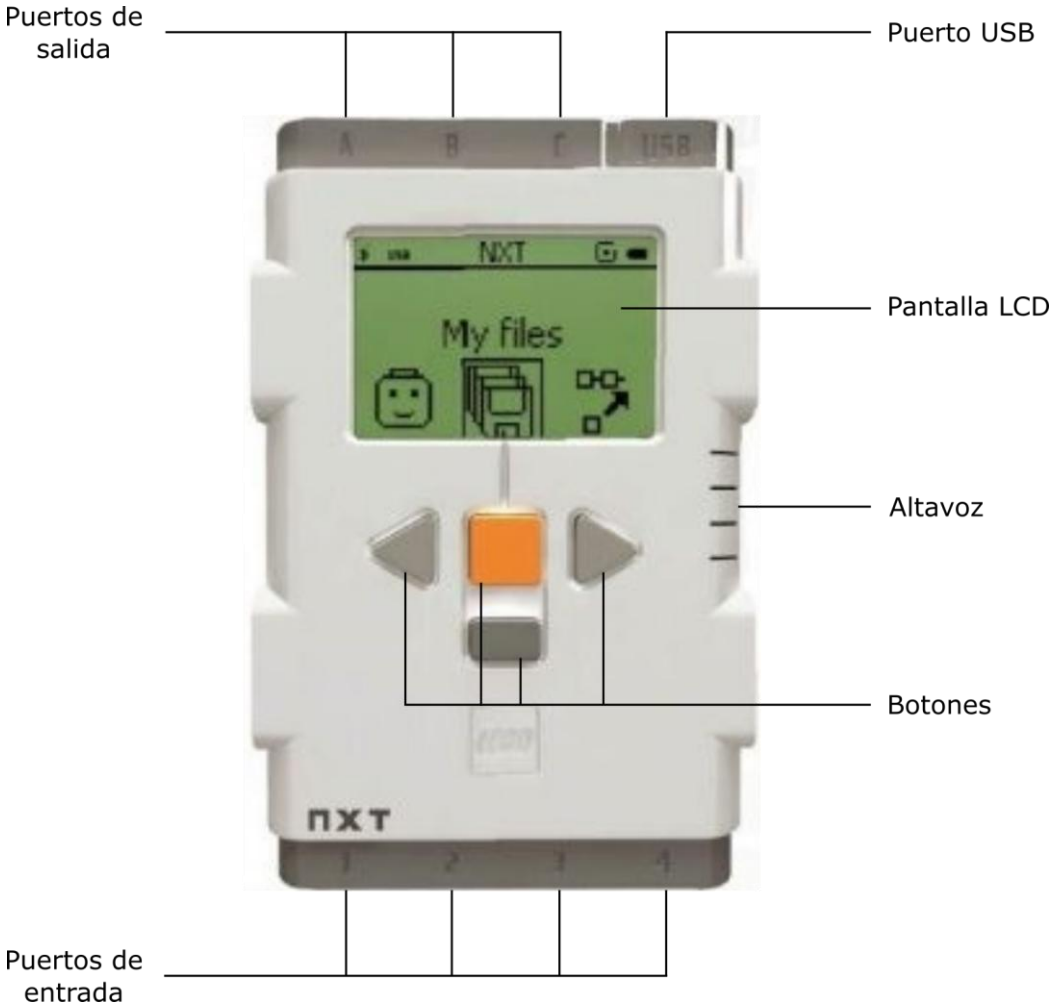


Figura 6. Partes del *brick* NXT [16]

2.2. Puertos

Los puertos de entrada y salida del *brick* utilizan un conector de tipo RJ12 (similar a un RJ11 pero con la pestaña desplazada). A continuación se detallan los puertos de salida y entrada que posee el *brick* inteligente.

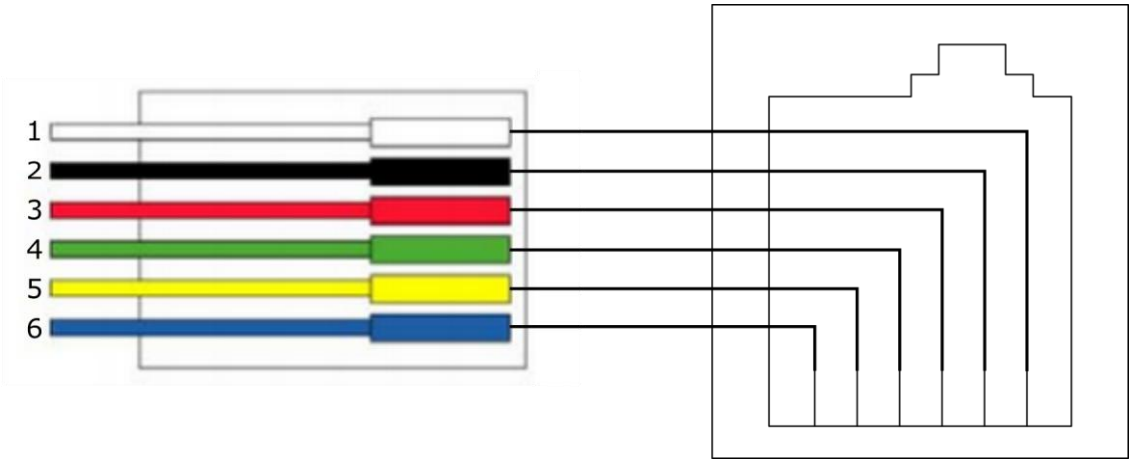


Figura 7. Conexionado puertos NXT [17]

2.2.1. *Puertos de salida*

Los puertos de salida del *brick* contienen seis conexiones que pueden ser diferenciadas en tres secciones: control de motores, alimentación y sensores de rotación.

Pin	Nombre	Función	Color
1	MA0	Señal de salida PWM	
2	MA1	Señal de salida PWM	
3	GND	Tierra	
4	POWERMA	+4.3V Alimentación	
5	TACHOA0	Señal de entrada encoder	
6	TACHOA1	Señal de entrada encoder	

Tabla 2. Conexiones puertos de salida *brick* NXT [18]

En la tabla, se puede observar que:

- Los pines número 1 y 2 transmiten señales de salida PWM, que serán enviadas a los motores para controlar su velocidad y sentido de giro.
- Los pines 3 y 4, son una toma de alimentación de 4.3 V utilizada por todos los puertos del *brick*, con capacidad para suministrar hasta 180 mA de corriente máxima.
- Los pines 5 y 6, son pines de entrada al *brick* que contienen las señales digitales del encoder óptico que llevan incorporado los motores.

2.2.2. *Puertos de entrada*

Los puertos de entrada del *brick* tienen seis conexiones que, al igual que los puertos de salida, se pueden diferenciar en tres secciones: señales analógicas, alimentación y señales digitales.

Pin	Nombre	Función	Color
1	ANA	Señal de entrada analógica, +9V	
2	GND	Tierra	
3	GND	Tierra	
4	IPOWERA	+4.3V Alimentación	
5	DIGIAI0	Señal de reloj I^2C (SCL), RS-485 B	
6	DIGIAI1	Señal de datos I^2C (SDA), RS-485 A	

Tabla 3. Conexiones puertos de entrada *brick* NXT [18]

- El pin número 1, es un pin de entrada analógico que está conectado al conversor analógico digital (ADC) de 10 bits del coprocesador y a una resistencia de pull-up de 10 KΩ. También está conectado al generador de corriente que se usa para suministrar mayor corriente a los sensores activos procedentes de la versión anterior de LEGO Mindstorms, el RCX.
- Los pines número 2, 3 y 4, forman parte de la misma toma de alimentación explicada en los puertos de salida.
- Los pines 5 y 6, se utilizan para la comunicación digital con otros dispositivos mediante el protocolo I^2C. En el puerto 4, estos dos pines también están conectados a un controlador de comunicación mediante la norma RS-485, que permite una comunicación a mayor velocidad.

2.2.3. *Puerto de comunicación a alta velocidad*

El puerto 4 del *brick* NXT tiene la posibilidad de mantener una comunicación a alta velocidad con otro dispositivo mediante una conexión RS-485. Esta norma, utiliza un bus de transmisión multipunto diferencial de dos hilos, es decir, la tensión diferencial entre los dos hilos define el nivel lógico que se va a enviar. Se trata de una norma diseñada para comunicaciones que necesiten altas velocidades a largas distancias (35 Mbps hasta 10 metros y 100 Kbps hasta 1200 metros) debido a que utiliza el balanceo de líneas [19]. Los pines 5 y 6 del puerto de entrada número 4, están conectados al conversor ST485 que será el encargado de adaptar los niveles lógicos para mantener comunicaciones, siguiendo la norma RS-485, con otros dispositivos.

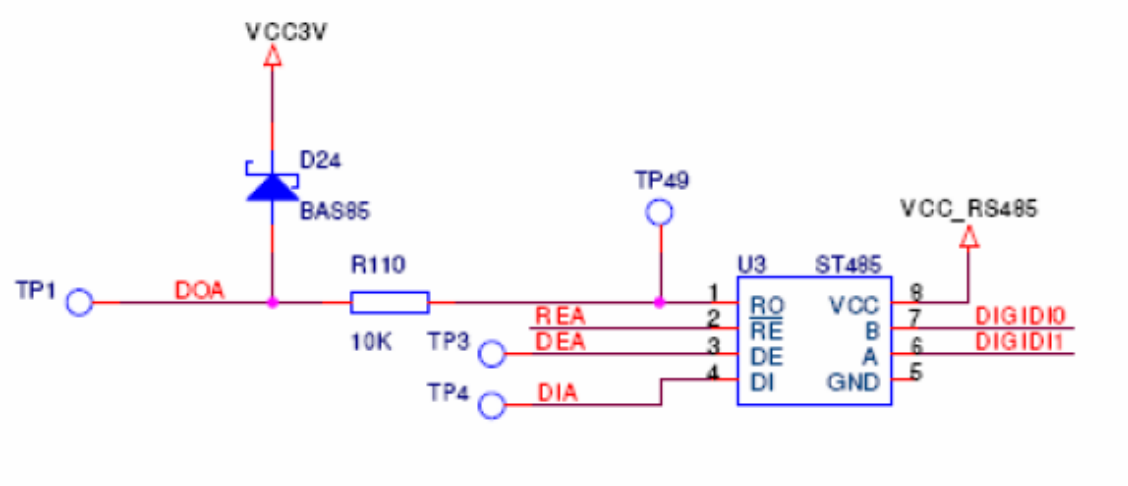


Figura 8. Circuito RS-485 *brick* NXT

Este conversor, consta de cuatro pines (RO, DI, DE y RE) los cuales están conectados al procesador principal (Figura 8).
RO y DI son los encargados de la recepción y transmisión de datos durante la comunicación, mientras que DE y RE se encargan de configurar el conversor como emisor o receptor. Esto último se debe a que la comunicación RS-485 del *brick* NXT, sólo puede ser half-dúplex, es decir, los dos dispositivos no pueden transmitir datos a la vez. Como el pin RE es activo a nivel bajo y el pin DE es activo a nivel alto, el procesador principal utiliza un único pin para el control de ambas señales (Figura 9).

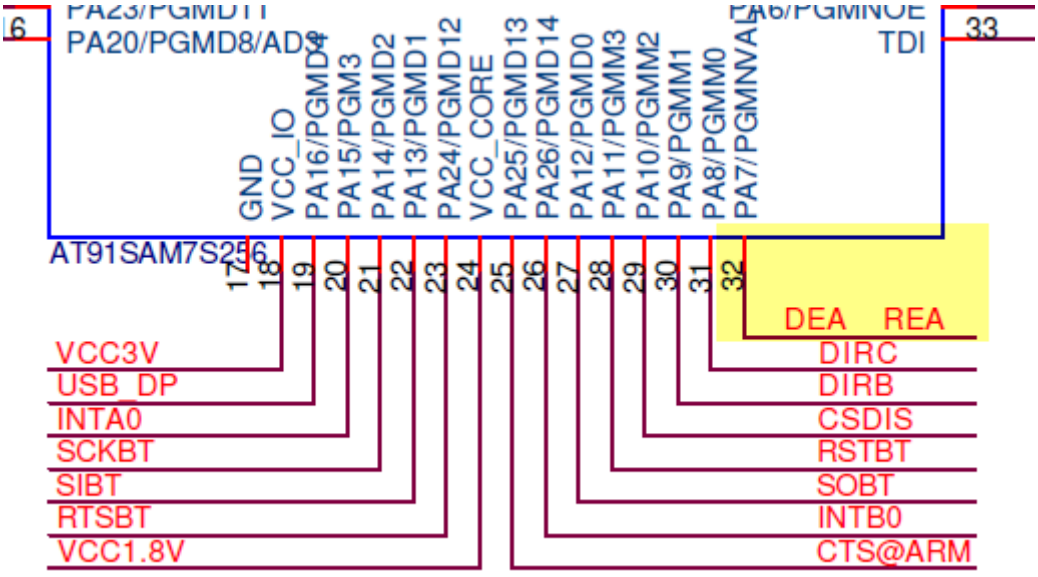


Figura 9. Control de comunicación RS-485

Los otros dos pines restantes, A y B, son los que constituyen el bus RS-485.

2.3. Sensores

Dentro de la categoría de sensores, LEGO diferencia entre tres tipos: sensores activos, sensores pasivos y sensores digitales.

2.3.1. *Sensores activos*

Este tipo de sensores son los pertenecientes a la anterior versión de LEGO Mindstorms, conocido como RCX, por tanto, requieren un adaptador para conectarse al *brick* NXT. Para poder tener retrocompatibilidad con los sensores del *brick* RCX, el *brick* NXT lleva incorporada una fuente de corriente que suministra, aproximadamente, 18 mA y que junto con el firmware de LEGO permite al *brick* NXT trabajar con los sensores de la versión anterior. Los sensores de esta categoría son los sensores de rotación y de luz del RCX (Figura 10).

Figura 10. Sensores de rotación y luz del RCX [20] [21]

2.3.2. *Sensores pasivos*

Dentro de esta categoría se encuentran los sensores analógicos, los cuales no necesitan un suministro de corriente especial. Estos sensores son muestreados, por el ADC, a una frecuencia de 333 Hz, debido a que los sensores activos necesitan ser muestreados a dicha frecuencia. Dentro del kit LEGO Mindstorms NXT se encuentran los sensores de contacto, de luz y de sonido (Figura 11).

Figura 11. De izda. a dcha. sensores de contacto, luz y sonido del NXT [22] [23] [24]

2.3.2.1. Sensor de contacto

Este sensor proporciona al robot el sentido del "tacto", permitiendo así que tenga conocimiento de si está en contacto con un objeto o superficie.

Está compuesto por un switch y una resistencia en serie (Figura 12), que sirve de elemento de protección en caso de conectar el sensor a un puerto de salida, evitando así, un cortocircuito.

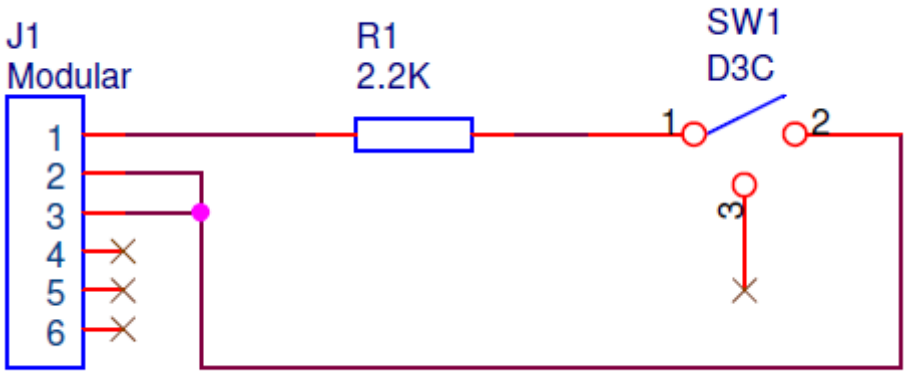


Figura 12. Circuito sensor de contacto

El funcionamiento es sencillo, cuando el sensor es presionado se cierra el switch y existe un cambio de tensión en el puerto de entrada analógico.

2.3.2.2. Sensor de luz

Este sensor dota del sentido de la "visión" al robot, es decir, permite al robot conocer la iluminación de una habitación o diferenciar superficies coloreadas.

Se trata de un sensor reflexivo compuesto por un LED (emisor) y un fototransistor (receptor) adyacentes dentro de la misma carcasa. El principio de funcionamiento es el siguiente, el LED emite luz que se refleja en una superficie y rebota hacia el fototransistor, que se encarga de medir la intensidad de luz reflejada.

Tiene dos modos de funcionamiento:

- Detectar la luz ambiente.
- Medir la cantidad de luz de reflejada. Para este modo es necesario activar el LED, por software, que lleva incorporado.

2.3.2.3. Sensor de sonido

Este sensor proporciona al robot el sentido del "oído", es decir, detecta el nivel de decibelios (dB) existente en torno a él.

Compuesto por un micrófono y una electrónica de acondicionamiento basada en filtros, se encarga de medir los niveles de presión del sonido.
Tiene dos modos de funcionamiento:

- Detectar frecuencias existentes dentro del espectro auditivo del ser humano (20 Hz – 20 KHz).
- Detectar frecuencias incluso fuera del rango auditivo del ser humano.

2.3.3. *Sensores digitales*

Estos sensores contienen la lógica y recursos necesarios para trabajar de una forma independiente, es decir, funcionan de forma autónoma y su interacción con el *brick* es únicamente para intercambiar información. Esto es posible porque estos sensores llevan incorporado un microcontrolador que se encarga del funcionamiento del sensor y de la comunicación con el procesador principal del *brick*. El protocolo de comunicación entre estos sensores y el procesador principal, es I^2C y la velocidad de comunicación es 9600 baudios (bits/s). El único sensor digital que contiene el kit LEGO Mindstorms NXT, es el sensor de ultrasonidos (Figura 13).

Figura 13. Sensor de ultrasonidos NXT [25]

2.3.3.1. Sensor de ultrasonidos

Este sensor dota al robot, al igual que el sensor de luz, del sentido de la "visión", aunque de forma diferente.

Compuesto por dos transductores, uno emisor y otro receptor, se encarga de la detección de obstáculos y la medición de la distancia existente entre el sensor y dichos obstáculos.

El funcionamiento de este sensor consiste en que el emisor (altavoz), emite una ráfaga de ultrasonidos a 40 kHz de frecuencia que rebota en el obstáculo más cercano y transcurrido un intervalo de tiempo (cuya longitud viene determinada por la distancia al obstáculo detectado), retorna al receptor (micrófono). Con el tiempo que tarda el ultrasonido en ir y en volver (tiempo de vuelo) y la velocidad del ultrasonido en el aire se calcula la distancia a la que se encuentra el obstáculo.

El encargado de controlar este funcionamiento es el microcontrolador eSC de 4 bits, basado en un procesador de sonido, que lleva incorporado el sensor, que además de realizar las medidas, establece una comunicación con el *brick* inteligente, mediante el bus I^2C, a través de la cual envía toda la información para que pueda ser procesada por el procesador principal.

2.4. Actuadores

Los únicos actuadores incluidos en el kit LEGO Mindstorms NXT son los servomotores (Figura 14), motores de corriente continua que incorporan un sensor de posicionamiento del eje (encoder en cuadratura) con un grado de resolución.

Figura 14. Servomotor NXT [26]

2.4.1. *Mecánica*

Un motor de corriente continua con escobillas genera la máxima potencia mecánica cuando gira a alta velocidad (del orden de miles de revoluciones por minuto). Estas altas velocidades son demasiado elevadas para mover una rueda de un robot y si se disminuye la tensión para obtener la velocidad deseada, la potencia entregada sufre una gran disminución proporcionando normalmente poco par (fuerza de rotación). Por este motivo suele utilizarse una reductora formada por engranajes que reduce la velocidad de giro a la vez que aumenta el par permitiendo el uso de motores de baja potencia y menor coste. En la Figura 15 puede la estructura interna del motor del kit LEGO Mindstorms NXT.

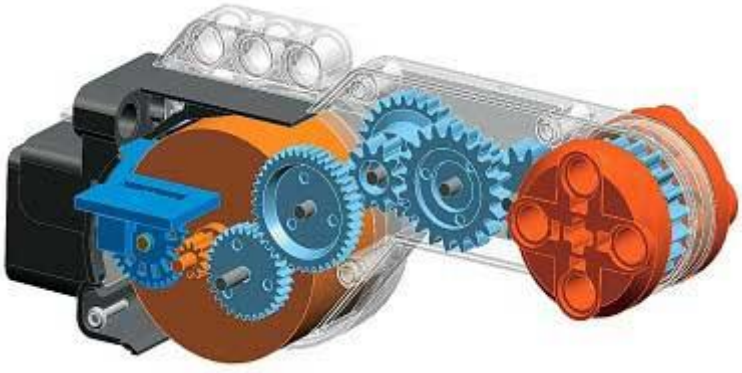


Figura 15. Reductora servomotor NXT (en azul oscuro puede verse el encoder) [27]

El tren de engranajes del motor del NXT se detalla a continuación:

Etapa (engranajes)	Número de dientes	Reducción
1ª	10:30:40	1:4
2ª	9:27	1:3
3ª	10:20	1:2
4ª	10:13:20	1:2
Total		1:48

Tabla 4. Relación entrada-salida servomotor NXT

Observando la tabla se puede ver que la relación entrada-salida del motor del NXT es 1:48, es decir, que para que el servomotor gire una vuelta completa, el motor de corriente continua tiene que girar 48 vueltas y del mismo modo, el par es 48 veces mayor.

Conocer la reductora que tiene el motor es importante, pero si no se conoce la velocidad angular que tiene el motor que se encuentra conectado a dicha reductora, se convierte en un dato que carece de utilidad. En la figura 16 se pueden ver la velocidad angular que tiene el motor en su etapa de salida en función de la carga que lleve incorporada.

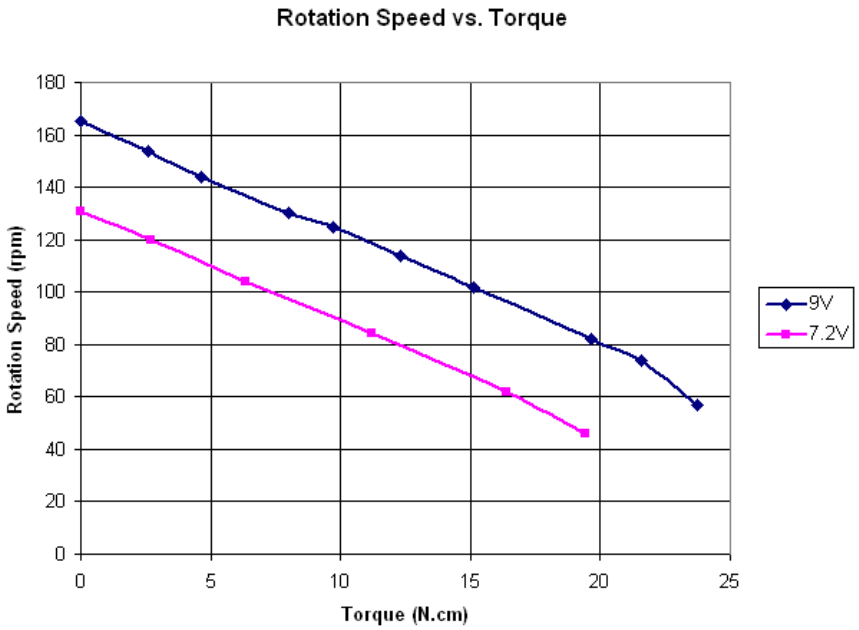


Figura 16. Relación velocidad angular-torque servomotor NXT [27]

Se va a centrar la atención en la velocidad angular cuando no existe carga. La línea morada indica la velocidad angular medida en revoluciones por minuto (rpm) del motor cuando se alimenta a 9 V sin carga. Se puede observar cómo llega a superar las 160 rpm. Utilizando la línea rosa que indica la velocidad angular cuando la alimentación es de 7.2 V (alimentación utilizada en este trabajo), se observa que la velocidad de rotación máxima aproximada asciende a 130 rpm.

Aunando los datos obtenidos (reductora y velocidad angular en la etapa de salida) se puede obtener que la velocidad angular del motor de corriente continua que lleva incorporado el servomotor del kit LEGO Mindstorms NXT es aproximadamente de 6.240 rpm alimentando a 7.2 V.

2.4.2. *Electrónica de control*

El control de motores del NXT se realiza desde el *brick* inteligente, que incluye los driver necesarios para controlar las señales que se envían a los tres puertos de salida. El puerto A, está conectado al circuito integrado LB1930M, un driver de motores de un sólo canal que implementa un puente en H, capaz de suministrar hasta 1A de corriente. Los puertos B y C, están conectados al circuito integrado LB1836M, un driver de motores similar al anterior pero con la diferencia de que tiene dos canales, es decir, es capaz de controlar dos motores de manera independiente.

Cada puente en H está compuesto por cuatro transistores bipolares (Q1, Q2, Q3 y Q4) que, en función de cuál de ellos entre en conducción, permitirá al motor girar en un sentido u otro, o permanecer estático. El circuito de control solo permite la conducción de dos transistores en el mismo momento, teniendo así varias posibilidades de funcionamiento (Figura 17):

- Q1 y Q4 entran en conducción, por tanto, el motor comienza a girar en un sentido.
- Q2 y Q3 entran en conducción, por tanto, el motor girará en sentido opuesto a la configuración anterior.
- Q1 y Q3 o Q2 y Q4 entran en conducción se produce un freno eléctrico aumentando la fuerza de frenado. Si se desconecta la alimentación o si se dejaran abiertos los cuatro transistores, el motor se frenaría por inercia.

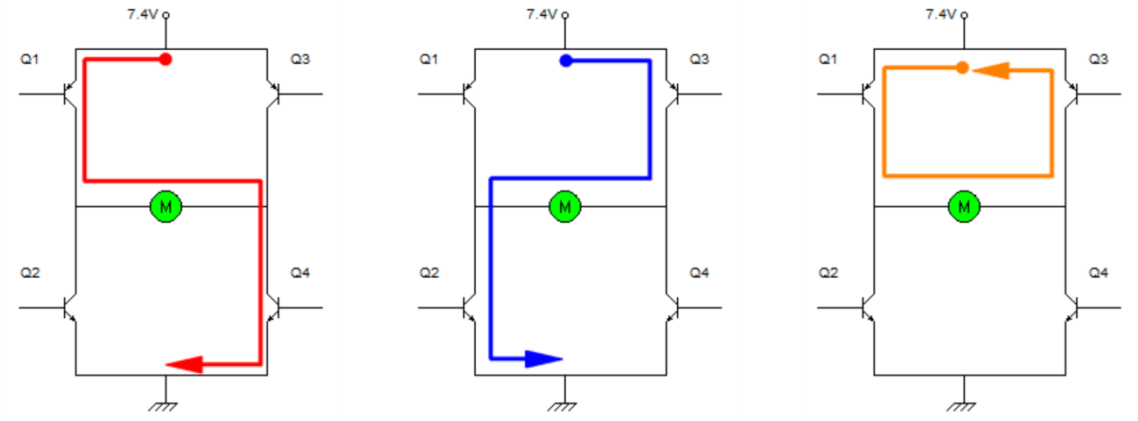


Figura 17. Distintas configuraciones puente en H

Además el circuito de control no permite activar a la vez los transistores Q1 y Q2 o Q3 y Q4, ya que se produciría un cortocircuito que dañaría irreversiblemente el dispositivo.

Otro aspecto importante a tener en cuenta en un motor, además del sentido de giro, es la velocidad de giro. La velocidad de los motores se regula mediante una modulación por ancho de pulso (PWM), una técnica que consigue producir el efecto de una señal analógica mediante la variación del ciclo de trabajo de una señal digital. Cuando una señal digital cambia de estado con una frecuencia lo suficientemente rápida, se comporta como una señal analógica que varía su valor en función del ciclo de trabajo (Figura 18).

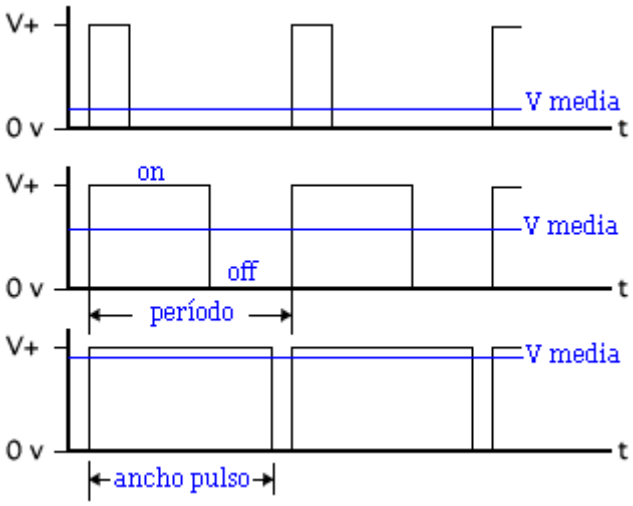


Figura 18. Señales PWM [28]

Esta señal PWM es generada por el coprocesador y enviada al driver de motores que será el encargado de controlar el sentido y velocidad del motor. Como se puede observar en la Figura 19, la frecuencia de la señal PWM generada en el *brick* NXT es 8 kHz (periodo de 125us).

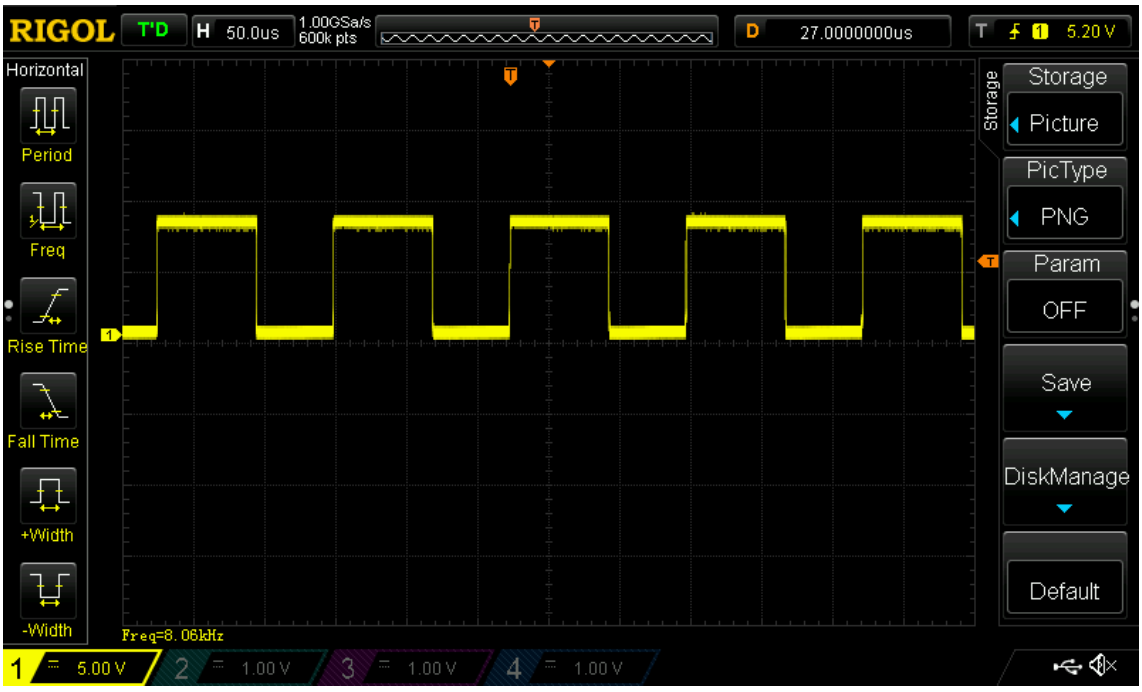


Figura 19. Señal PWM generada por el *brick* NXT

2.4.3. *Sistema de posicionamiento*

Los motores contienen en su interior un sistema de posicionamiento que se utiliza para controlar el sentido y la velocidad de giro, además de la posición en la que se encuentra el motor. Este sistema está basado en encoder ópticos en cuadratura y consta de tres partes: una fuente emisora de luz, un disco codificador (encoder) y dos receptores de luz con un pequeño desfase entre ellos (Figura 20) [29].

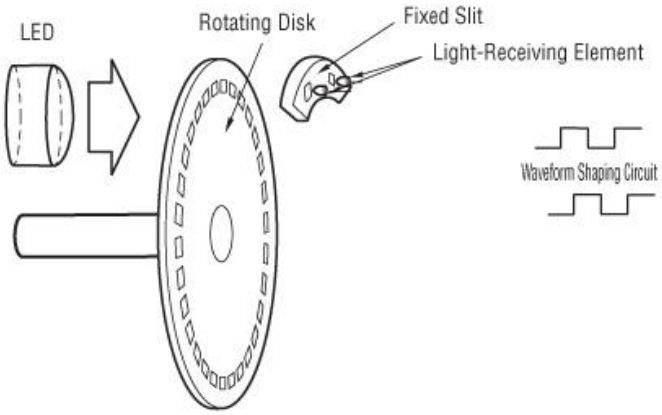


Figura 20. Encoder óptico en cuadratura [30]

El principio de funcionamiento es el siguiente: el disco codificador cuenta con ranuras que dejan pasar la luz desde el emisor hasta los receptores, por tanto, las señales que generarán los receptores tendrán dos estados en función de si detectan luz o no, lo que las convierte en señales digitales. De esta forma, ya se puede obtener la velocidad que lleva el motor, pero como se menciona anteriormente, las dos fuentes receptoras tienen un pequeño desfase entre ellas, es decir, una va a ir siempre por delante de la otra, con lo que en función de qué señal sea la que va adelantada se conocerá cuál es el sentido de giro del motor [29]. Esta explicación puede verse gráficamente en las figuras 21 y 22.

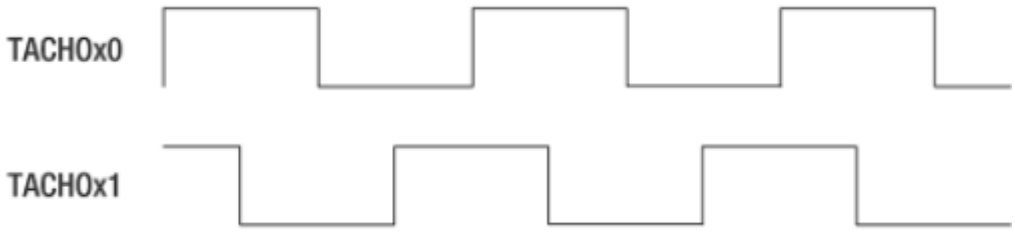


Figura 21. Señales encoder servomotor avanzando [29]

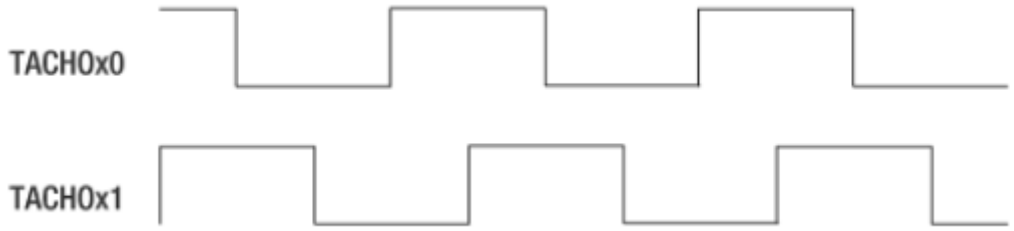


Figura 22. Señales encoder servomotor retrocediendo [29]

Para poder utilizar esta información correctamente es importante saber cuántos cambios de estado ocurren en el encoder por vuelta completa del motor.

Figura 23. Encoder servomotor NXT [27]

Como se puede observar en la figura 23, el encoder es un engranaje de 32 dientes con 12 ranuras que está unido directamente al engranaje de 10 dientes del motor de corriente continua, es decir, la relación entrada-salida es 10:32. Recordando que es necesario que el motor de corriente continua realice 48 giros para que el servomotor NXT de una vuelta, podemos calcular que el encoder realiza 15 giros por vuelta del motor. Esto supone que habrá 180 ranuras por vuelta, es decir, 360 cambios de estado por vuelta, lo que proporciona una precisión de un grado. En la Figura 15 puede observarse el encoder en color azul oscuro en su posición dentro del motor.

Las señales de los encoder no son cuadradas puras, por esta razón, el *brick* inteligente cuenta con un disparador de Schmitt que se encargará de convertir las señales de los encoder en señales digitales puras.

2.5. Herramientas de programación

Existe una gran variedad de entornos de programación que permiten realizar programas para el robot LEGO Mindstorms NXT en distintos lenguajes de programación, tanto gráfica como en código texto. Dentro de estos lenguajes predominan NXT-G, RobotC y NXC [32].

2.5.1. *NXT-G*

Es un software oficial de LEGO basado en LabVIEW que utiliza un lenguaje de programación gráfico. La forma de programar es mediante bloques, lo que le hace el entorno de programación ideal para usuarios que no tengan conocimientos de programación. El inconveniente se encuentra a la hora de realizar programas complejos, ya que será necesario utilizar muchos bloques lo que dificultará su uso.

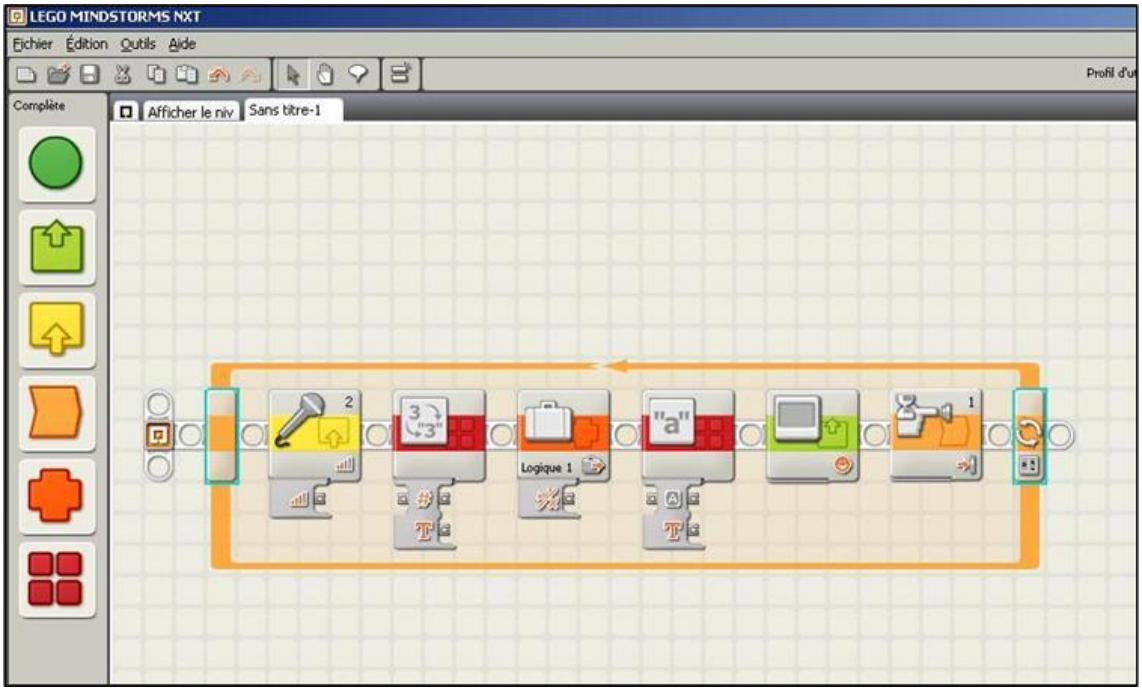


Figura 24. Ejemplo programa NXT-G [33]

2.5.2. *RobotC*

Es un lenguaje de programación en código texto basado en lenguaje C. Permite crear programas complejos de una forma más estructurada y utilizando menos recursos de memoria. El inconveniente es que para su utilización, es necesario tener conocimientos de programación al tratarse de un lenguaje textual.

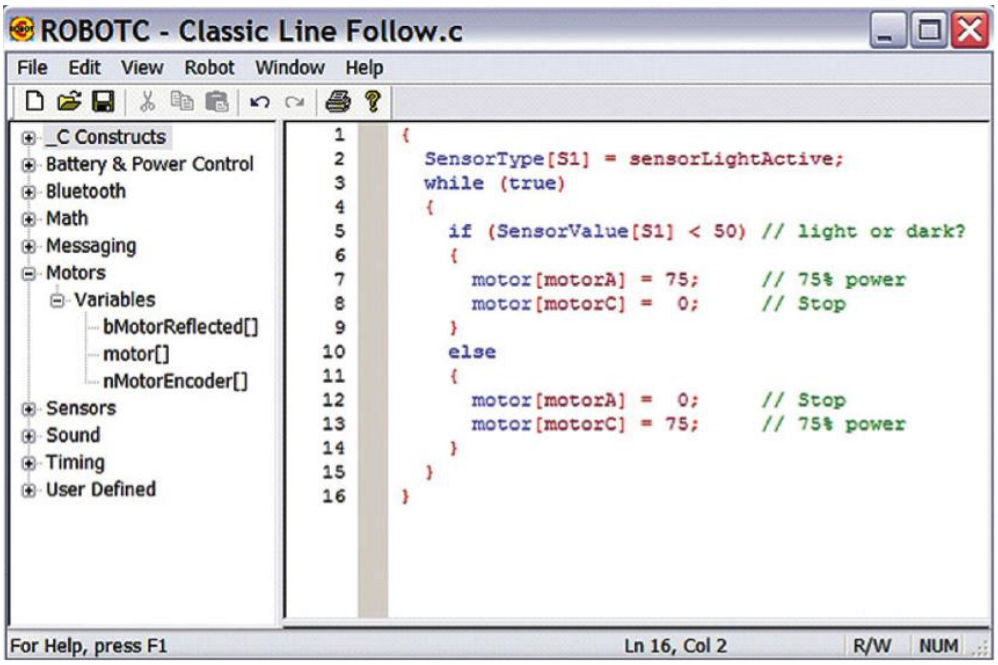


Figura 25. Ejemplo programa RobotC [34]

2.5.3. *(Not eXactly C)*

Es un lenguaje de programación en código texto similar a C. Utiliza el firmware original de LEGO por lo que no es necesario instalar uno nuevo en el *brick* para cargar los programas creados. Su principal ventaja es que se trata de un software libre y, por tanto, no hay que pagar licencia para su uso.

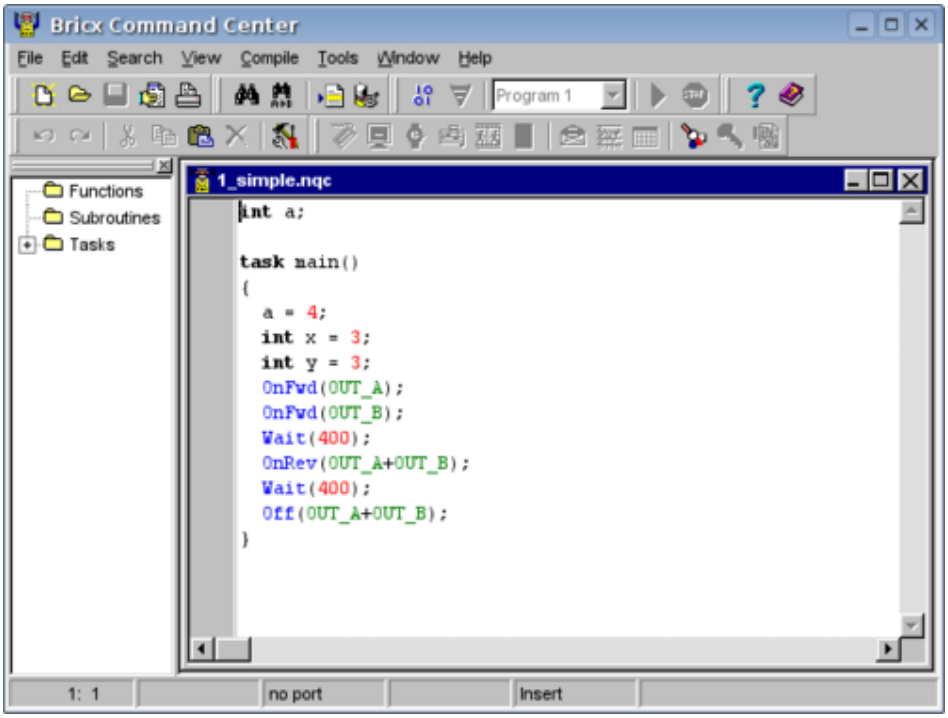


Figura 26. Ejemplo programa NXC [35]

3. Microcontrolador, sensores y actuadores comerciales

3.1. Microcontrolador

La tarjeta elegida para este trabajo es la Arduino Nano (Figura 27). Se trata de una placa basada en el microcontrolador ATMEGA328, que será utilizado para dos propósitos diferentes, el primero es comunicarse con el *brick* inteligente mediante el puerto de alta velocidad para aumentar así, el número de puertos de entrada y salida del *brick*, y el segundo es controlar los sensores y actuadores del kit LEGO Mindstorms NXT.

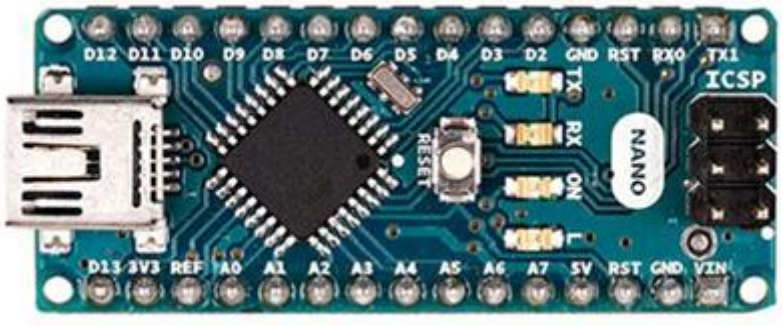


Figura 27. Placa Arduino Nano [36]

3.1.1. *Alimentación*

La placa Arduino Nano tiene dos formas de alimentarse: mediante cable USB o mediante baterías externas. El inconveniente de la alimentación mediante USB, es la necesidad de estar constantemente conectado al ordenador, por esta razón se ha elegido la segunda opción como forma de alimentación para este trabajo.

Para alimentar mediante baterías a la placa Arduino Nano es necesario hacerlo a través del puerto Vin ya que, como se observa en la figura 28, está conectado directamente al regulador de tensión que alimenta al microcontrolador y al resto de componentes de la placa.

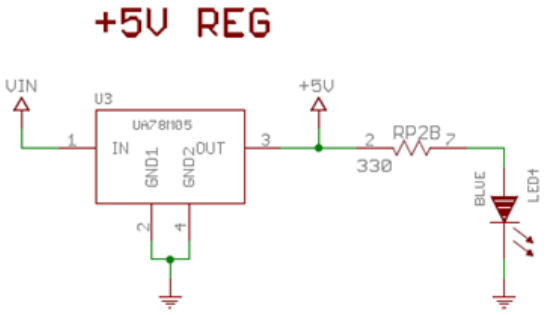


Figura 28. Circuito de alimentación Arduino

Para seleccionar la tensión ideal que alimentará la placa, hay que tener en cuenta la tensión mínima que necesita el regulador en sus terminales para funcionar según las especificaciones (tensión de *dropout*), 2 V según el fabricante, lo que hace necesario

que como mínimo la tensión de entrada sea de 7 V. En adición se ha introducido en el circuito, a la entrada del regulador de tensión, un diodo de protección para evitar daños en el sistema producidos por inversiones de polaridad. Esto conlleva una caída de tensión añadida en el circuito de aproximadamente 0.7 V, ya que el consumo de corriente que tendrá la placa será aproximadamente de 100 mA y, como se puede observar en la figura 29, ese será el valor de tensión que consuma el diodo para ese valor de circulación de corriente. Esto supone que la tensión de alimentación de la placa tenga que ser aproximadamente de 7.7 V.

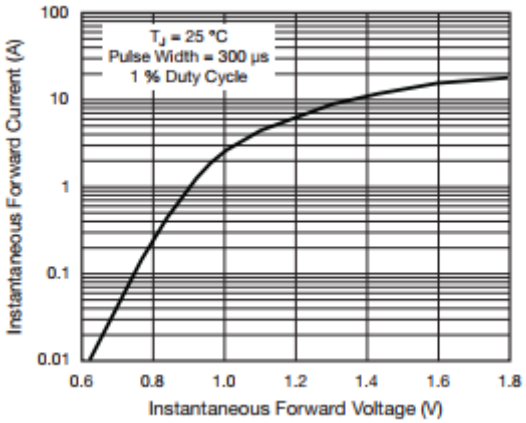


Figura 29. Curva V/I diodo 1N4007

3.1.2. *Puertos*

La placa Arduino Nano contiene 14 líneas de entrada/salida digitales que pueden suministrar hasta un máximo de 40 mA cada uno. A continuación se detallan las funciones especiales de alguna de estas líneas:

- RX0 y TX1 conforman el puerto serie de la placa y están conectados al circuito integrado FTDI USB-to-TTL para poder comunicarse con el ordenador a través de conexión USB.
- D2 y D3 se pueden configurar como interrupciones externas.
- D3, D5, D6, D9, D10 y D11 pueden generar señales PWM (pulse width modulation). La frecuencia por defecto de los puertos D3, D9, D10 y D11 es 490 Hz mientras que la de los puertos D5 y D6 es 980 Hz.
- D10, D11, D12 y D13 soportan comunicación SPI (Serial Peripheral Interface).

También contiene 8 puertos analógicos con 10 bits de resolución cada uno, es decir, la información recibida es tratada en un rango de 0 a 1023. Los puertos A4 (SDA) y A5 (SCL) están adaptados para soportar comunicación mediante bus I^2C.

3.1.3. *Herramienta de programación*

La programación de las placas Arduino se realiza a través de su IDE (Integrated Development Enviroment), un programa que contiene un editor de código, un compilador, un depurador y una interfaz gráfica. [12]

Al tratarse Arduino de una plataforma *open source*, este IDE no tiene licencia y, por tanto, es gratuito. Se puede programar en varios lenguajes aunque el más utilizado es el código C, que será el utilizado en este trabajo.

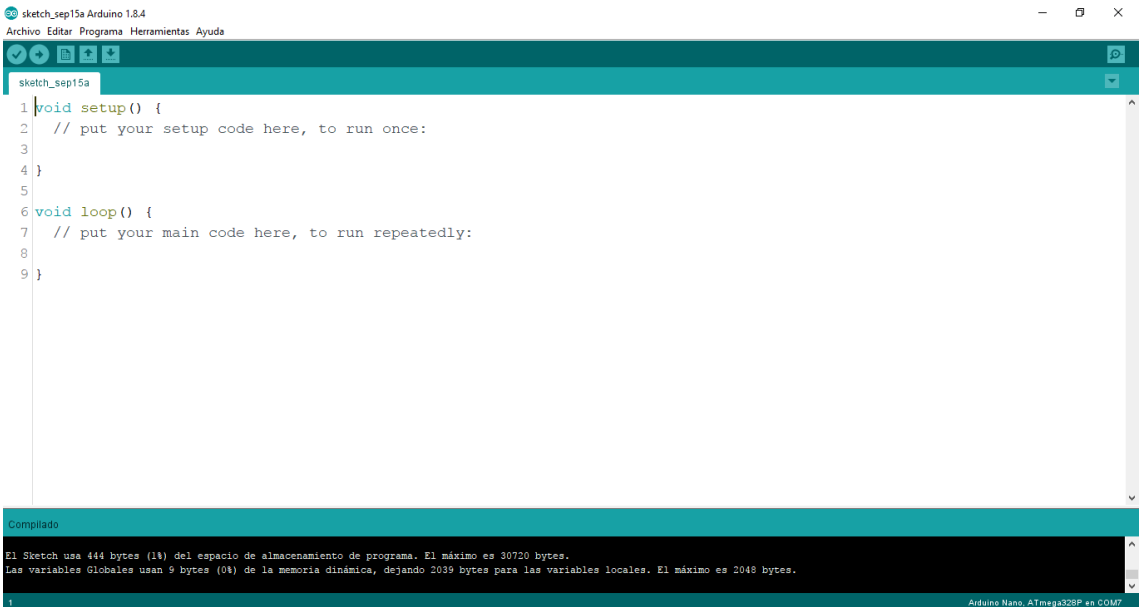


Figura 30. IDE Arduino

3.2. Sensores

3.2.1. *CNY70*

Es un sensor reflexivo que incluye dentro de su encapsulado un diodo emisor de infrarrojos y un fototransistor orientados en la misma dirección (Figura 31). De esta manera, cuando el diodo emita un haz de luz infrarroja sobre una superficie, se reflejará parte de esa luz y será detectada por el fototransistor.

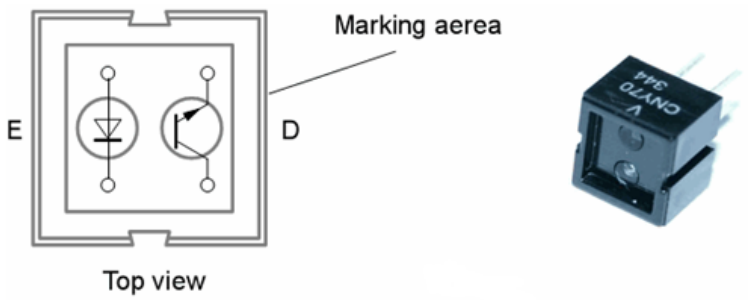


Figura 31. Sensor reflexivo CNY70 [37]

Estos sensores son utilizados para medir la cantidad de luz infrarroja reflejada por una superficie. Se puede utilizar para diferenciar entre blanco y negro debido a que el color negro absorbe la mayor parte de luz recibida mientras que, el color blanco es totalmente al contrario, refleja la mayor parte de luz recibida.

3.2.2. *GPD2D12*

Es un sensor reflexivo que está compuesto por un diodo emisor de luz infrarroja, un detector sensible a la posición (PSD) y un circuito de procesamiento de señales.

Figura 32. Sensor reflexivo GP2D12 [38]

Este sensor se utiliza para la medición de distancia a la que se encuentra un objeto. El sistema que utiliza para calcular la distancia a un objeto, se basa en el principio de triangulación. Este principio, consiste en que el haz de luz infrarroja emitido por el diodo, al reflejar en una superficie y regresar al PSD, genera un triángulo que en función de la distancia a la que se encuentre la superficie tiene ángulos diferentes [38].

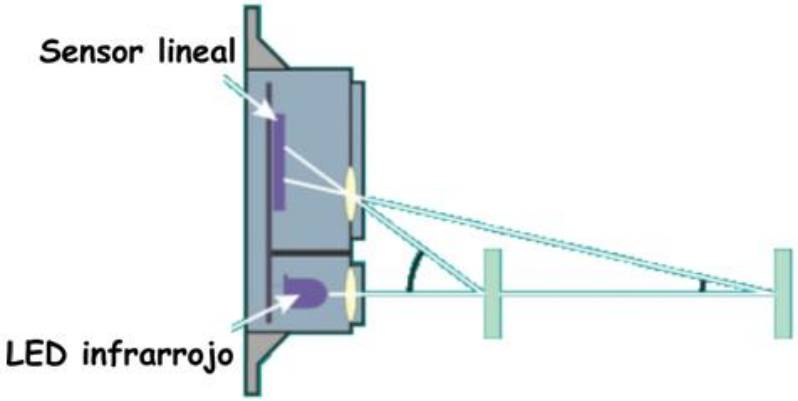


Figura 33. Triangulación sensor GP2D12 [38]

Como se puede observar en la figura 33, cuando un objeto se encuentra cerca, el haz de luz infrarroja incide en el sensor con un ángulo mayor que cuando el objeto se encuentra alejado. Por esta razón, estos sensores no son capaces de medir distancias largas con precisión ya que el ángulo de incidencia apenas varía.

3.2.3. *TCST1103*

Es un sensor óptico de transmisión compuesto por un diodo emisor de luz infrarroja y un fototransistor, emplazados dentro del mismo encapsulado y situados uno en frente del otro (Figura 34).

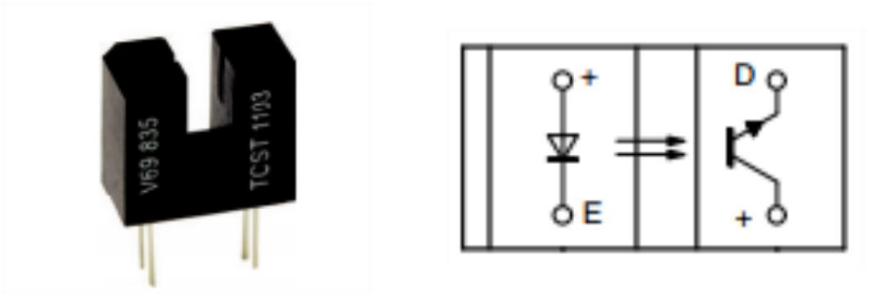


Figura 34. Sensor de transmisión TCST1103 [39]

El funcionamiento de este sensor se basa en la recepción o no de luz, emitida por el diodo en el fototransistor. Si existe algún objeto que interrumpa la visión directa entre el diodo y el fototransistor, este último no conducirá, mientras que si no existe nada que impida que la luz del diodo llegue al fototransistor, este último comenzará a conducir.

3.2.4. *Nunchuck Wii*

Es una extensión del mando inalámbrico de la videoconsola de Nintendo denominada Wii. Contiene 2 botones, un joystick bidireccional, un acelerómetro de tres ejes y un microcontrolador que se encarga de tratar la información suministrada por los sensores anteriormente mencionados, y comunicarse con otro microcontrolador, mediante bus I^2C, para enviar la información obtenida.

Figura 35. Nunchuck Wii [40]

El Nunchuck envía los datos en paquetes de 6 bytes como se muestra en la siguiente figura:

Data byte receive								Address
Joystick X								0x00
Joystick Y								0x01
Accelerometer X (bit 9 to bit 2 for 10-bit resolution)								0x02
Accelerometer Y (bit 9 to bit 2 for 10-bit resolution)								0x03
Accelerometer Z (bit 9 to bit 2 for 10-bit resolution)								0x04
Accel. Z bit 1	Accel. Z bit 0	Accel. Y bit 1	Accel. Y bit 0	Accel. X bit 1	Accel. X bit 0	C-button	Z-button	0x05

Figura 36. Bytes de datos Nunchuck [41]

Los dos primeros bytes, almacenan los valores de los ejes X e Y del joystick respectivamente, en un rango de 0 a 255. Los tres siguientes bytes, contienen los 8 bits de mayor peso de los ejes X, Y y Z del acelerómetro que trabaja con un rango de 0 a 1023. Y el último byte, contiene los dos bits restantes de cada eje del acelerómetro y los dos bits que indican el estado de los dos botones del mando [41].

3.3. Actuadores

3.3.1. *Motores de corriente continua*

Para la realización de este proyecto se han utilizado tres motores de corriente continua, dos de ellos forman parte del conjunto motriz y el motor restante es un ventilador.

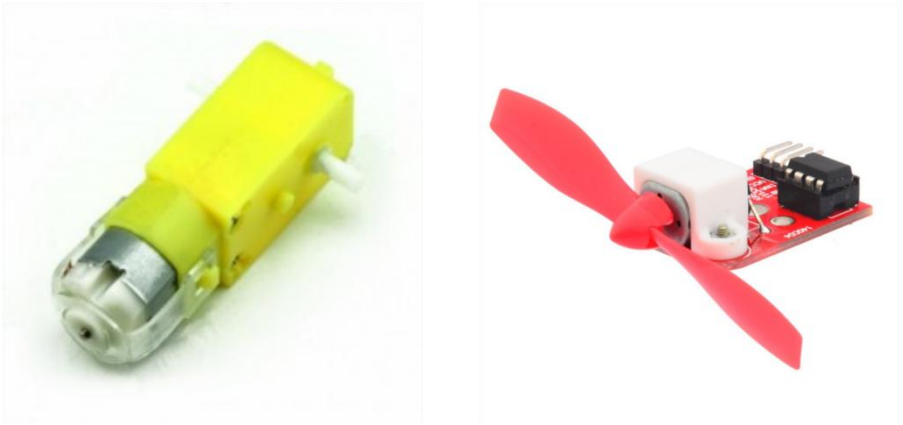


Figura 37. Motores DC [42] [43]

Los motores utilizados para dotar de movimiento al robot están compuestos por un motor de corriente continua y un tren de engranajes compuesto, que sirve para reducir la velocidad del motor y aumentar el par (Figura 38).

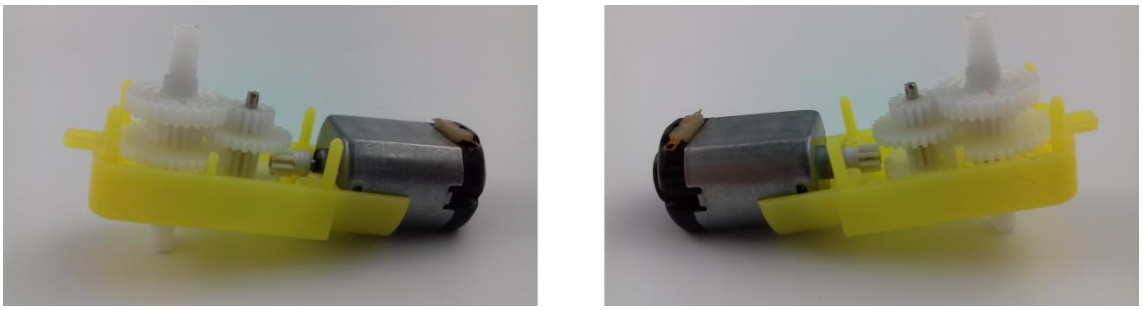


Figura 38. Reductora motor DC

La relación entrada-salida del motor se detalla a continuación en la siguiente tabla:

Etapa (engranajes)	Número de dientes	Reducción
1	8:26	4:13
2	9:36	1:4
3	16:28	4:7
4	14:30	7:15
Total		4:195 ≈ 1:48

Tabla 5. Relación entrada-salida motor DC

Estos motores tienen una velocidad angular en su doble eje de salida de 42 rpm aproximadamente, una velocidad suficiente para la funcionalidad que van a tener.
El ventilador, es el módulo Keyes L9110 Fan, que incluye un motor de corriente continua y el driver de motores L9110 capaz de suministrar 800 mA de corriente.

3.3.2. *Servomotor*

Es un servomotor de radiocontrol de tipo S3003 cuyas características se muestran en la siguiente tabla:

Velocidad	0.23seg/60 grados
Par	3.2 Kg-cm
Frecuencia (PWM)	50 Hz
Rango de giro	180 grados
Tensión de alimentación	4.8 – 6 V

Tabla 6. Parámetros servomotor S3003

El control de posición del servomotor se realiza mediante una señal PWM de 50 Hz de frecuencia (20 ms de periodo). El tiempo en alto de dicha señal, es el que determina la posición que tomará el servomotor (Figura 39).

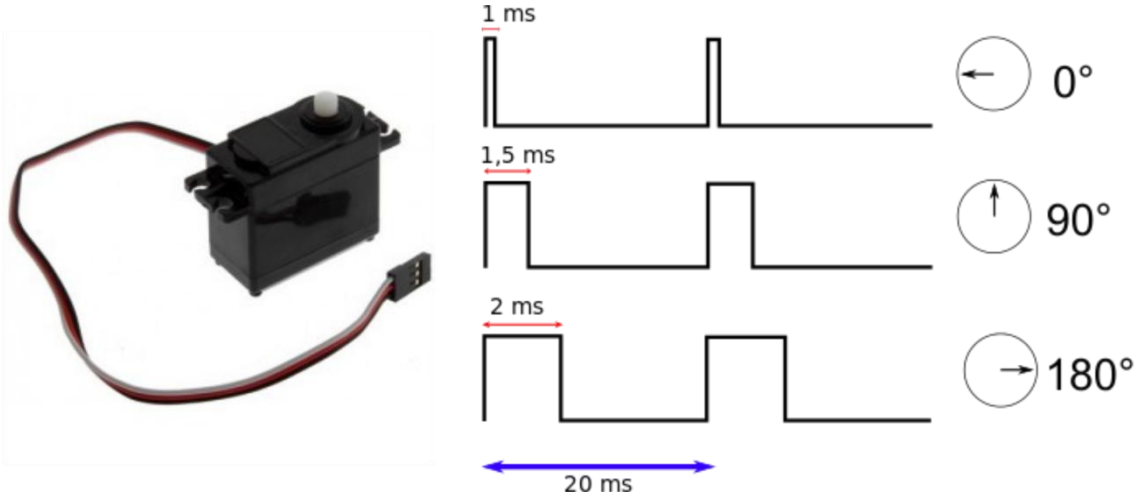


Figura 39. Señales control servomotor [44] [45]

4. Acondicionamiento de sensores y actuadores comerciales para el NXT

4.1. Comunicación entre el *brick* NXT y la placa Arduino Nano

Como se ha explicado con anterioridad, el *brick* NXT tiene un puerto de comunicación serie de alta velocidad que utiliza la norma RS-485, y la placa Arduino Nano tiene otro puerto de comunicación serie que trabaja con señales TTL. Esta diferencia implica que cada puerto de comunicación tiene distintos valores de tensión para definir los niveles lógicos de las señales con las que se comunicarán, y será necesario realizar una adaptación entre ambos puertos.

4.1.1. *Señales TTL*

Los niveles de tensión de las señales TTL se encuentra entre 4.75 V y 5.25 V aunque normalmente son 5 V. Los valores de tensión que definen los niveles lógicos de estas señales se pueden visualizar en la siguiente figura.

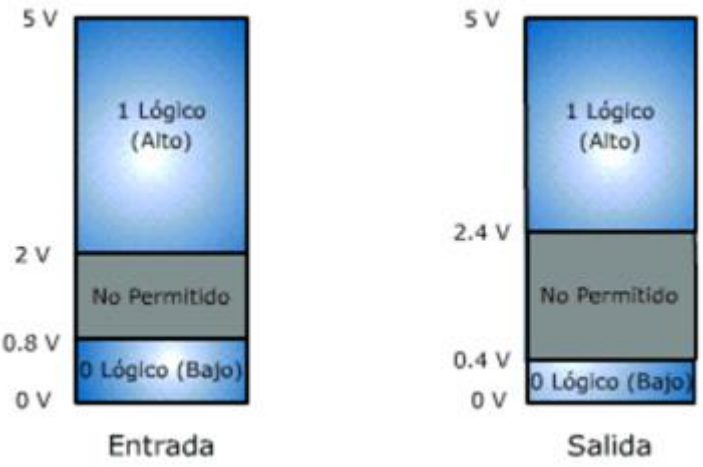


Figura 40. Niveles de tensión señales TTL [46]

4.1.2. *Norma RS-485*

La norma RS-485 se basa en un sistema diferencial para definir los niveles lógicos, es decir, el nivel lógico de la señal viene determinado por la diferencia de tensión que existe entre sus dos líneas (A y B). Si la tensión en A, es mayor que la tensión en B al menos 0.2 V (Figura 42), se tratará de un nivel alto. Por el contrario, si la tensión en B, es mayor que la tensión en A al menos 0.2 V, se tratará de un nivel bajo [19].

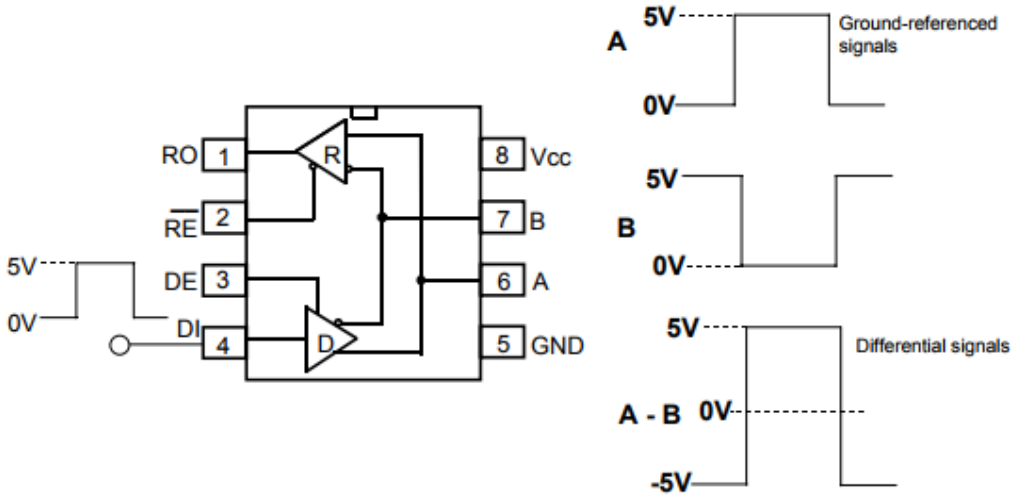


Figura 41. Circuito MAX485

INPUTS			OUTPUTS	
$\overline{RE}$	DE	DI	Z	Y
X	1	1	0	1
X	1	0	1	0
0	0	X	High-Z	High-Z
1	0	X	High-Z*	High-Z*

INPUTS			OUTPUT
$\overline{RE}$	DE	A-B	RO
0	0	≥ +0.2V	1
0	0	≤ -0.2V	0
0	0	Inputs open	1
1	0	X	High-Z*

Figura 42. Niveles lógicos norma RS-485

4.1.3. *Adaptación de señales*

La diferencia existente entre los niveles lógicos de tensión de ambos puertos, hace imposible una correcta comunicación entre ellos. Para solucionar este problema se ha utilizado el circuito integrado MAX485, que será el encargado de realizar las conversiones de niveles lógicos entre la placa Arduino y el *brick* NXT.

Con el objetivo de tener una interfaz sencilla de utilizar, se ha diseñado una placa de circuito impreso (PCB) que incluye la placa Arduino Nano, el circuito integrado MAX485 y el conector RJ12 hembra que utiliza el *brick* para sus conexiones (Figura 43).

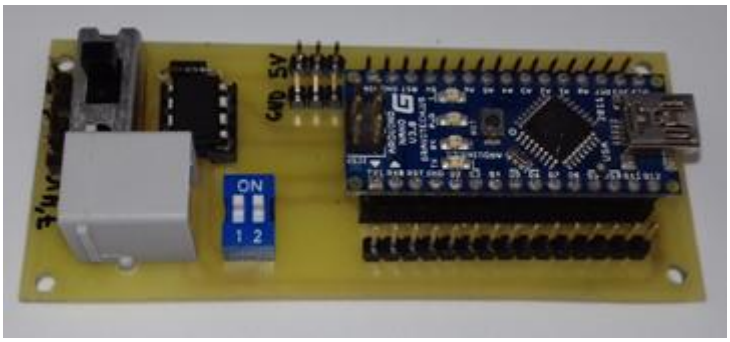


Figura 43. Interfaz comunicación NXT-Arduino

En el PCB están conectados internamente los pines digitales 0 (Rx) y 1 (Tx) de la placa Arduino, con los pines de recepción (RO) y transmisión (DI) del circuito integrado MAX485.

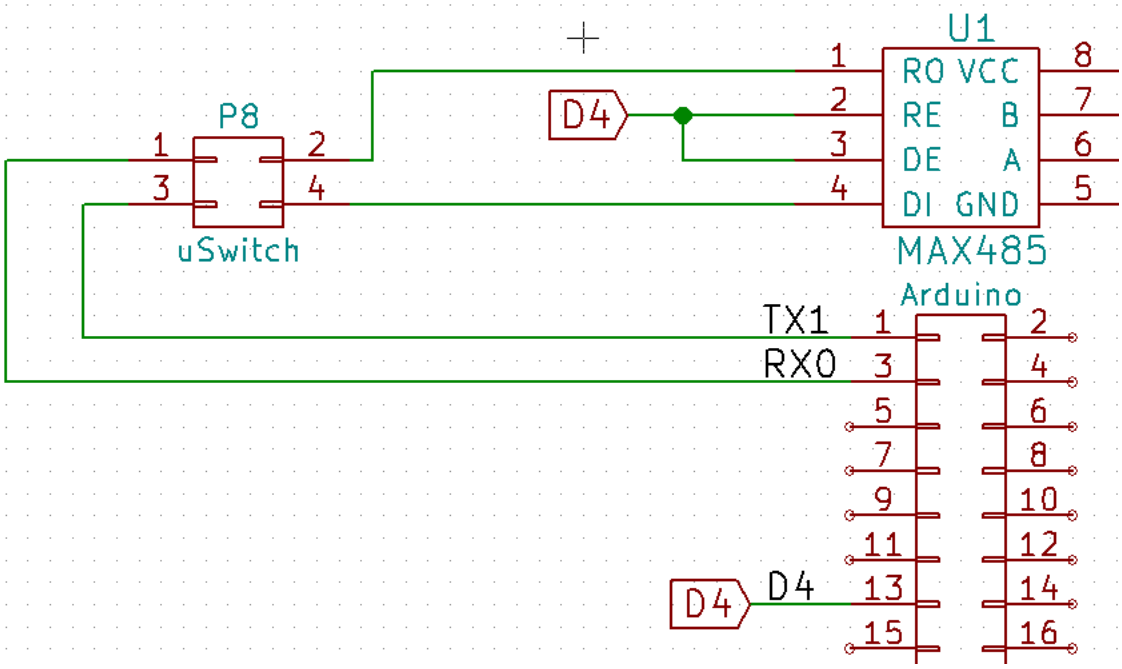


Figura 44. Circuito interfaz comunicación NXT-Arduino

Como se puede observar en la figura 44, entre estas conexiones hay situado un microswitch doble, cuya función es habilitar o inhabilitar dichas conexiones. Este componente es necesario ya que los pines digitales 0 y 1 de Arduino también forman parte del sistema de comunicación con el ordenador, y en caso de estar las dos comunicaciones habilitadas a la vez, se producirían colisiones y, por lo tanto, errores en la comunicación. También se puede ver que el pin digital 4 de la placa Arduino Nano está conectado a los pines del circuito integrado MAX485, que controlan los modos recepción o transmisión del circuito, RE Y DE. Como añadido el PCB contiene un interruptor para activar la alimentación cuando se necesite.

4.2. Sensores

En este apartado de presentan los circuitos desarrollados para adaptar sensores comerciales comúnmente utilizados en robótica con Arduino, para ser utilizados con el *brick* NXT.

4.2.1. *CNY70*

Como ya se mencionó anteriormente, se trata de un sensor que está compuesto por un diodo emisor de infrarrojos y un fototransistor, que tienen que ser polarizados correctamente. Para ello se ha diseñado un PCB que contiene el propio sensor y varias resistencias.

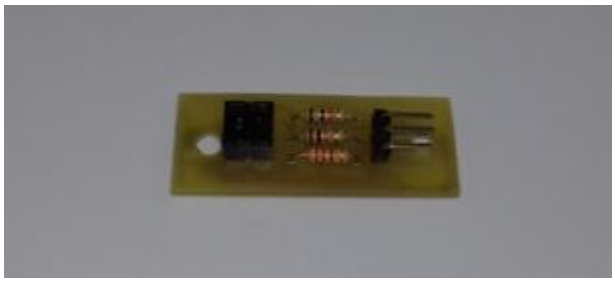


Figura 45. PCB adaptador CNY70

Observando la figura 46 se puede ver que las resistencias R1 y R2 se utilizan para polarizar el diodo y el fototransistor respectivamente.

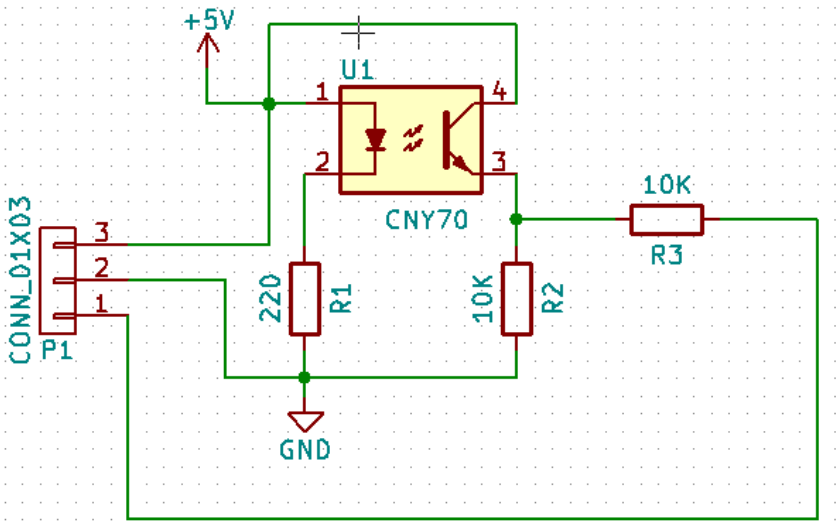


Figura 46. Circuito adaptador CNY70

Los valores de dichas resistencias han sido elegidos tomando como referencia los datos proporcionados por el fabricante. R1 vale 220 Ω, para que circule por el diodo una corriente de 15 mA aproximadamente y R2 vale 10K, para que la corriente de colector del fototransistor, cuando entre en conducción, sea pequeña y no se queme.

Con esta configuración, cuando el sensor esté situado sobre una superficie negra, reflejará poca luz infrarroja y el transistor no entrará en conducción, por tanto, la tensión será de 0 V en el sensor. Mientras que cuando esté situado sobre una superficie blanca, reflejará la mayor parte de luz infrarroja y el transistor entrará en conducción, por lo que a la salida la tensión será próxima a 5 V.

Para explicar R3 hay que recordar cómo está diseñado electrónicamente el puerto de entrada analógico del *brick* NXT.

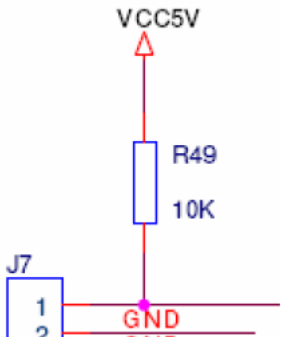


Figura 47. Entrada analógica puerto NXT

Como se observa en la figura 47, conectado al pin de entrada analógico hay una resistencia de pull-up de 10 KΩ, que va a influir a la hora de introducir valores analógicos en dicho pin. Para poder tener una lectura estable en el pin analógico se ha situado una resistencia de 10 KΩ (R3) en serie a la salida del sensor, de esta forma tendremos una configuración divisor de tensión. Esta configuración hará, que cuando el sensor esté sobre negro la tensión será cercana a 2.5 V y cuando esté sobre blanco será aproximadamente 0 V. Una diferencia considerable para poder distinguir ambos colores.

4.2.2. *GP2D12*

Se trata de un sensor que no es compatible con la configuración de entrada analógica del *brick* NXT si se conecta directamente, debido a la resistencia de pull-up que tiene internamente el *brick*.

En la siguiente figura se puede ver cómo afecta a la tensión de salida del sensor distintos valores de resistencia de pull-up.

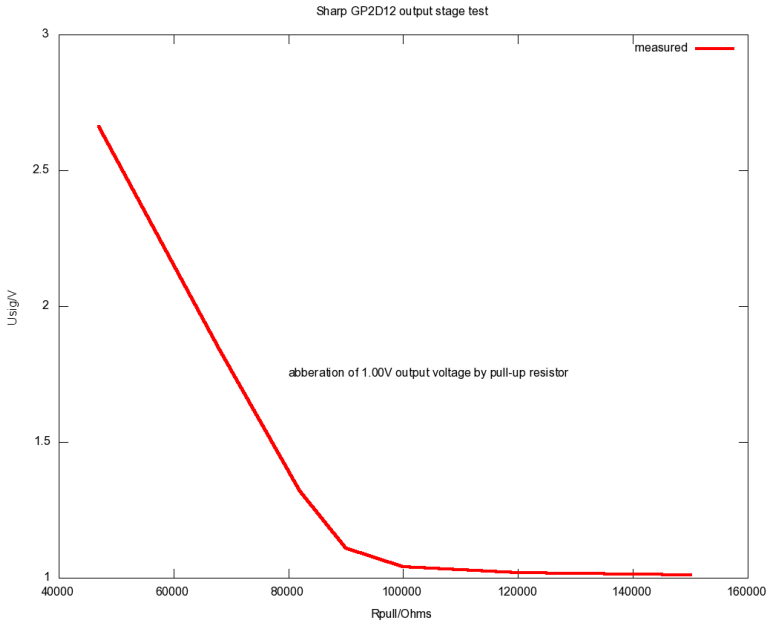


Figura 48. Relación lectura analógica frente a resistencia pull-up GP2D12 [48]

El estudio está hecho para obtener una salida de 1V. De la gráfica se deduce que para valores mayores de 100 KΩ en la resistencia de pull-up, apenas se ve afectada la salida. El problema se encuentra cuando utilizamos valores menores de 100 KΩ, ya que la tensión de salida se dispara, de forma lineal, según se reduce la resistencia de pull-up. Esto indica que se está comportando como una fuente de corriente y al calcular la pendiente de la recta, se obtiene que la corriente que está suministrando es de aproximadamente 39 µA. Debido a que la resistencia interna del *brick* NXT es de 10 KΩ es necesario montar un transistor con la configuración colector común [49], también conocida como seguidor emisor, ya que la resistencia pasará a formar parte de la configuración del transistor y a la salida de éste, tendremos la misma tensión que está entregando la salida del sensor GP2D12 (Figura 49).

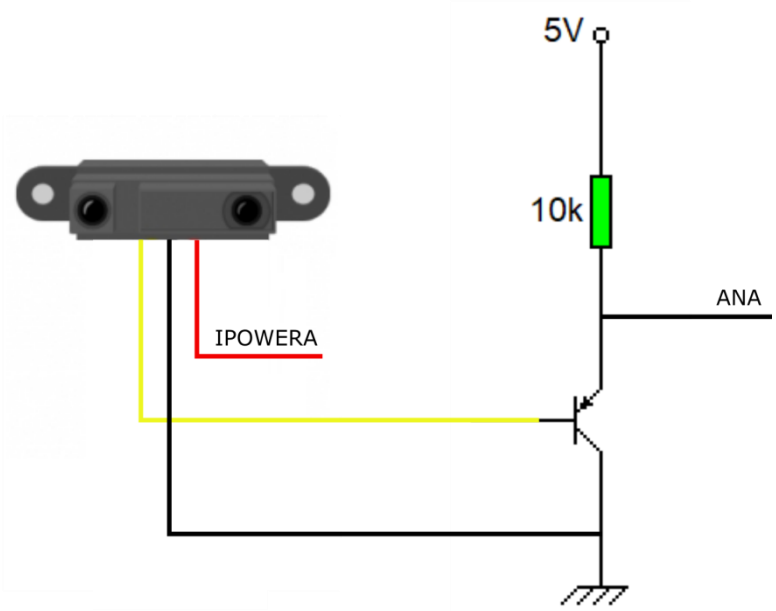


Figura 49. Circuito acondicionamiento GP2D12

4.2.3. *TCST1103*

Como se comentó anteriormente, está compuesto por un diodo emisor de infrarrojos y un fototransistor, que deben ser polarizados para su correcto funcionamiento. Por ello se ha diseñado un PCB que incluye las resistencias de polarización y un circuito de acondicionamiento para tener una señal digital pura a la salida.

Figura 50. PCB adaptador TCST1103

En la figura 51 se puede observar que R1, es la resistencia que polariza el led emisor de infrarrojos y R2, la resistencia que polariza el fototransistor.

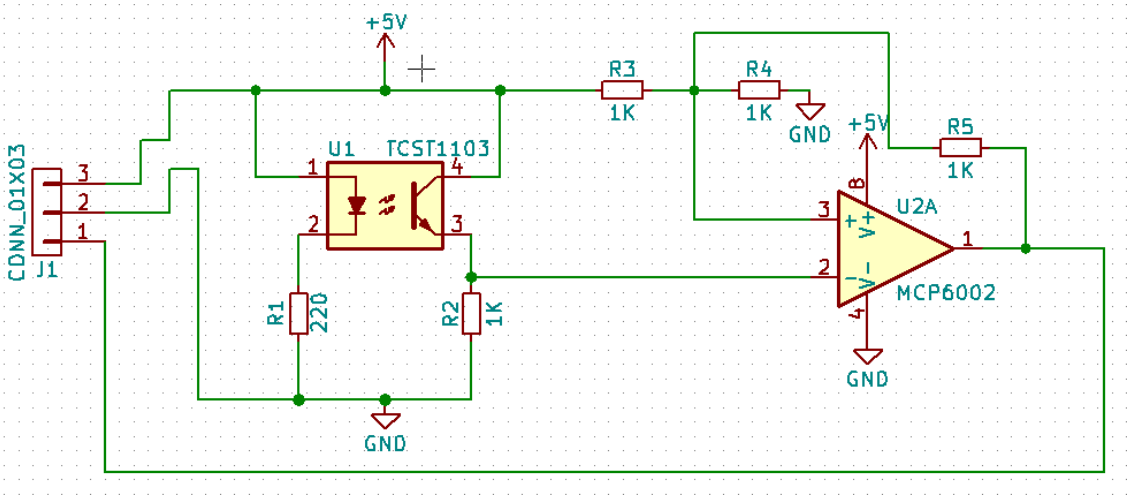


Figura 51. Circuito adaptador TCST1103

La señal de salida del sensor TCST1103 es analógica y, por tanto, existen tiempos de transición entre estados que pueden llevar a dar medidas erróneas. Para solventar este inconveniente se ha utilizado un amplificador operacional "rail to rail", configurado como disparador de Schmitt, que permitirá obtener una señal digital pura [50]. En la Figura 52, la señal amarilla corresponde a la salida del TCST1103 y la señal azul a la salida del circuito adaptador con disparador Schmitt.

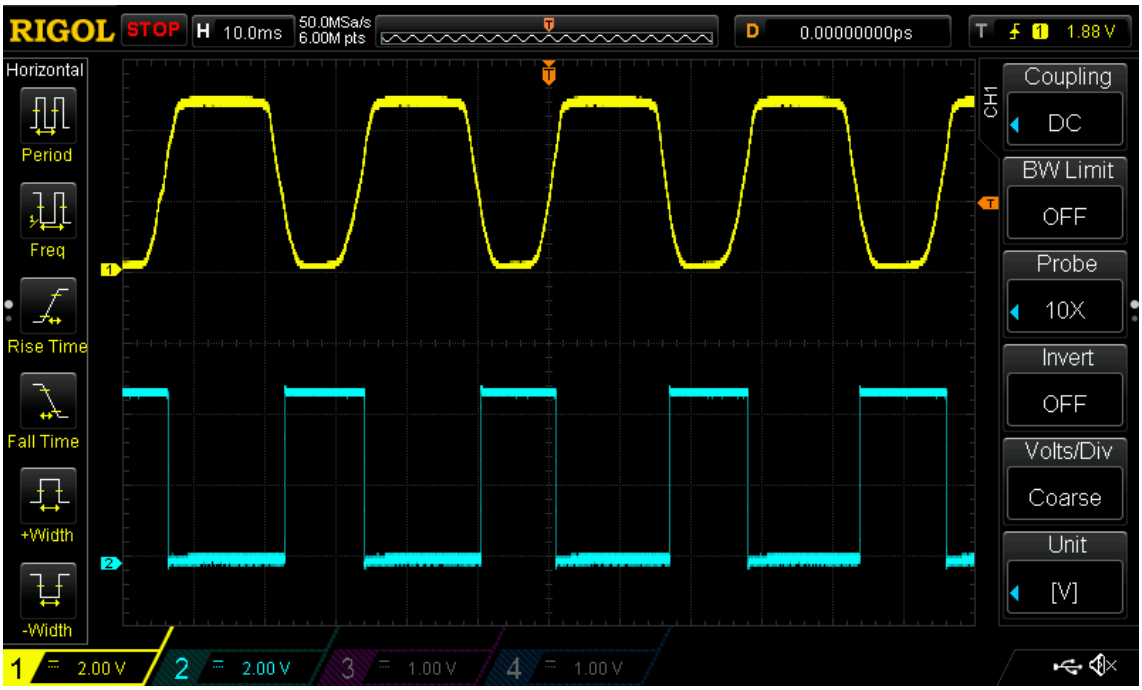


Figura 52. Señales de entada y salida del Schmitt trigger

4.3. Actuadores

El *brick* NXT contiene dos circuitos integrados que son capaces de controlar hasta tres motores de corriente continua. Por esta razón, no se necesita ningún acondicionamiento para que los motores funcionen correctamente cuando se conecten al *brick*.

Para facilitar las conexiones entre los motores que no son de Lego y el *brick*, se ha diseñado un PCB en el que se interconectan los conectores RJ12 de LEGO Mindstorms con tiras de seis pines macho, con el fin de poder conectar los cables que proceden de los motores (Figura 53).

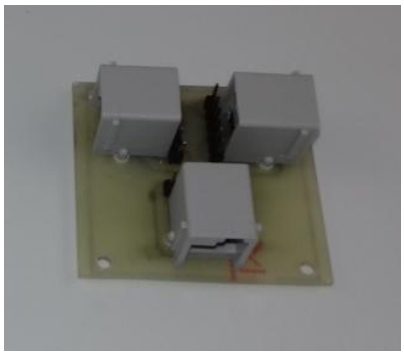


Figura 53. PCB adaptador conectores LEGO

En el caso del servomotor, hay que tener en cuenta dos parámetros importantes para su funcionamiento, la tensión y la corriente. El servomotor va a ser controlado por la placa Arduino Nano que, como se vio en capítulos anteriores, alimenta con una tensión de 5 V (tensión suficiente para el servomotor) pero solo es capaz de suministrar una corriente máxima de 40 mA (corriente insuficiente para poder controlar un servomotor). La solución elegida ha sido alimentar el servomotor con las baterías de ion-litio (7.4 V), ya que serán capaces de suministrar la corriente demandada, pero es necesario reducir la tensión de alimentación debido a que el servomotor trabaja en un rango de tensiones de 4.8 V a 6 V. Para ello se han introducido tres diodos 1N4007 en serie, que supondrán una caída de tensión que oscilará entre de 2.1 y 2.4 V en función de la corriente que demande el servomotor, reduciendo así la tensión de alimentación del servomotor como máximo hasta 5 V (Figura 54).

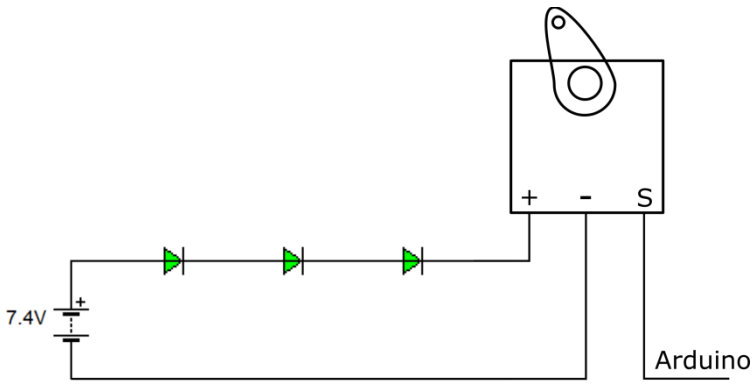


Figura 54. Circuito acondicionamiento servomotor

5. Acondicionamiento de sensores y actuadores LEGO

El control de los sensores y actuadores del kit de LEGO Mindstorms NXT, se hace mediante un sistema embebido diseñado especialmente para esa funcionalidad. Este sistema tiene capacidad para controlar todos los sensores del kit además de dos servomotores.

5.1. Sistema embebido

El núcleo de este sistema es el microcontrolador ATMEGA328P que está incluido en la placa Arduino Nano. Este microcontrolador será el encargado de procesar la información enviada por los cuatro tipos de sensores que incluye el kit. También se encargará de generar las señales PWM con las que se controlará el movimiento de los motores.

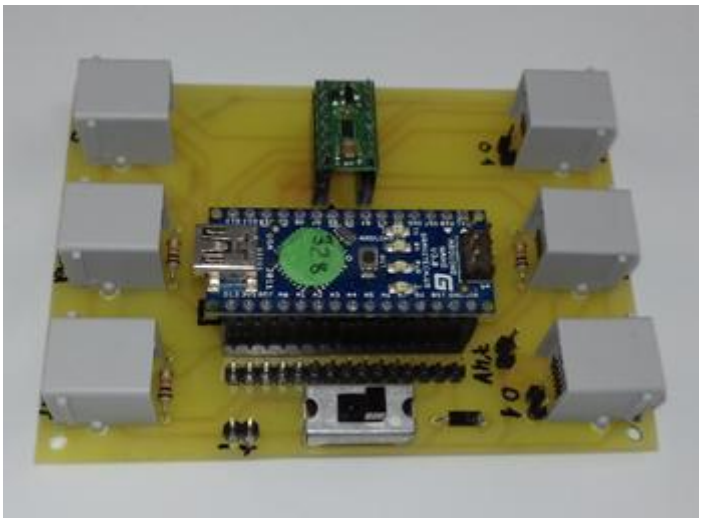


Figura 55. PCB sistema embebido

Como se puede observar en la figura 55, el sistema embedido incluye los conectores RJ12 de LEGO para conectar de una forma rápida y sencilla tanto los sensores, como los actuadores.

5.2. Sensores LEGO Mindstorms NXT

En este apartado se van ver las conexiones entre los sensores del kit LEGO Mindstorms NXT, explicados en apartados anteriores, con la placa Arduino.

5.2.1. *Sensor de contacto*

Recordando su descripción en el apartado IV.2.3.2.1, está compuesto por un switch en serie con una resistencia, por lo que es necesario añadir una resistencia de pull-up al circuito para que se pueda diferenciar entre dos niveles lógicos (Figura 56).

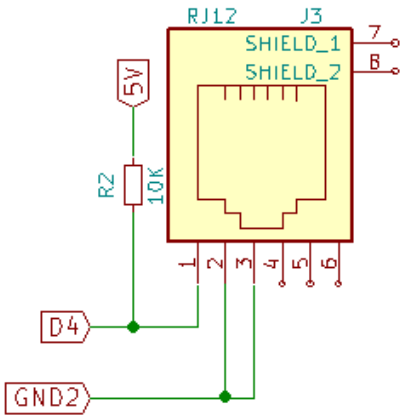


Figura 56. Conector sensor de conctacto NXT

Con esta configuración, cuando el *switch* esté sin presionar, habrá un nivel alto debido a la tensión a la que está conectada la resistencia de *pull-up*. Cuando el *switch* esté presionado, se producirá un divisor de tensión formado por la resistencia de *pull-up* y la resistencia interna que incorpora el sensor, teniendo como resultado en el pin de entrada al microcontrolador, una tensión aproximada de 1 V o lo que es equivalente, un nivel bajo.

5.2.2. *Sensor de luz*

Recordando lo explicado anteriormente en el apartado IV.2.3.2.2, este sensor está compuesto por un led emisor y un fototransistor que tendrán que ser alimentados por el microcontrolador.

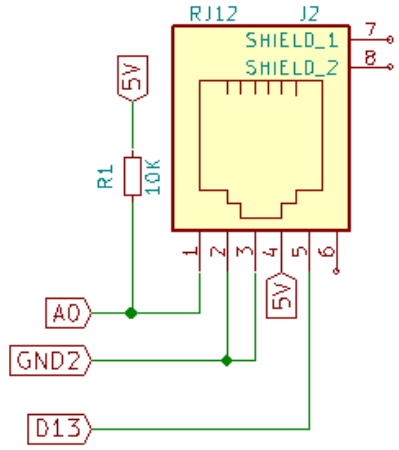


Figura 57. Conector sensor de luz NXT

El led puede ser activado o no mediante software, por tanto, se ha conectado ese pin del sensor a una salida digital del microcontrolador. Para realizar la lectura analógica del sensor, se ha conectado el pin a una entrada analógica del microcontrolador manteniendo la resistencia de 10 KΩ que contiene el *brick* NXT para obtener las mismas lecturas analógicas (Figura 57).

5.2.3. *Sensor de sonido*

Recordando su descripción en el apartado IV.2.3.2.3, está formado por un micrófono y una electrónica de acondicionamiento basada en filtros, por lo que solo es necesario alimentarlo y leer los valores que entrega.

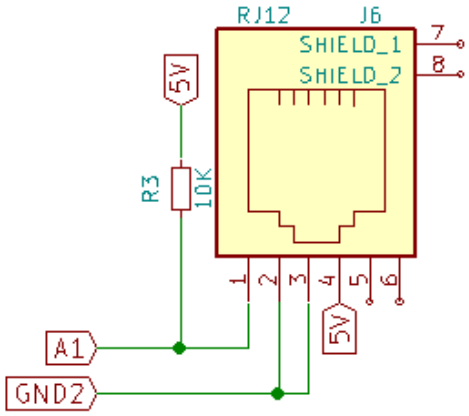


Figura 58. Conector sensor de sonido NXT

Se ha conectado el pin a una entrada analógica manteniendo, al igual que con el sensor de luz, la resistencia de 10 KΩ para respetar las lecturas analógicas que proporciona el sensor al *brick* NXT (Figura 58).

5.2.4. *Sensor de ultrasonidos*

Recordando su explicación en el apartado IV.2.3.3.2, está compuesto por dos transductores y un microcontrolador, por tanto, necesita dos tensiones de alimentación diferentes.

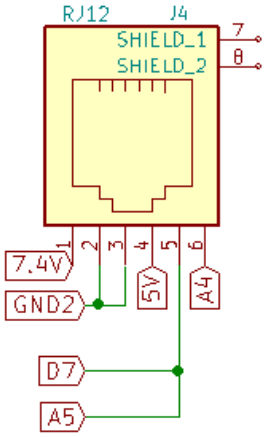


Figura 59. Conector sensor ultrasonidos NXT

Se comunica con el microcontrolador mediante bus I^2C, por tanto, es necesario utilizar dos líneas de comunicación: SDA (datos) y SCL (reloj). Además es necesario utilizar un pin digital para añadir otra señal de reloj adicional ya que LEGO utiliza un protocolo I^2C personalizado (Figura 59).

5.3. Actuadores

El microcontrolador ATMEGA328 no incluye electrónica para el control de motores, por lo que el sistema embebido lleva incorporado un driver para esta función, con capacidad para controlar dos motores de hasta 1.2 A de consumo cada uno. Se ha utilizado un módulo de Pololu basado en el circuito integrado DRV8835 que contiene dos puentes en H (Figura 60).

Figura 60. Driver DRV8835 [51]

Este driver cuenta con dos modos de funcionamiento:

- PHASE/ENABLE: solo son necesarios dos pines para controlar la velocidad y sentido de giro del motor. Con el pin denominado PHASE se puede controlar el sentido, mientras que con el pin denominado ENABLE (mediante una señal PWM) se puede controlar la velocidad de giro del motor. Como se puede observar en la siguiente figura, este modo solo permite el freno eléctrico, que se consigue al alimentar los dos bornes del motor con el mismo potencial.

Simplified drive/brake operation with MODE=1 (PHASE/ENABLE)				
xPHASE	**xENABLE**	**xOUT1**	**xOUT2**	**operating mode**
0	PWM	PWM	L	forward/brake at speed *PWM %*
1	PWM	L	PWM	reverse/brake at speed *PWM %*
X	0	L	L	brake low (outputs shorted to ground)

Figura 61. Configuración modo PHASE/ENABLE [51]

- IN/IN: también utiliza dos pines para controlar la velocidad y el sentido de giro del motor. En este modo se utilizan los dos pines tanto para controlar la velocidad como el sentido de giro del motor. A diferencia del modo anterior, además del freno eléctrico existe la posibilidad de frenar el motor por inercia. Esto se consigue dejando de alimentar el motor, con lo que el motor se irá frenando por la fuerza de rozamiento.

Drive/coast or drive/brake operation with MODE=0 (IN/IN)				
xIN1	**xIN2**	**xOUT1**	**xOUT2**	**operating mode**
0	0	OPEN	OPEN	coast (outputs off)
PWM	0	PWM	L	forward/coast at speed *PWM %*
0	PWM	L	PWM	reverse/coast at speed *PWM %*
1	PWM	$\overline{\text{PWM}}$	L	forward/brake at speed *100% – PWM %*
PWM	1	L	$\overline{\text{PWM}}$	reverse/brake at speed *100% – PWM %*
1	1	L	L	brake low (outputs shorted to ground)

Figura 62. Configuración modo IN/IN [51]

El sistema embebido está configurado para trabajar en el modo PHASE/ENABLE. Este modo implica utilizar dos pines de salida digitales (uno para cada motor) para controlar el sentido de los motores y otros dos pines de salida digitales, en modo PWM, para controlar la velocidad de los motores. Estas señales PWM son generadas en los pines 9 y 10 de la placa Arduino Nano que, por defecto, tienen una frecuencia de 490 Hz. Esta frecuencia es baja cuando se utiliza para controlar un motor, por ello se ha modificado mediante software para obtener una frecuencia de 3.9 kHz, una frecuencia mucho más adecuada para el control de motores.

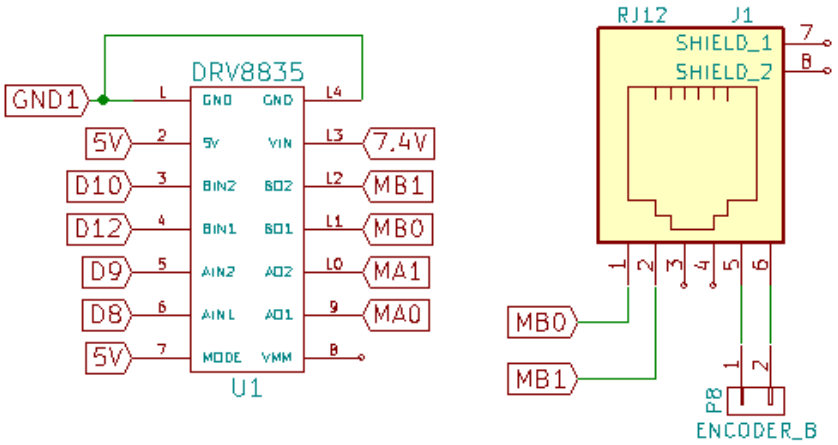


Figura 63. Conector servomotor NXT

En la figura 63 se puede observar que el conector RJ12 que conecta el motor al sistema, tiene además de las dos conexiones que permiten la rotación del motor, otras dos conexiones para poder leer los valores del encoder que lleva incorporado cada motor. Estas dos conexiones, no están unidas directamente a ningún pin del microcontrolador debido a que la forma de controlar su estado, para que sea eficiente, debe ser mediante interrupciones externas, y el microcontrolador ATMEGA328 solo cuenta con dos pines que tengan esta funcionalidad. Al ser un encoder en cuadratura, tiene dos señales por motor, por tanto, serían necesarias cuatro interrupciones externas para monitorizar todas las señales que provienen de los encoder. Al existir la

posibilidad de monitorizar solo dos señales, se deja a elección del usuario si quiere controlar las dos señales de un mismo servomotor para poder controlar la velocidad y el sentido de giro, o si por el contrario desea controlar una señal de cada servomotor y así regular la velocidad de ambos motores con el objetivo de conseguir que el robot avance recto.

6. Robot A

Una vez finalizado el estudio de las plataformas LEGO Mindstorms NXT y Arduino, junto con los sensores y actuadores comerciales, se ha desarrollado un primer robot que tendrá como núcleo central el *brick* inteligente NXT. A este *brick*, se conectarán los sensores y actuadores estudiados anteriormente, además de la placa Arduino Nano con la que se podrán aumentar las prestaciones del robot (Figura 64).

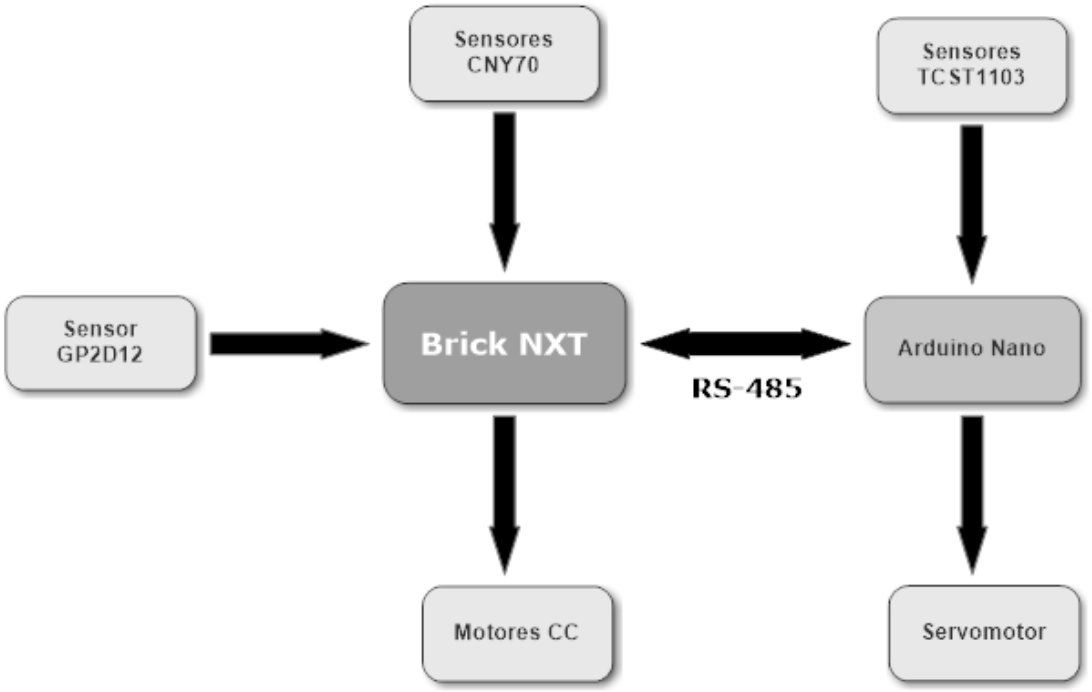


Figura 64. Esquema robot A

Este robot persigue el objetivo de, supliendo todos los periféricos que se incluyen dentro del kit LEGO Mindstorms NXT por otros de uso comercial, obtener un rendimiento similar al del kit.

6.1. Chasis

Se trata de la estructura que va a soportar todos los dispositivos que integre el robot. Para su diseño es necesario tener en cuenta varios factores como el tamaño, el peso y su distribución, el material de fabricación, la separación entre las ruedas, etc.

El material utilizado para el desarrollo del chasis del robot ha sido DM (densidad media), un material derivado de la madera que está compuesto por fibras de madera aglutinadas con resinas sintéticas, lo que le permite tener una mayor densidad que otro tipo de aglomerados [52].

La técnica de corte utilizada para la creación del chasis ha sido el corte por láser, una técnica que se basa en la focalización de un haz de láser sobre un punto del material que se desea tratar, para que se funda o evapore. Se trata de una técnica muy ventajosa debido a su alta velocidad y precisión. Para poder cortar una pieza es

necesario crear un archivo vectorial 2D que será el que envíe la instrucciones de corte a la cortadora [53].

En la Figura 65 puede verse el diseño vectorial en 2D de la estructura del chasis. Las dos piezas superiores son las dos plantas que tiene el robot, estando situada a la izquierda la planta superior y a la derecha la planta inferior. Las piezas restantes forman parte de la sujeción de los motores (4 piezas en forma de T) y del soporte del sensor de infrarrojos GP2D12.

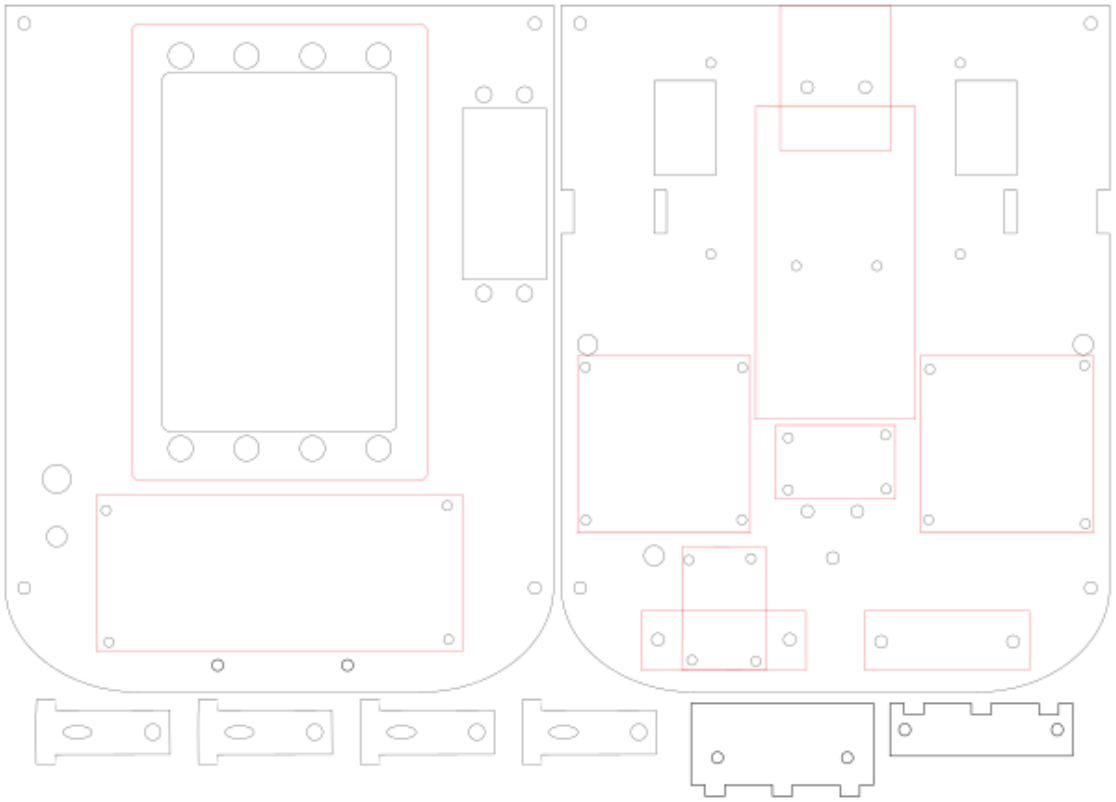


Figura 65. Diseño chasis robot A

6.2. Robot

Estructurado en dos plantas para hacerlo más compacto y robusto, el robot contiene varios periféricos que se conectarán al *brick* NXT.

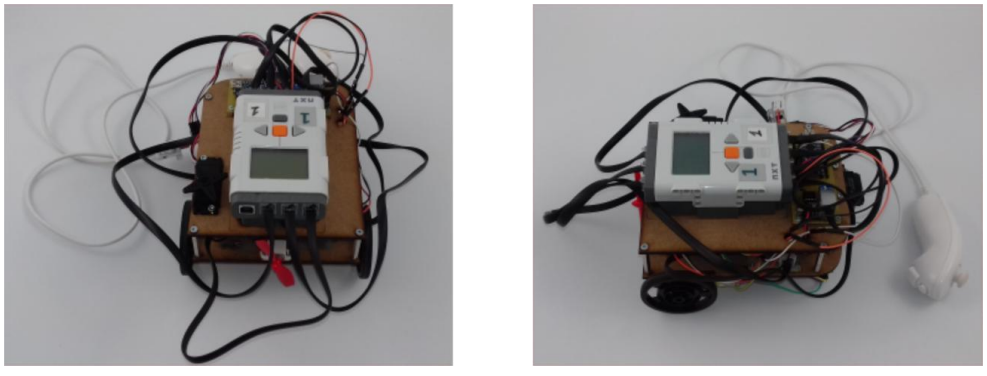


Figura 66. Robot A

Como ya se comentó con anterioridad, el *brick* NXT sólo permite el conexionado de tres actuadores y cuatro sensores, un número en ocasiones limitado para ciertas aplicaciones que se puedan necesitar. Para solucionar este problema y ampliar esa cantidad de entradas y salidas, el *brick* está conectado a la tarjeta Arduino Nano, con la que intercambiará información mediante su puerto de alta velocidad, permitiendo así conectar más dispositivos al robot aumentando sus prestaciones.

El *brick* NXT lleva conectados directamente dos sensores reflexivos CNY70 adosados en la parte inferior de la primera planta (Figura 67), enfocando al suelo, con los que podrá diferenciar entre distintos colores que haya en la superficie donde se encuentre.

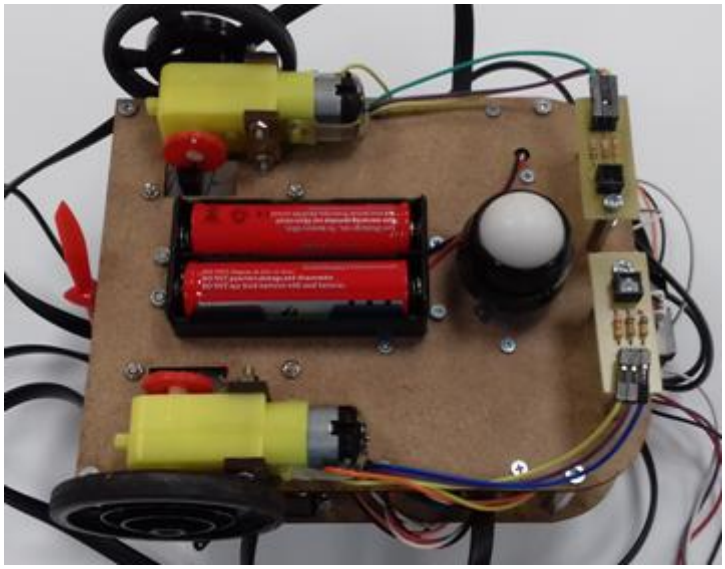


Figura 67. Cara inferior robot A

También cuenta con un sensor reflexivo GP2D12, junto con su circuito de acondicionamiento, situado en la parte frontal del robot, con el que podrá detectar objetos que se encuentren a su alrededor.

Dentro del apartado de actuadores conectados al *brick*, el robot contiene dos motores de corriente continua que se encargarán de dotar de movimiento al robot y otro motor de corriente continua, con una hélice incorporada que puede ser utilizada para mover objetos de peso liviano.

Es importante recordar que los servomotores del kit permiten tener un control preciso del movimiento gracias a los encoder ópticos que contienen en su interior. Los motores de corriente continua utilizados en el robot, no tienen esa funcionalidad incorporada, pero sí tienen un doble eje de salida donde además de la rueda se puede conectar un disco codificador. Para poder leer los datos del disco codificador de cada motor se utilizan los sensores ópticos de transmisión TCST1103 (uno por cada motor), con los que se pueden controlar la velocidad de rotación y la distancia avanzada por el motor, que irán conectados a la placa Arduino Nano debido a la ocupación total de los puertos disponibles en el *brick* NXT.

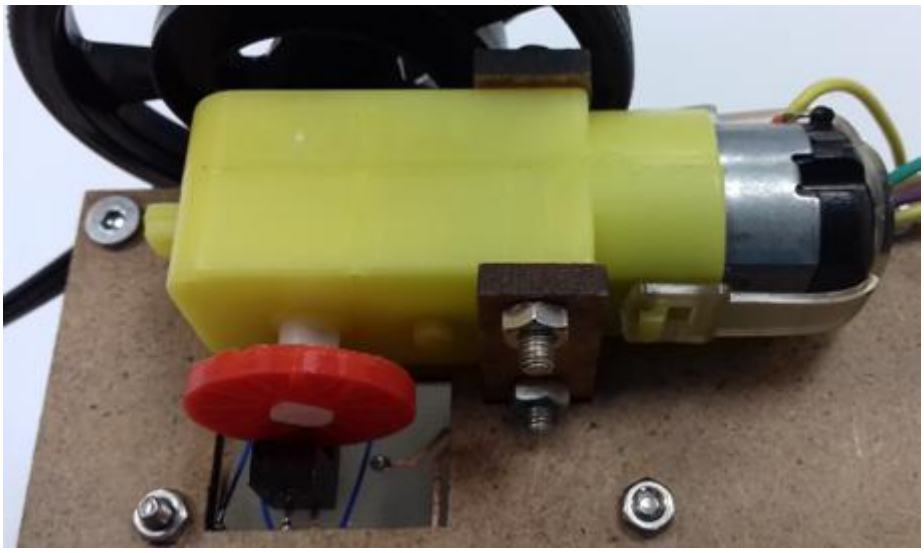


Figura 68. Encoder motor robot A

En el caso de este robot se ha utilizado un disco codificador de 12 ranuras, con el que se tendrá una precisión de 15 grados (Figura 68).

El robot también incluye la posibilidad de utilizar un servomotor que será controlado por la placa Arduino.

Como añadido, existe la posibilidad de controlar el robot con el mando de la Wii llamado Nunchuck, ya que la placa Arduino Nano cuenta con un bus I^2C. Este control manual supondría la pérdida de la denominación robot debido a que ya no tomaría decisiones de forma autónoma, si no que se limitaría a recibir órdenes y cumplirlas.

En la tablas 7 y 8 se detallan las conexiones del robot y de la placa Arduino Nano.

Dispositivo	NXT
Ventilador	Puerto A
Motor derecho	Puerto B
Motor izquierdo	Puerto C
Sensor reflexivo derecho (CNY70)	Puerto 1
Sensor reflexivo izquierdo (CNY70)	Puerto 2
Sensor de distancia (GP2D12)	Puerto 3
Placa Arduino Nano	Puerto 4

Tabla 7. Conexiones de sensores y actuadores con el *brick*

Dispositivo	Arduino Nano
Encoder derecho (TCST1103)	Entrada digital D2 (interrupción)
Encoder izquierdo (TCST1103)	Entrada digital D3 (interrupción)
Servomotor	Salida digital D4
Nunchuck	Bus I^2C SDA (A4)
	Bus I^2C SCL (A5)

Tabla 8. Conexiones de sensores y actuadores con la placa Arduino Nano

6.3. Aplicaciones

Este robot está diseñado para que utilizando solo el *brick* inteligente del kit LEGO Mindstorms NXT, se puedan realizar aplicaciones similares a las que se conseguiría con el kit completo. También se abre la posibilidad de ampliar las prestaciones del *brick* utilizando la placa Arduino Nano como periférico. La programación de estas aplicaciones se realiza con la herramienta Bricx Command Center bajo lenguaje NXC cuando no se utiliza la placa Arduino. En caso de utilizar la placa Arduino, ésta se programará a través del IDE de Arduino con código C.

A continuación se presentan los ejemplos programados para cada una de las arquitecturas. Al principio de cada uno de los programas, hay una breve descripción del mismo.

6.3.1. *Robot siguelíneas*

```
/*
  Robot siguelíneas

  El robot sigue una línea negra sobre un fondo blanco con dos
  sensores reflexivos
*/
#include "NXCDefs.h"

int x,y,right_sensor,left_sensor;

task main()
{
  // Configura la medida del sensor como reflexivo
  SetSensorType(IN_1, SENSOR_TYPE_LIGHT);
  // Configura el rango de medida en porcentaje (0 - 100)
  SetSensorMode(IN_1, SENSOR_MODE_PERCENT);
  SetSensorType(IN_2, SENSOR_TYPE_LIGHT);
  SetSensorMode(IN_2, SENSOR_MODE_PERCENT);

  while(true)
  {
    y = Sensor(IN_1); // Almacena el valor del sensor en la variable y
    x = Sensor(IN_2);
    left_sensor = x*2; // Duplica el rango de valores del sensor
    right_sensor = y*2;
    NumOut(0, LCD_LINE1,right_sensor);
    NumOut(0, LCD_LINE2,left_sensor);

    if(right_sensor >= 70) // Evalua si lee negro el sensor derecho
    {
      OnFwd(OUT_B,-20);
    }
    else
    {
      OnFwd(OUT_B,50);
    }

    if(left_sensor >= 70) // Evalua si lee negro el sensor izquierdo
    {
      OnFwd(OUT_C,-20);
```

```
    }
    else
    {
      OnFwd(OUT_C,50);
    }

  }
}
```

6.3.2. *Robot detector de obstáculos*

```
/*
  Robot detector de obstáculos

  El robot detecta un obstáculo a una distancia definida mediante
  un umbral y se detiene al pasar el umbral
*/

#include "NXCDefs.h"

int distance;

task main()
{ // Configura la medida del sensor como reflexivo
  SetSensorType(IN_3, SENSOR_TYPE_LIGHT_ACTIVE);
  // Configura el rango de medida a 0 - 1023
  SetSensorMode(IN_3, SENSOR_MODE_RAW);

  while(true)
  {
    // Almacena la lectura del sensor en la variable distance
    distance = Sensor(IN_3);
    NumOut(0, LCD_LINE1,distance);

    if(infra >= 800)  //Evalúa la distancia a un objeto
    {
     OnFwd(OUT_B,50);
     OnFwd(OUT_C,50); // Avanza
    }
    else
    {
     OnFwd(OUT_B,0);
     OnFwd(OUT_C,0); // Para
    }

}
}
```

6.3.3. *Robot con control preciso de avance*

Para este programa es necesario comentar que se utiliza la placa Arduino Nano como un periférico más, con la diferencia de que en ella se pueden conectar varios dispositivos y así ampliar el número de puertos del sistema.

Como ya se ha comentado anteriormente, el *brick* y la placa de Arduino se comunican bajo la norma RS-485. Para poder mantener una correcta comunicación entre ellos, se

ha diseñado una máquina de estados (Figura 69) para cada uno debido a que tienen distinta frecuencia de trabajo (48 Mhz el *brick* y 16 Mhz la placa Arduino).

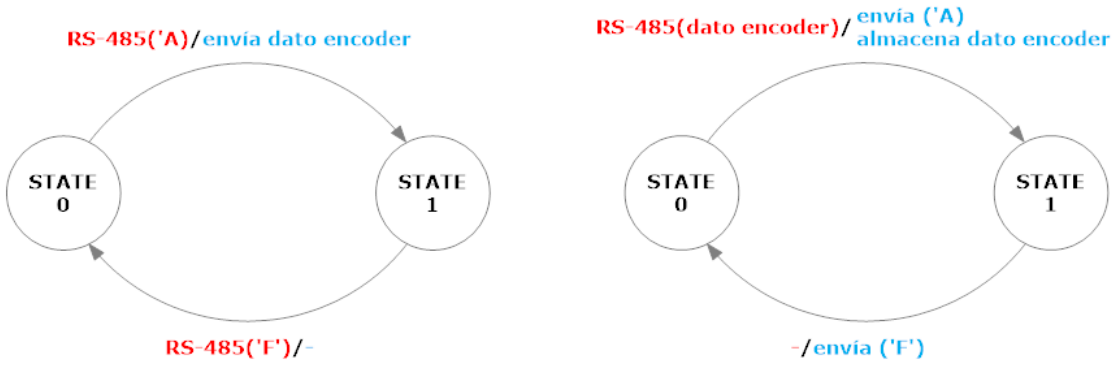


Figura 69. De izda. a dcha. grafos de máquinas de estado Arduino y NXT

```
/*
  Encoder NXT

  Programa que lee los cambios de estado del sensor TCST1103 y
  envía los datos por el puerto serie al NXT
*/

#define ENC 3
#define RS485CTRL 4
#define STATE_0 0
#define STATE_1 1
#define STATE_2 2
#define STATE_3 3
#define LED 13

byte ACK;
int fsm_state = STATE_0;
int x = 0;


void setup()
{
  // Velocidad de comunicación fijada a 115200 baudios
  Serial.begin(115200);
  // Pin asociado al control de comunicación RS-485
  pinMode(RS485CTRL, OUTPUT);
  pinMode(ENC, INPUT); // Pin asociado al sensor TCST1103
  // Configuración de interrupción
  attachInterrupt(digitalPinToInterrupt(ENC), encoder, CHANGE);
  digitalWrite(RS485CTRL, LOW); // Configuración en modo recepción

}

void loop() {
  switch (fsm_state) // Máquina de estados
  {
    case STATE_0:

      if (Serial.available())
      {
        ACK = Serial.read();
        if (ACK == 'A')
```

```
        {
          // Configuración en modo transmisión
          digitalWrite(RS485CTRL, HIGH);
          Serial.println(x);
          delay(1);
          digitalWrite(RS485CTRL, LOW);
          fsm_state = STATE_1;
        }
      }
      break;
    case STATE_1:
      if (Serial.available())
      {
        ACK = Serial.read();
        if (ACK == 'F')
        {
          fsm_state = STATE_0;
        }
      }
  }
}

void encoder() // Función de interrupción
{
  x++;
}
```

```
/*
  Encoder NXT

  Programa que recibe el valor del encoder enviado por Arduino y
  utiliza los datos para mover el robot
*/

#include "NXCDefs.h"
#define HS_BAUD_9600 5    //Velocidad comunicación 9600 bits/seg
#define HS_BAUD_57600 10    //Velocidad comunicación 57600 bits/seg
#define HS_BAUD_115200 12    //Velocidad comunicación 115200 bits/seg
#define HS_MODE_8_DATA 0x00C0 //Comunicación 8 bits
#define HS_MODE_10_STOP 0x0000 //1 bit de stop
#define HS_MODE_N_PARITY 0x0800 //Sin paridad
#define HS_MODE_8N1 (HS_MODE_8_DATA|HS_MODE_N_PARITY|HS_MODE_10_STOP)
#define STATE_0 0
#define STATE_1 1

string x;
int encoder;
int state = STATE_0;
task FSM() // máquina de estados
{
  while(true)
  {
    switch(state)
    {
      case STATE_0:
        RS485Write("A"); // Envía caracteres A por puerto serie
        until(RS485DataAvailable()); // Espera que llegue un dato
        Wait(10);
```

```
        RS485Read(x); // Lee dato
        encoder = StrToNum(x);
        TextOut(0, LCD_LINE1,x);
        state = STATE_1;
        break;
      case STATE_1:
        RS485Write("F"); // Envía carácter F por puerto serie
        Wait(1);
        state = STATE_0;
        break;
    }
  }
}

task main()
{
  UseRS485(); // Configura comunicación RS-485
  RS485Enable();
  RS485Uart(HS_BAUD_115200,HS_MODE_8N1);
  Wait(100);

  while(true)
  {
    StartTask(FSM);
    if(encoder < 300) // Evalúa el valor del encoder
    {
      OnFwd(OUT_B,57);
      OnFwd(OUT_C,55); // Avanza
    }
    else if(encoder >300 && encoder <400)
    {
      OnFwd(OUT_B,20);
      OnFwd(OUT_C,20); // Avanza lento
    }
    else
    {
      OnFwd(OUT_B,0);
      OnFwd(OUT_C,0); // Para
    }

  }
}
```

6.3.4. *Control con Nunchuck Wii*

Recordando el estudio realizado del Nunchuck, se sabe que envía los datos mediante un bus I^2C. Esta lectura de datos se la va a realizar la placa Arduino Nano que luego enviará los datos al *brick* NXT para que éste los utilice para controlar el robot.

Como el *brick* y la placa van a comunicarse para intercambiar los datos, es necesario diseñar dos máquinas de estado para sincronizar la comunicación (Figura 70).

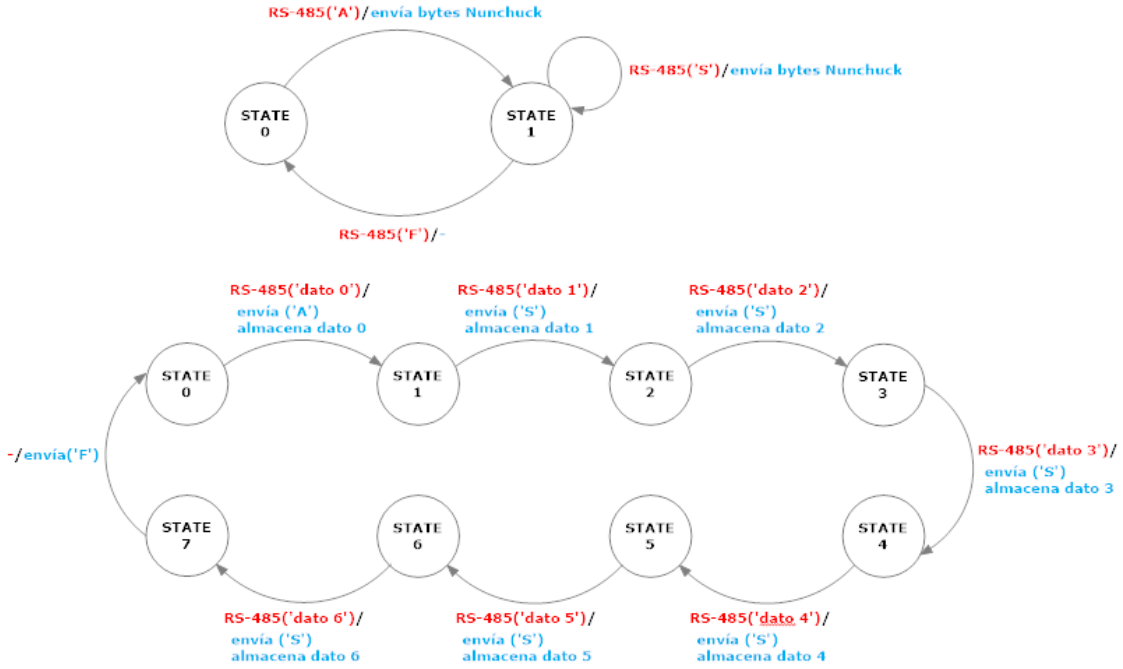


Figura 70. De arriba a abajo grafos de máquinas de estado Arduino y NXT

```
/*
  Nunchuck NXT

  Programa que lee los values enviados por el nunchuck y
  los envía por el puerto serie al NXT
*/

#include <Wire.h> // Incluye librería I2C
#define DEVICE (0x52)// Dirección Nunchuck
#define RS485CTRL 4
#define STATE_0 0
#define STATE_1 1
#define STATE_2 2
#define STATE_3 3

byte ACK;
int fsm_state = STATE_0;
byte value[2];
int nunchuck[7];
int i = 0;
int joy_x;
int joy_y;
int acel_x;
int acel_y;
int acel_z;
int but_c;
int but_z;


void setup() {
  Serial.begin(57600); // Velocidad de comunicación a 57600 baudios
  Wire.begin();
  pinMode(RS485CTRL, OUTPUT);
  digitalWrite(RS485CTRL, LOW);

  Wire.beginTransmission(0x52); // Configuración registros
```

```
  Wire.write (0xF0);
  Wire.write (0x55);
  Wire.endTransmission();
  delay(30);

  Wire.beginTransmission (0x52);
  Wire.write (0xFB);
  Wire.write (0x00);
  Wire.endTransmission();
  delay(30);

  Wire.beginTransmission(0x52);
  Wire.write (0xFA);
  Wire.endTransmission();
  delay(30);


}

void loop() {
  switch (fsm_state) // Máquina de estados
  {
    case STATE_0:

      if (Serial.available())
      {
        ACK = Serial.read();
        if (ACK == 'A')
        {
          readFrom(0x00, 6, value); // Lee datos Nunchuk
          nunchuck[0] = value[0]; // Eje x joystick
          nunchuck[1] = value[1]; // Eje y joystick
          nunchuck[2] = ((value[2] << 2) | ((value[5] & 0x0C) >> 2) &
0x03FF); // Eje x acelerómetro
          nunchuck[3] = ((value[3] << 2) | ((value[5] & 0x30) >> 4) &
0x03FF); // Eje y acelerómetro
          nunchuck[4] = ((value[4] << 2) | ((value[5] & 0xC0) >> 6) &
0x03FF); // Eje z acelerómetro
          nunchuck[5] = !((value[5] & 0x02) >> 1); // botón c
          nunchuck[6] = !(value[5] & 0x01); // botón z
          digitalWrite(RS485CTRL, HIGH);
          // Envía datos por el puerto serie
          Serial.println(nunchuck[i]);
          delay(1);
          digitalWrite(RS485CTRL, LOW);
          fsm_state = STATE_1;
        }
      }
      break;
    case STATE_1:
      if (Serial.available())
      {
        ACK = Serial.read();
        if (ACK == 'S')
        {
          i++;
          digitalWrite(RS485CTRL, HIGH);
          delay(1);
          Serial.println(nunchuck[i]);
          delay(1);
          digitalWrite(RS485CTRL, LOW);
```

```
          fsm_state = STATE_1;
        }
        else if (ACK == 'F')
        {
          fsm_state = STATE_0;
          i = 0;
        }
      }
  }
}

void writeTo(byte address, byte val)
{
  // Comienza la comunicación con el Nunchuck
  Wire.beginTransmission(DEVICE);
  Wire.write(address); // Envía dirección
  Wire.write(val); // Envía valor del registro
  Wire.endTransmission(); // Finaliza la comunicación
}

void readFrom(byte address, int num, byte value[])
{
  int i = 0;
  // Comienza la comunicación con el Nunchuck
  Wire.beginTransmission(DEVICE);
  Wire.write(address); // Envía dirección de lectura
  Wire.endTransmission(); // Finaliza la comunicación

  // Comienza la comunicación con el Nunchuck
  Wire.beginTransmission(DEVICE);
  Wire.requestFrom(DEVICE, num); // solicita bytes de datos

  while (Wire.available())
  {
    value[i] = Wire.read(); // Recibe los bytes uno por uno
    i++;
  }
  Wire.endTransmission(); // Finaliza la comunicación
}
```

```
/*
  Nunchuck NXT

  Programa que recibe los datos del Nunchuck enviados por Arduino y
  los utiliza para controlar el robot
*/

#include "NXCDefs.h"
#define HS_BAUD_9600 5    //Velocidad comunicación 9600 bits/seg
#define HS_BAUD_57600 10    //Velocidad comunicación 57600 bits/seg
#define HS_BAUD_115200 12    //Velocidad comunicación 115200 bits/seg
#define HS_MODE_8_DATA 0x00C0 //Comunicación 8 bits
#define HS_MODE_10_STOP 0x0000 //1 bit de stop
#define HS_MODE_N_PARITY 0x0800 //Sin paridad
#define HS_MODE_8N1 (HS_MODE_8_DATA|HS_MODE_N_PARITY|HS_MODE_10_STOP)
#define STATE_0 0
#define STATE_1 1
#define STATE_2 2
#define STATE_3 3
#define STATE_4 4
```

```
#define STATE_5 5
#define STATE_6 6
#define STATE_7 7

string joy_x;
string joy_y;
string acel_x;
string acel_y;
string acel_z;
string but_c;
string but_z;
int x;
int y;
int ac_x;
int ac_y;
int ac_z;
int bt_c;
int bt_z;
int state = STATE_0;

task FSM() // Máquina de estados
{
  while(true)
  {
    switch(state)
    {
      case STATE_0:
        RS485Write("A"); // Envía carácter A por puerto serie
        until(RS485DataAvailable()); // Espera que llegue un dato
        Wait(10);
        RS485Read(joy_x); // Lee dato 0
        x = StrToNum(joy_x);
        TextOut(0, LCD_LINE1,joy_x);
        state = STATE_1;
        break;
      case STATE_1:
        RS485Write("S");
        until(RS485DataAvailable());
        Wait(10);
        RS485Read(joy_y); // Lee dato 1
        y = StrToNum(joy_y);
        TextOut(0, LCD_LINE2,joy_y);
        state = STATE_2;
        break;
      case STATE_2:
        RS485Write("S");
        until(RS485DataAvailable());
        Wait(10);
        RS485Read(acel_x); // Lee dato 2
        ac_x = StrToNum(acel_x);
        TextOut(0, LCD_LINE3,acel_x);
        state = STATE_3;
        break;
      case STATE_3:
        RS485Write("S");
        until(RS485DataAvailable());
        Wait(10);
        RS485Read(acel_y); // Lee dato 3
        ac_y = StrToNum(acel_y);
        TextOut(0, LCD_LINE4,acel_y);
        state = STATE_4;
```

```
        break;
      case STATE_4:
        RS485Write("S");
        until(RS485DataAvailable());
        Wait(10);
        RS485Read(acel_z); // Lee dato 4
        ac_z = StrToNum(acel_z);
        TextOut(0, LCD_LINE5,acel_z);
        state = STATE_5;
        break;
      case STATE_5:
        RS485Write("S");
        until(RS485DataAvailable());
        Wait(10);
        RS485Read(but_c); // Lee dato 5
        bt_c = StrToNum(but_c);
        TextOut(0, LCD_LINE6,but_c);
        state = STATE_6;
        break;
      case STATE_6:
        RS485Write("S");
        until(RS485DataAvailable());
        Wait(10);
        RS485Read(but_z); // Lee dato 6
        bt_z = StrToNum(but_z);
        TextOut(0, LCD_LINE7,but_z);
        state = STATE_7;
        break;
     case STATE_7:
       RS485Write("F");
       Wait(1);
       state = STATE_0;
       break;
    }
  }
}

task main()
{
  UseRS485(); // Configura comunicación RS-485
  RS485Enable();
  RS485Uart(HS_BAUD_57600,HS_MODE_8N1);
  Wait(100);

  while(true)
  {
    StartTask(FSM);
    if (bt_c == 1) // Evalua valor botón c
    {
      if(x == 127) // Evalua eje x joystick
      {
        if(y == 128) // Evalua eje y joystick
        {
          OnFwd(OUT_B,0);
          OnFwd(OUT_C,0);
        }
        else if (y > 128)
        {
          OnFwd(OUT_B,50);
          OnFwd(OUT_C,50);
        }
```

```
        else
        {
          OnFwd(OUT_B,-50);
          OnFwd(OUT_C,-50);
        }
      }
      else if(x > 127)
      {
        OnFwd(OUT_B,0);
        OnFwd(OUT_C,50);
      }
      else
      {
        OnFwd(OUT_B,50);
        OnFwd(OUT_C,0);
      }
    }
    else
    {
      if(bt_z == 0) // Evalúa botón z
      {
        if((ac_y > 375) && (ac_y < 512)) // Evalúa eje y acelerómetro
          {
            OnFwd(OUT_B,-50);
            OnFwd(OUT_C,-50);
          }
        else if((ac_y > 512) && (ac_y < 650))
          {
            OnFwd(OUT_B,50);
            OnFwd(OUT_C,50);
          }
      }
      else
      {
        OnFwd(OUT_B,0);
        OnFwd(OUT_C,0);
      }
    }

  }
}
```

7. Robot B

Hasta ahora el único elemento del kit LEGO Mindstorms NXT que se ha utilizado ha sido el *brick* inteligente. En este nuevo robot se va a realizar el proceso inverso que con el robot anterior, es decir, con la placa Arduino Nano se van a controlar los periféricos del kit. En la figura 71 se puede observar el esquema de control del robot.

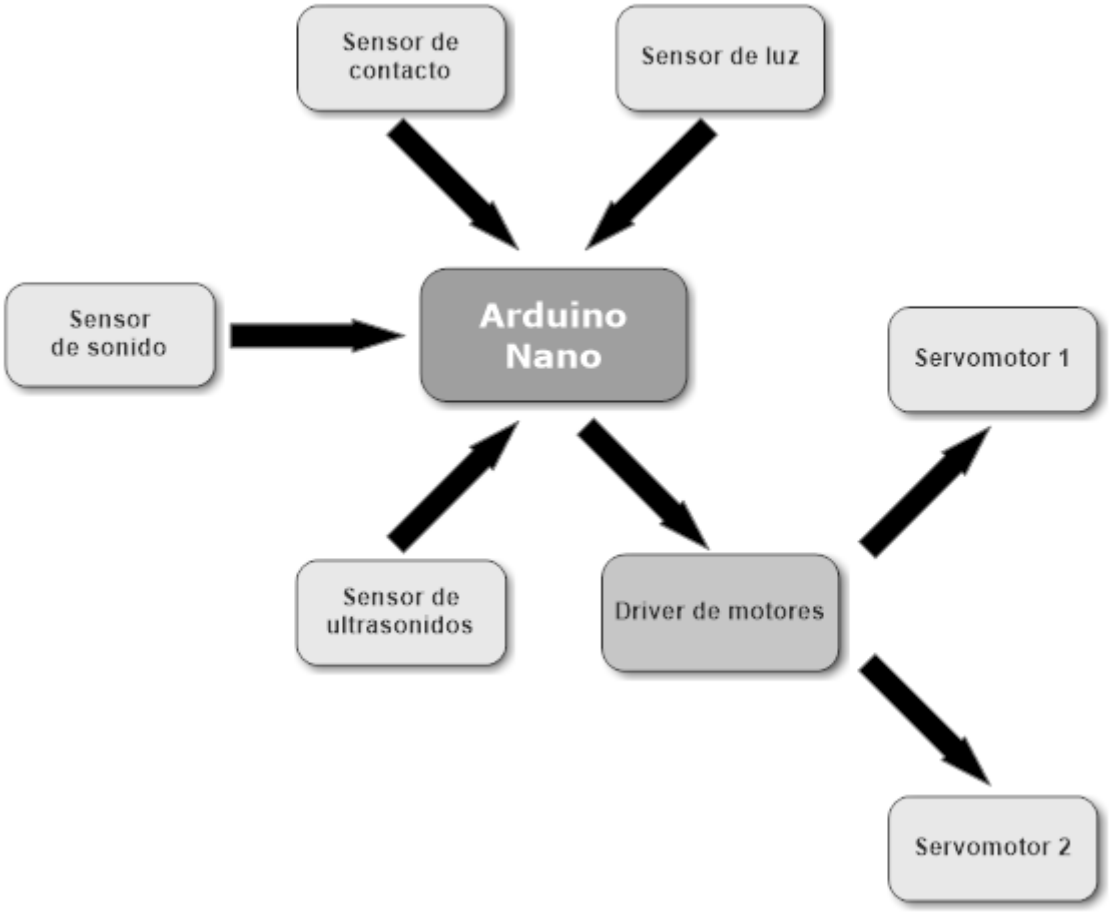


Figura 71. Esquema robot B

7.1. Chasis

Se trata de la estructura en la que va a ir integrado el sistema embebido que se encarga del control de los periféricos del kit. Está compuesto por piezas de LEGO Technic junto con una pieza realizada mediante impresión 3D.

El chasis desarrollado para este robot está basado en la construcción “Five Minute Bot” de Dave Parker, que se puede visualizar en la siguiente figura [54].

Figura 72. Five Minute Bot

Para completar el chasis, dado que el *brick* NXT no forma parte del robot, se ha creado una pieza en 3D que se encargará de sustituir al *brick* y mantener la estructura de la figura anterior. La pieza se ha realizado mediante impresión 3D, que consiste en el proceso de materialización de un objeto a través de la información digital contenida en un archivo tridimensional.

Existiendo varias técnicas de impresión 3D, para este robot se ha utilizado la técnica FDM (modelado por deposición fundida). Esta técnica consiste en la deposición de un termo-plástico fundido sobre una superficie a menor temperatura para favorecer la rápida solidificación del plástico [55]. Este proceso se realiza capa por capa.

Para poder obtener una pieza es necesario crear un archivo tridimensional que contenga la información de cómo tiene, la impresora, que construirla.

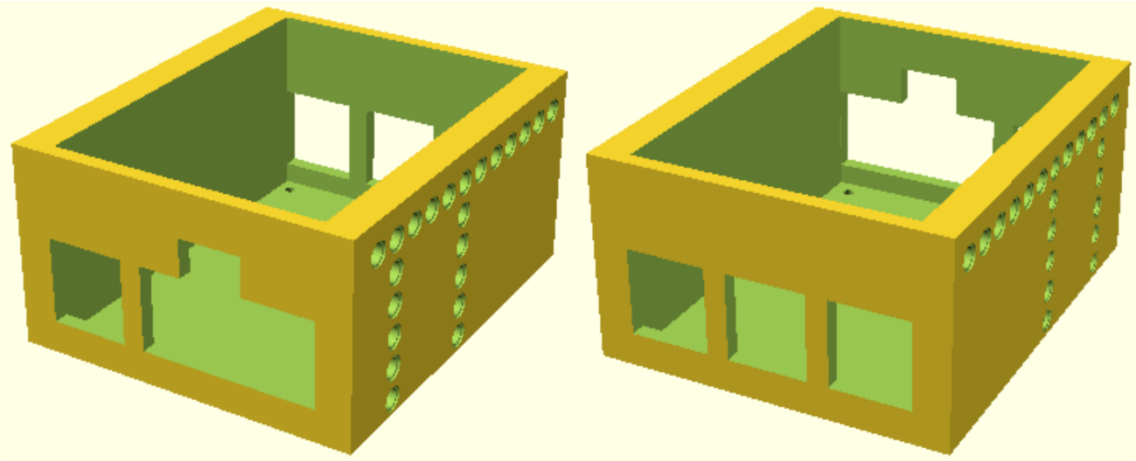


Figura 73. Diseño 3D chasis Robot B

7.2. Robot

Basado en el sistema embebido explicado anteriormente, cuyo núcleo es la placa Arduino Nano, el robot está diseñado para poder conectar todos los sensores del kit LEGO Mindstorms NXT y dos de sus actuadores.

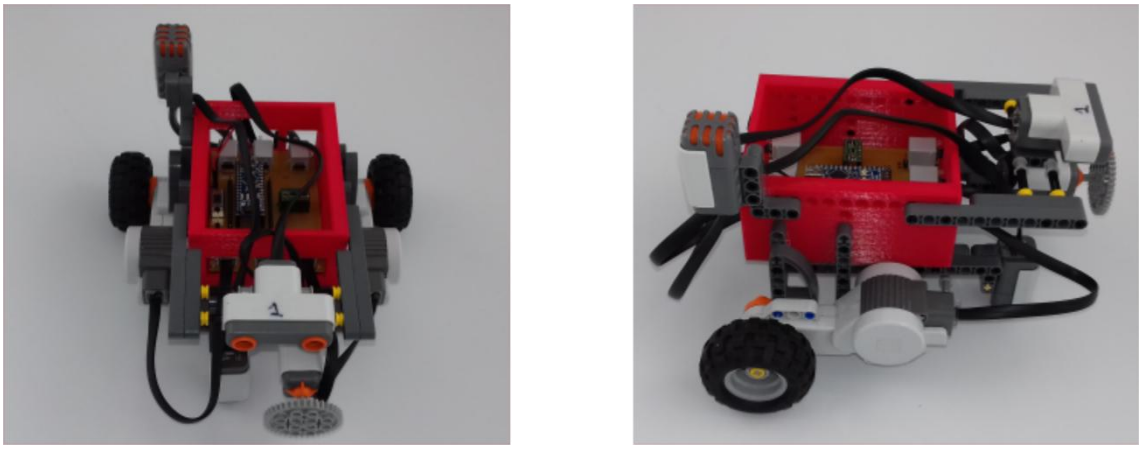


Figura 74. Robot B

Este robot cuenta, por tanto, con los sensores de tacto, sonido, luz y ultrasonidos del kit, que están adaptados para poder trabajar con el software y el hardware de Arduino. También tiene dos servomotores conectados al driver de motores DRV8835 para poder controlar su movimiento, y la posibilidad de monitorizar los encoder ópticos, que incorpora cada servomotor, a través de las interrupciones externas de la placa.

En el apartado de alimentación, el sistema está alimentado por dos baterías ion-litio que suman un total de 7.4 V, que por seguridad, están conectadas a un diodo para evitar daños en el sistema en caso de inversión de polaridad.

En la siguiente tabla se detallan las conexiones internas del sistema embebido.

Dispositivo	Pin	Arduino Nano (pin)
Pololu DRV8835	PHASE A	D8
	ENABLE A	D9
	PHASE B	D12
	ENABLE B	D10
Sensor de contacto (NXT)	ANA	D4
	GND	GND
	GND	GND
Sensor de luz (NXT)	ANA	A0
	GND	GND
	GND	GND
	VCC	VCC
	DIGIxI0	D13
Sensor de sonido (NXT)	ANA	A1
	GND	GND
	GND	GND
	VCC	VCC
Sensor de US (NXT)	ANA	7.4 V
	GND	GND
	GND	GND
	VCC	VCC
	DIGIxI0	D7
		A5 (SCL)
	DIGIxI1	A4 (SDA)

Tabla 9. Conexionado periféricos NXT con Arduino

7.3. Aplicaciones

Este robot está diseñado para poder tener unas prestaciones similares a las que se tendría con un robot montado con el kit LEGO Mindstorms. La programación se realiza mediante código C utilizando como herramienta el IDE de Arduino.

A continuación se detallan diferentes aplicaciones que se pueden llegar a realizar con este robot.

7.3.1. *Robot detector de obstáculos con sensor de contacto*

```
/*
  Robot detector de obstáculos con sensor de contacto

  El robot avanza hasta detectar un obstáculo con el sensor de
contacto,en ese instante retrocede durante 2 segundos y hace un giro
durante 500 milisegundos para comenzar otra trayectoria

*/

#define TOUCH_SENSOR 4 // Pin asociado al sensor de contacto
```

```
#define PHASE_A 8 // Pin de control de sentido del motor derecho (HIGH
hacia adelante y LOW hacia atrás)
#define ENABLE_A 9 // Pin de control de velocidad del motor derecho
#define PHASE_B 12 //Pin de control de sentido del motor izquierdo
(HIGH hacia adelante y LOW hacia atrás)
#define ENABLE_B 10 //Pin de control de velocidad del motor izquierdo

int touch;

void setup()
{
  // Configura el sensor de contacto como entrada digital
  pinMode(TOUCH_SENSOR, INPUT);
  pinMode(PHASE_A, OUTPUT);
  pinMode(ENABLE_A, OUTPUT);
  pinMode(PHASE_B, OUTPUT);
  // Configuración de los cuatro pines del control de motores como
salidas digitales
  pinMode(ENABLE_B, OUTPUT);
  // Configura la frecuencia de PWM a 3,9 kHz
  TCCR1B = TCCR1B & 0b11111000 | 0x02;
}

void loop()
{
  // Almacena el valor del sensor de contacto en la variable touch
  touch = digitalRead(TOUCH_SENSOR);
  if (touch == 1) // Evalúa si el sensor está sin presionar
  {
    digitalWrite(PHASE_A, HIGH);
    analogWrite(ENABLE_A, 127);
    digitalWrite(PHASE_B, HIGH);
    analogWrite(ENABLE_B, 127); // Avanza
  }
  else // Evalúa si el sensor está presionado
  {
    digitalWrite(PHASE_A, LOW);
    digitalWrite(PHASE_B, LOW); // Retrocede
    delay(2000);
    digitalWrite(PHASE_B, HIGH); //Gira
    delay(500);
  }
}
```

7.3.2. *Robot detector de sonido*

```
/*
  Robot detector de sonido

  El robot permanece girando hasta que detecta cierto nivel de
  ruido. En ese momento comienza a avanzar.

*/

#define SOUND_SENSOR A1 // Pin asociado al sensor de sonido
#define PHASE_A 8 // Pin de control de sentido del motor derecho (HIGH
hacia adelante y LOW hacia atrás)
#define ENABLE_A 9 // Pin de control de velocidad del motor derecho
#define PHASE_B 12 //Pin de control de sentido del motor izquierdo
(HIGH hacia adelante y LOW hacia atrás)
```

```
#define ENABLE_B 10 //Pin de control de velocidad del motor izquierdo

int sound;

void setup()
{
  pinMode(PHASE_A, OUTPUT);
  pinMode(ENABLE_A, OUTPUT);
  pinMode(PHASE_B, OUTPUT);
  // Configuración de los cuatro pines del control de motores como
salidas digitales
  pinMode(ENABLE_B, OUTPUT);
  // Configura la frecuencia de PWM a 3,9 kHz
  TCCR1B = TCCR1B & 0b11111000 | 0x02;
  analogWrite(ENABLE_A, 127);
  analogWrite(ENABLE_B, 127);
  digitalWrite(PHASE_A, HIGH); //Motor derecho avanza
}

void loop()
{
  // Almacena el valor del sensor de sonido en la variable sound
  sound = analogRead(SOUND_SENSOR);
  if (sound < 500) // Evalúa si hay mucho ruido
  {
    digitalWrite(PHASE_B, HIGH); //Avanza
  }
  else // Evalúa si hay poco ruido
  {
    digitalWrite(PHASE_B, LOW); //Gira
  }
}
```

7.3.3. *Robot siguelíneas*

```
/*
  Robot siguelíneas

  El robot sigue una línea negra sobre un fondo blanco con
  un sensor reflexivo.

*/

#define LIGHT_SENSOR A0 // Pin asociado al sensor de luz
#define LED 13 // Pin de control del LED del sensor
#define PHASE_A 8 // Pin de control de sentido del motor derecho (HIGH
hacia adelante y LOW hacia atrás)
#define ENABLE_A 9 // Pin de control de velocidad del motor derecho
#define PHASE_B 12 //Pin de control de sentido del motor izquierdo
(HIGH hacia adelante y LOW hacia atrás)
#define ENABLE_B 10 //Pin de control de velocidad del motor izquierdo

int light;
int Kp = 10; //Constante proporcional
int vel = 127; //velocidad de los motores
int offset = 54; //valor medio de medida del sensor
int error;
int turn;
int m_a,m_b;
```

```
void setup()
{
  pinMode(LED,OUTPUT); // Configuración del LED como salida digital
  pinMode(PHASE_A,OUTPUT);
  pinMode(ENABLE_A,OUTPUT);
  pinMode(PHASE_B,OUTPUT);
  // Configuración de los cuatro pines del control de motores como
salidas digitales
  pinMode(ENABLE_B,OUTPUT);
  digitalWrite(LED,HIGH); // Enciende el LED del sensor
  digitalWrite(PHASE_A,HIGH);
  digitalWrite(PHASE_B,HIGH);
}

void loop()
{
 // Almacena el valor del sensor de luz en la variable light
 light = analogRead(LIGHT_SENSOR);
 // Normaliza la medida a porcentaje (0 - 100)
 light = map(light,0,1023,0,100);
 error = light - offset; //error del sistema
 turn = Kp*error;
 m_a = vel - turn;
 m_b = vel + turn;
 analogWrite(ENABLE_A,m_a);
 analogWrite(ENABLE_B,m_b);

 }
```

7.3.4. *Robot sumo*

```
/*
  Robot sumo

  El robot permanece dentro un recinto delimitado por una línea negra
y
  cuando detecta que el sensor de contacto está presionado (ha
detectado
  el rival), aumenta la velocidad de los motores para sacarlo del
recinto

*/

#define LIGHT_SENSOR A0 // Pin asociado al sensor de luz
#define TOUCH_SENSOR 4 // Pin asociado al sensor de contacto
#define LED 13 // Pin de control del LED del sensor
#define PHASE_A 8 // Pin de control de sentido del motor derecho (HIGH
hacia adelante y LOW hacia atrás)
#define ENABLE_A 9 // Pin de control de velocidad del motor derecho
#define PHASE_B 12 //Pin de control de sentido del motor izquierdo
(HIGH hacia adelante y LOW hacia atrás)
#define ENABLE_B 10 //Pin de control de velocidad del motor izquierdo

int light;
int touch;

void setup()
{
  pinMode(LED, OUTPUT); // Configuración del LED como salida digital
  pinMode(PHASE_A, OUTPUT);
```

```
  pinMode(ENABLE_A, OUTPUT);
  pinMode(PHASE_B, OUTPUT);
  // Configuración de los cuatro pines del control de motores como
salidas digitales
  pinMode(ENABLE_B, OUTPUT);
  // Configura la frecuencia de PWM a 3,9 kHz
  TCCR1B = TCCR1B & 0b11111000 | 0x02;
  digitalWrite(LED, HIGH); // Enciende el LED del sensor
  digitalWrite(PHASE_A, HIGH);
  digitalWrite(PHASE_B, HIGH);
}

void loop()
{
  // Almacena el valor del sensor de luz en la variable light
  light = analogRead(LIGHT_SENSOR);
  // Almacena el valor del sensor de contacto en la variable touch
  touch = digitalRead(TOUCH_SENSOR);
  // Normaliza la medida a porcentaje (0 - 100)
  light = map(light, 0, 1023, 0, 100);
  if (light < 54) // Evalúa si ve blanco
  {
    digitalWrite(PHASE_A, HIGH);
    digitalWrite(PHASE_B, HIGH);
    if (touch == 1)
    {
      analogWrite(ENABLE_A, 127);
      // Avanza al 50% de su velocidad máxima
      analogWrite(ENABLE_B, 127);
    }
    else
    {
      analogWrite(ENABLE_A, 255);
      // Avanza al 100% de su velocidad máxima
      analogWrite(ENABLE_B, 255);
    }
  }
  else // Evalúa si ve negro
  {
    digitalWrite(PHASE_A, LOW);
    digitalWrite(PHASE_B, LOW); // Retrocede
    delay(1500);
    digitalWrite(PHASE_B, HIGH); // Gira
    delay(500);
  }
}
```

7.3.5. *Robot con avance preciso*

```
/*
  Robot con avance preciso

  El robot es capaz de avanzar o retroceder una distancia
  determinada en cm.

*/

#define PHASE_A 8 // Pin de control de sentido del motor derecho (HIGH
hacia adelante y LOW hacia atrás)
#define ENABLE_A 9 // Pin de control de velocidad del motor derecho
```

```
#define PHASE_B 12 //Pin de control de sentido del motor izquierdo
(HIGH hacia adelante y LOW hacia atrás)
#define ENABLE_B 10 //Pin de control de velocidad del motor izquierdo
#define ENCODER_B 2


int x;

void setup()
{
  pinMode(PHASE_A, OUTPUT);
  pinMode(ENABLE_A, OUTPUT);
  pinMode(PHASE_B, OUTPUT);
  // Configuración de los cuatro pines del control de motores como
salidas digitales
  pinMode(ENABLE_B, OUTPUT);
  pinMode(ENCODER_B, INPUT);
  attachInterrupt(digitalPinToInterrupt(ENCODER_B), encoder, CHANGE);
  // Configura la frecuencia de PWM a 3,9 kHz
  TCCR1B = TCCR1B & 0b11111000 | 0x02;
  digitalWrite(PHASE_A, HIGH);
  digitalWrite(PHASE_B, HIGH);
}

void loop()
{
  // Evalúa si sobrepasa la distancia seleccionada
  if (x < distance(100))
  {
    analogWrite(ENABLE_A, 127);
    analogWrite(ENABLE_B, 127); //Avanza
  }
  else
  {
    analogWrite(ENABLE_A, 0);
    analogWrite(ENABLE_B, 0);// Para
  }
}

void encoder()
{
  x++;
}

int distance(int cm)
{
  //Devuelve el número de cambios que tendrá el encoder
  return (cm * 36) / 1.76;
}
```

8. Conclusiones y trabajos futuros

8.1. Conclusiones

A lo largo de la realización del trabajo se ha llegado a las siguientes conclusiones:

- La tecnología del *brick* NXT, a pesar de ser del 2006, no está obsoleta y puede competir todavía con mucha de la que existe actualmente en el mercado.
- Arduino es una plataforma que está en constante crecimiento.
- El kit LEGO Mindstorms es compatible con otros dispositivos que pueden hacer de él una plataforma más potente.
- Arduino y LEGO Mindstorms son dos plataformas que pueden acercar a la gente a la robótica de una forma fácil y sencilla.
- La importancia que tiene en la fabricación de un robot un correcto diseño del chasis es alta.
- Las técnicas de corte por láser e impresión 3D son muy útiles en robótica.
- La aparición de plataformas *open source* han potenciado el desarrollo de la robótica entre la población.

8.2. Trabajos futuros

Partiendo del desarrollo que se ha realizado durante el trabajo y del punto en que han quedado los dos prototipos, a continuación se detallan posibles líneas de trabajo a seguir en el futuro:

- Incorporación de nuevos dispositivos al robot A, como pueden ser un sensor de temperatura, una brújula electrónica, un sensor de contacto, etc.
- Incorporación de una pantalla OLED al robot B.
- Modificación de ambos robots para que sean capaces de controlar un número mayor de actuadores.
- Estudio de la posibilidad de introducir la visión artificial en ambos robots.
- Aplicación de los robots en el aula para ampliar el espectro de enseñanza en robótica educativa que tienen las dos plataformas utilizadas.

9. Anexo

9.1. Fabricación de PCB

Partiendo del diseño realizado con la herramienta KiCad, se toman medidas del tamaño del PCB y se corta una placa fibra de vidrio recubierta de cobre con el tamaño correspondiente (Figura 75).

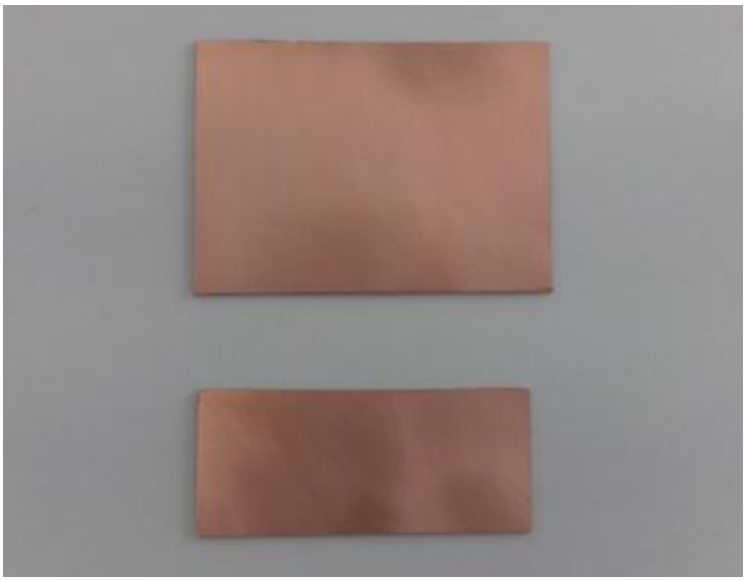


Figura 75. Placas de cobre cortadas

A continuación se imprime el diseño con una impresora láser sobre papel de transferencia térmica (Figura 76).

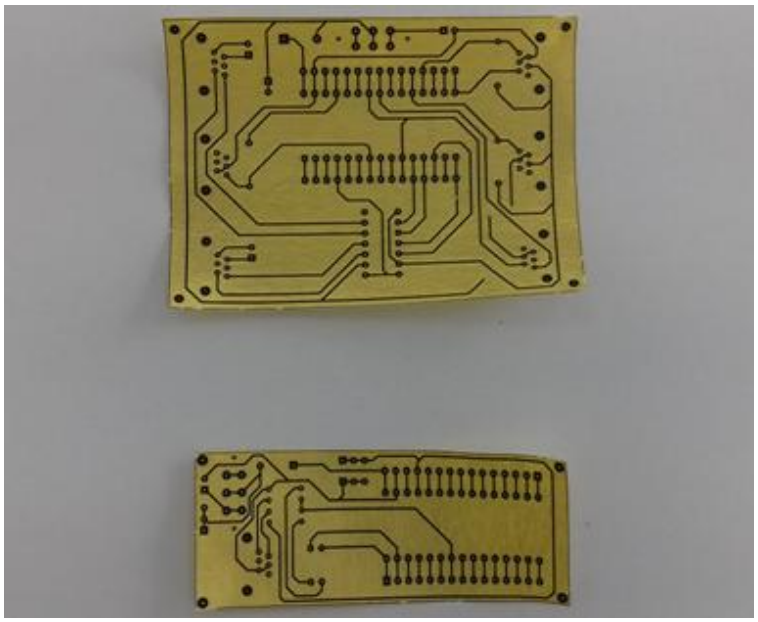


Figura 76. Diseño PCB impreso

Una vez que se tiene la placa de cobre cortada y el diseño impreso, se va a proceder a transferir el diseño a la placa de cobre mediante transferencia térmica (Figura 77). Antes de realizar este proceso es necesario limpiar bien la placa de cobre para quitar el óxido y la grasa que pueda tener. Esto se puede hacer con un estropajo y con alcohol.

Para transferir el diseño a la placa se ha utilizado una plancha, ya que la impresión está hecha con tóner y de esta forma se adherirá a la placa de cobre una vez se le aplique calor.

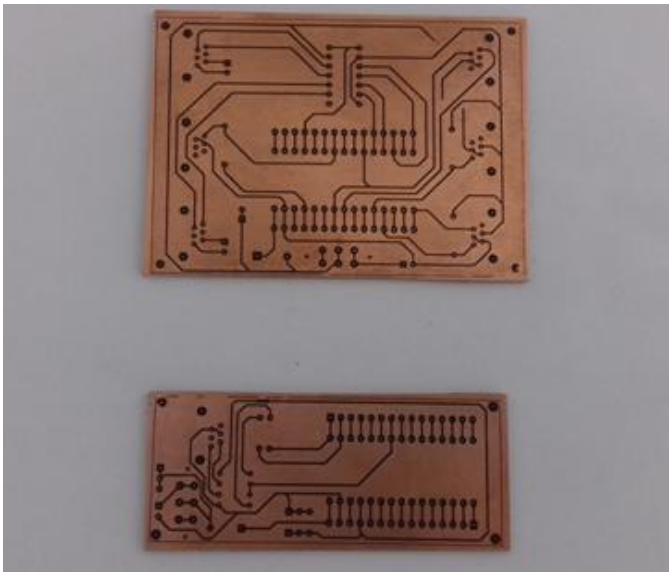


Figura 77. Diseño sobre placa de cobre

El siguiente paso consiste en eliminar todo el cobre que no está recubierto de tóner. Para ello es necesario elaborar una solución química compuesta por ácido clorhídrico (HCl) y percarbonato sódico (Na_2CO_3). Se introduce la placa en la solución y se espera a que desaparezca el cobre (Figura 78).

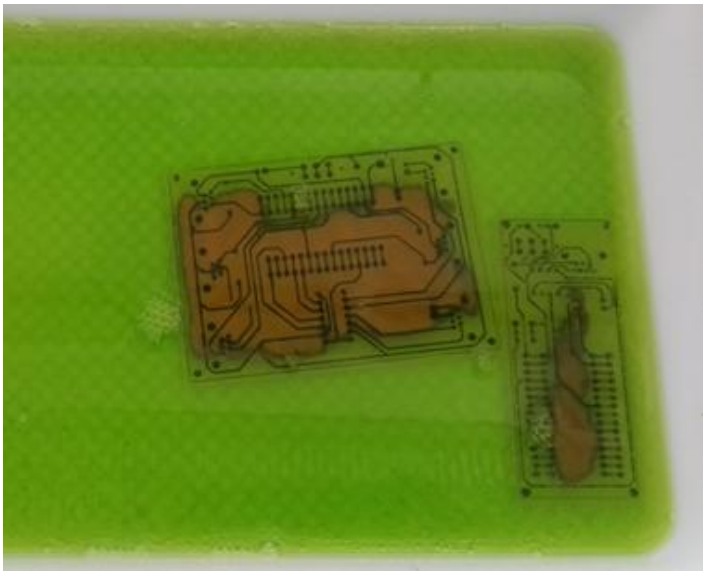


Figura 78. Proceso químico diseño PCB

Finalizado el proceso químico se procede a eliminar los restos de tóner de la placa, dejando así las pistas de circuito visibles (Figura 79).

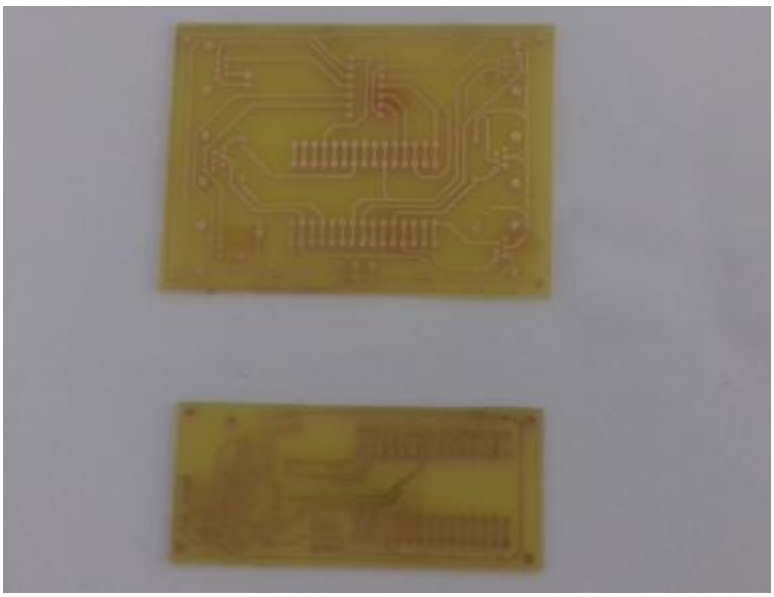


Figura 79. PCB

Para finalizar, solo falta taladrar los agujeros donde irán insertados los componentes y proceder a la soldadura de dichos componentes (Figura 80).

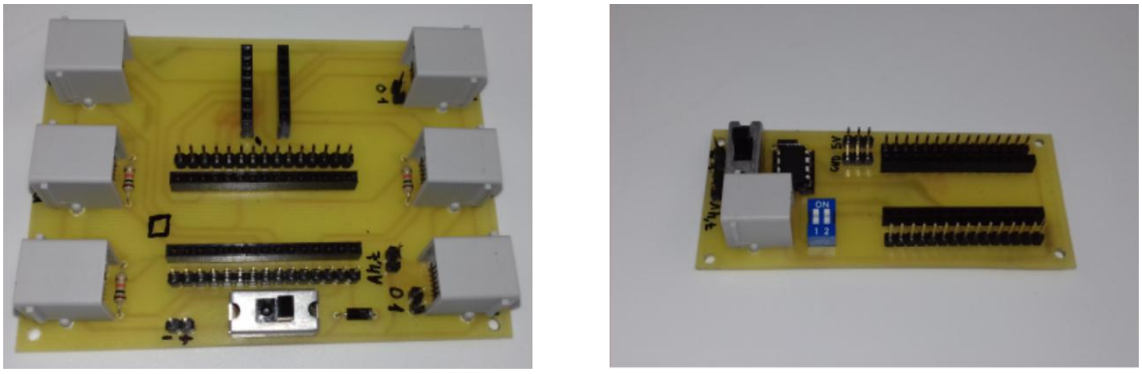
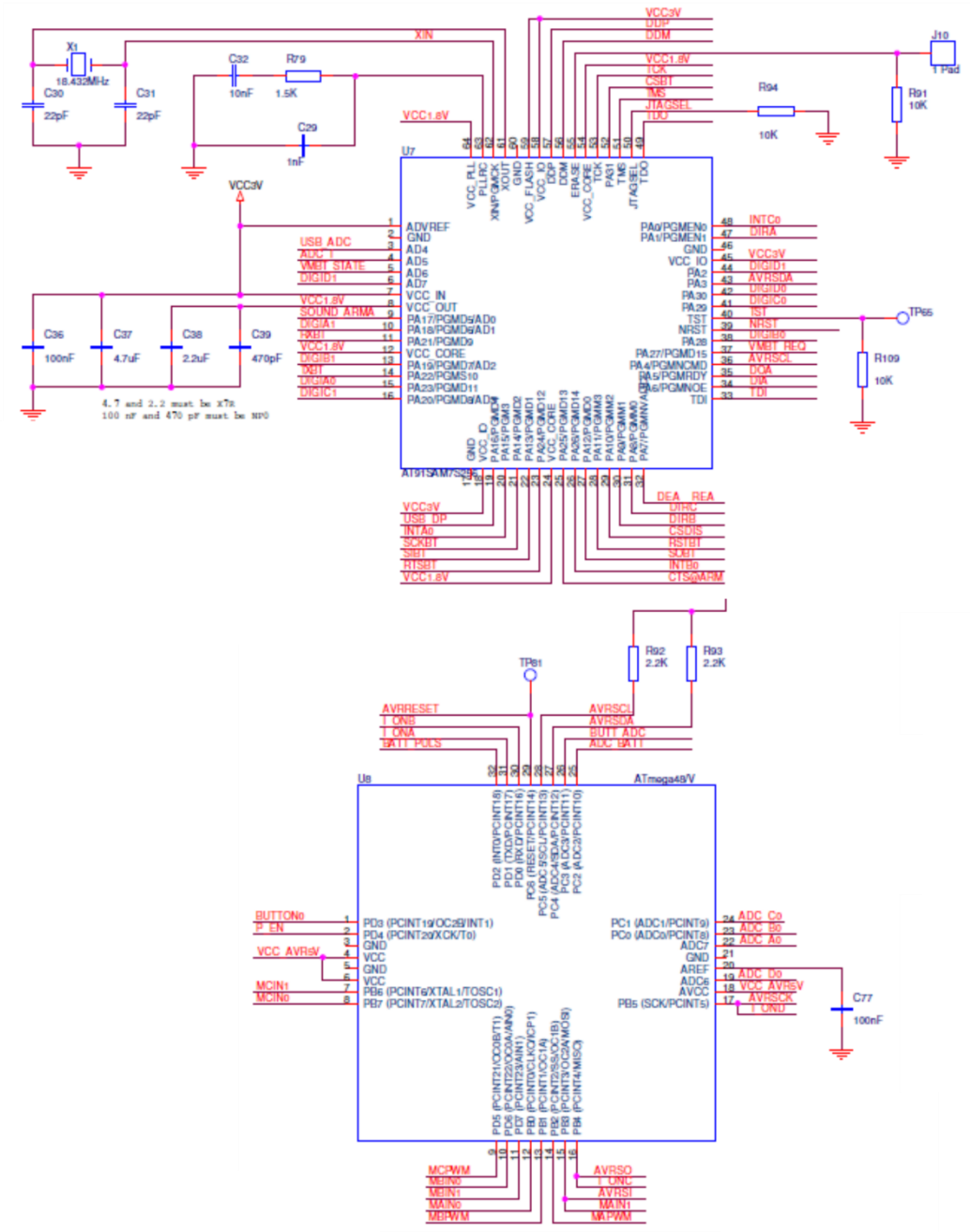


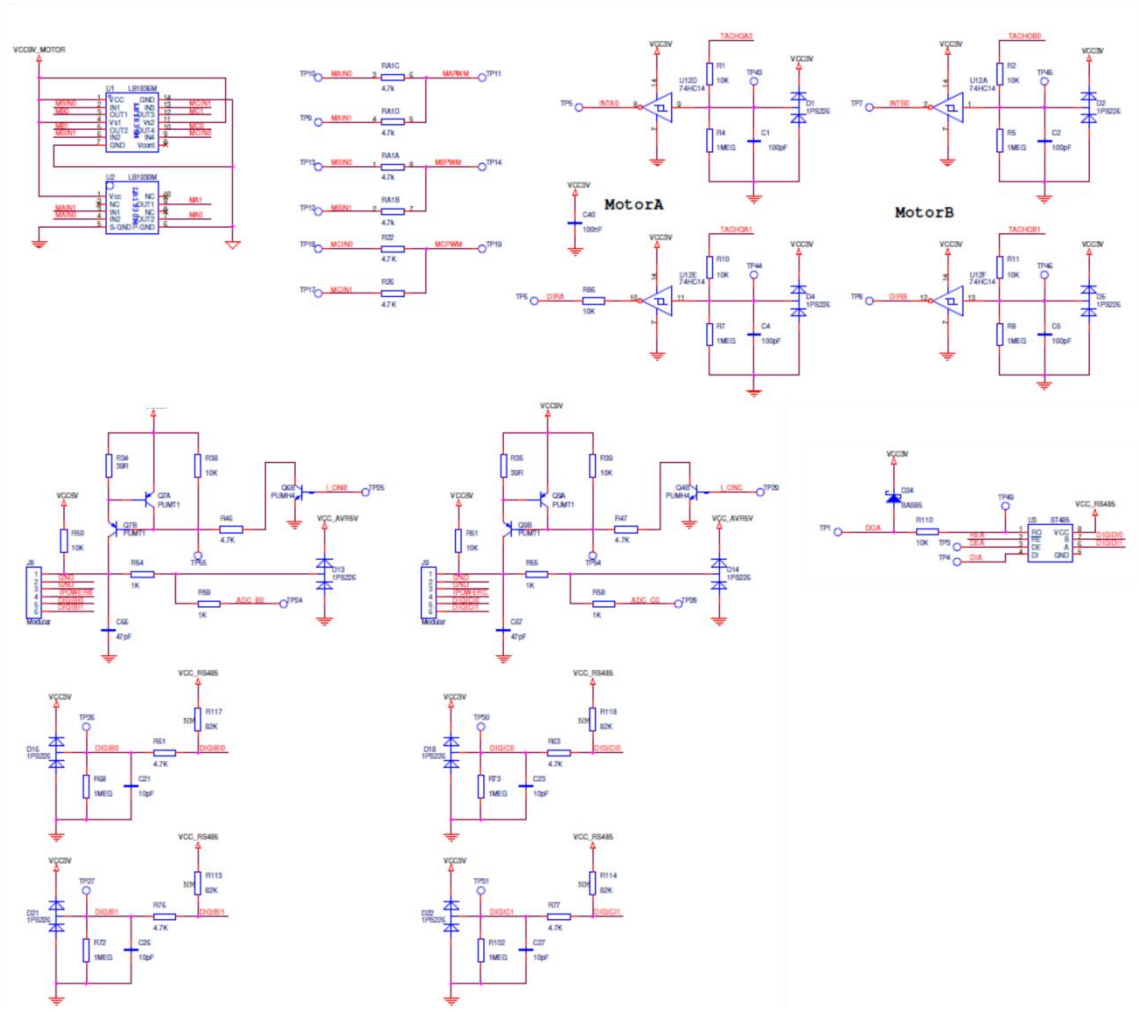
Figura 80. PCB terminada

V. Planos

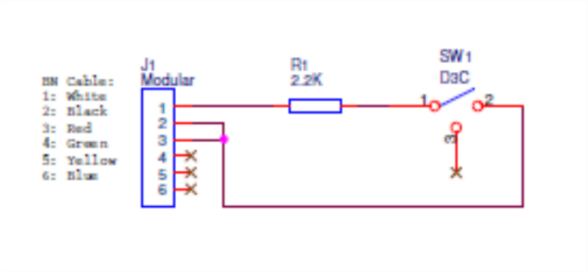
1. Documentación NXT

1.1. Brick NXT

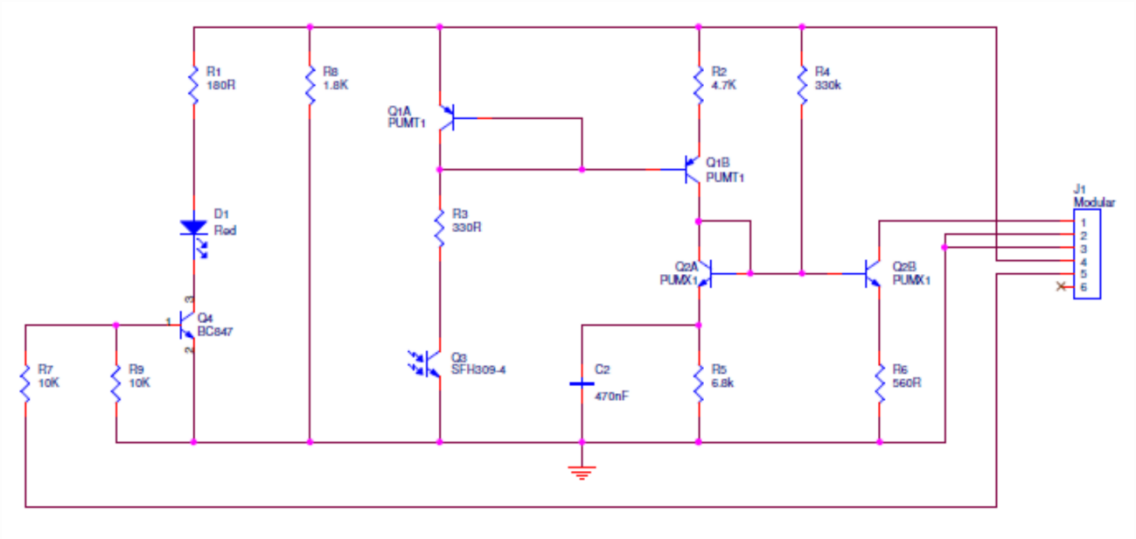




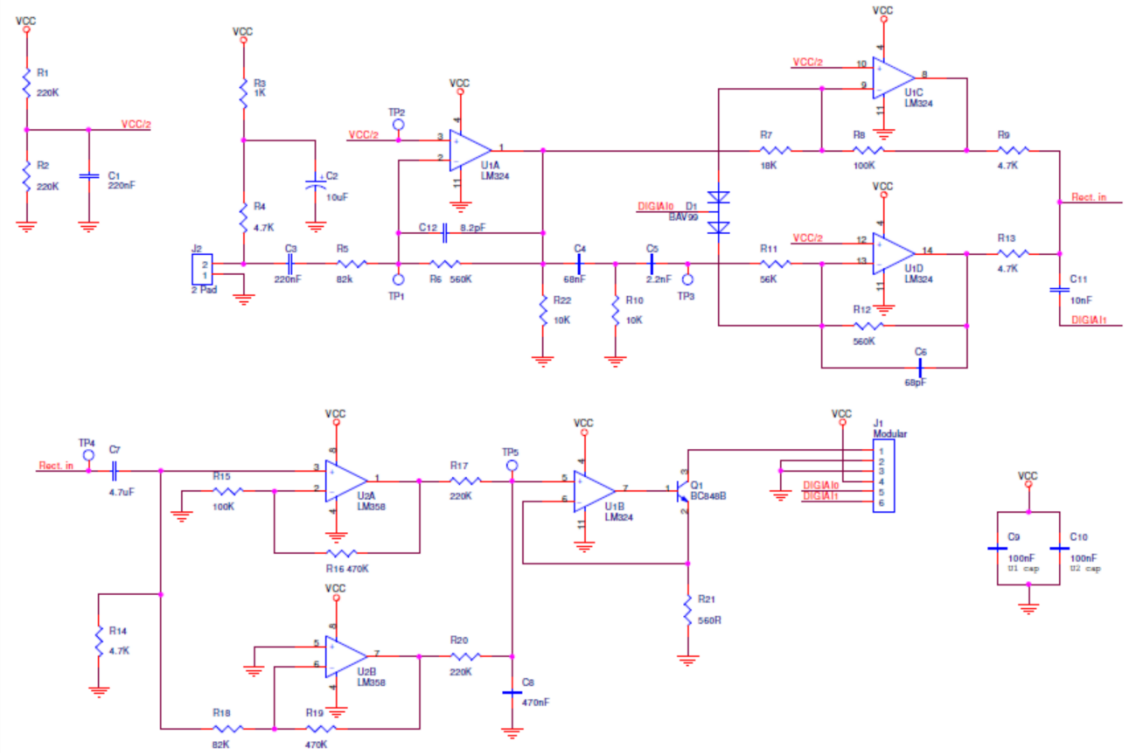
1.2. Sensor de contacto



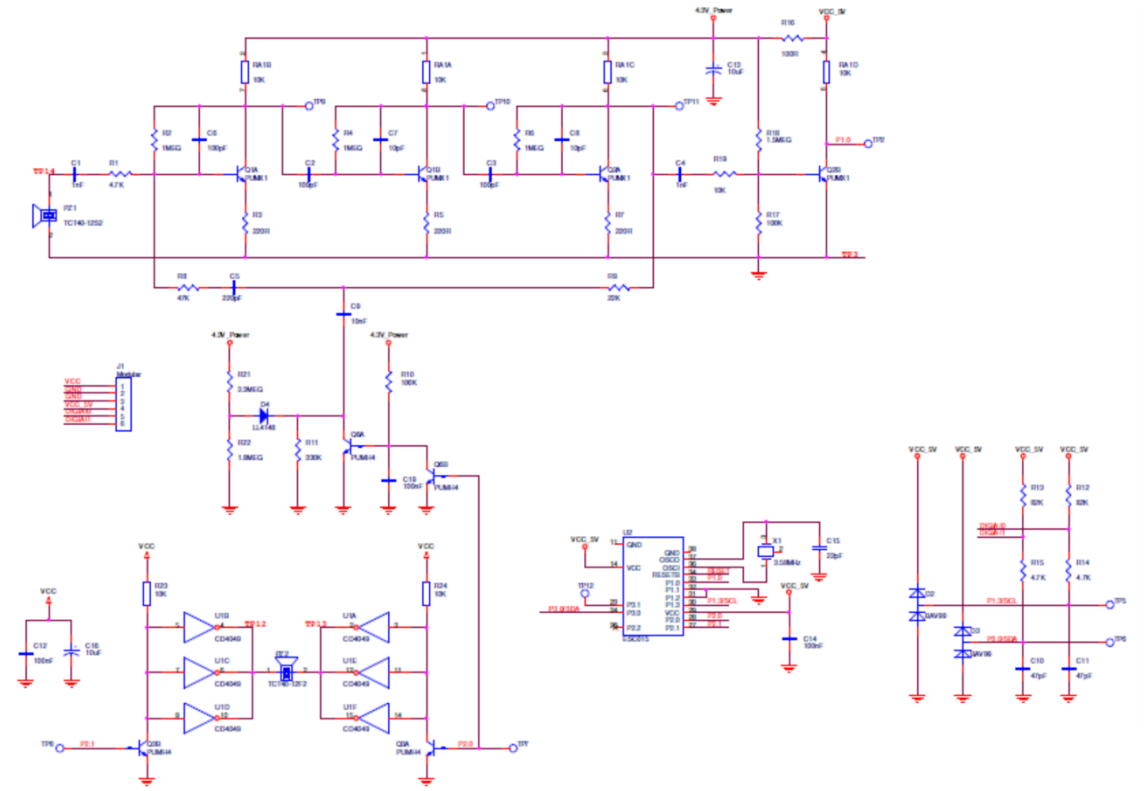
1.3. Sensor de luz



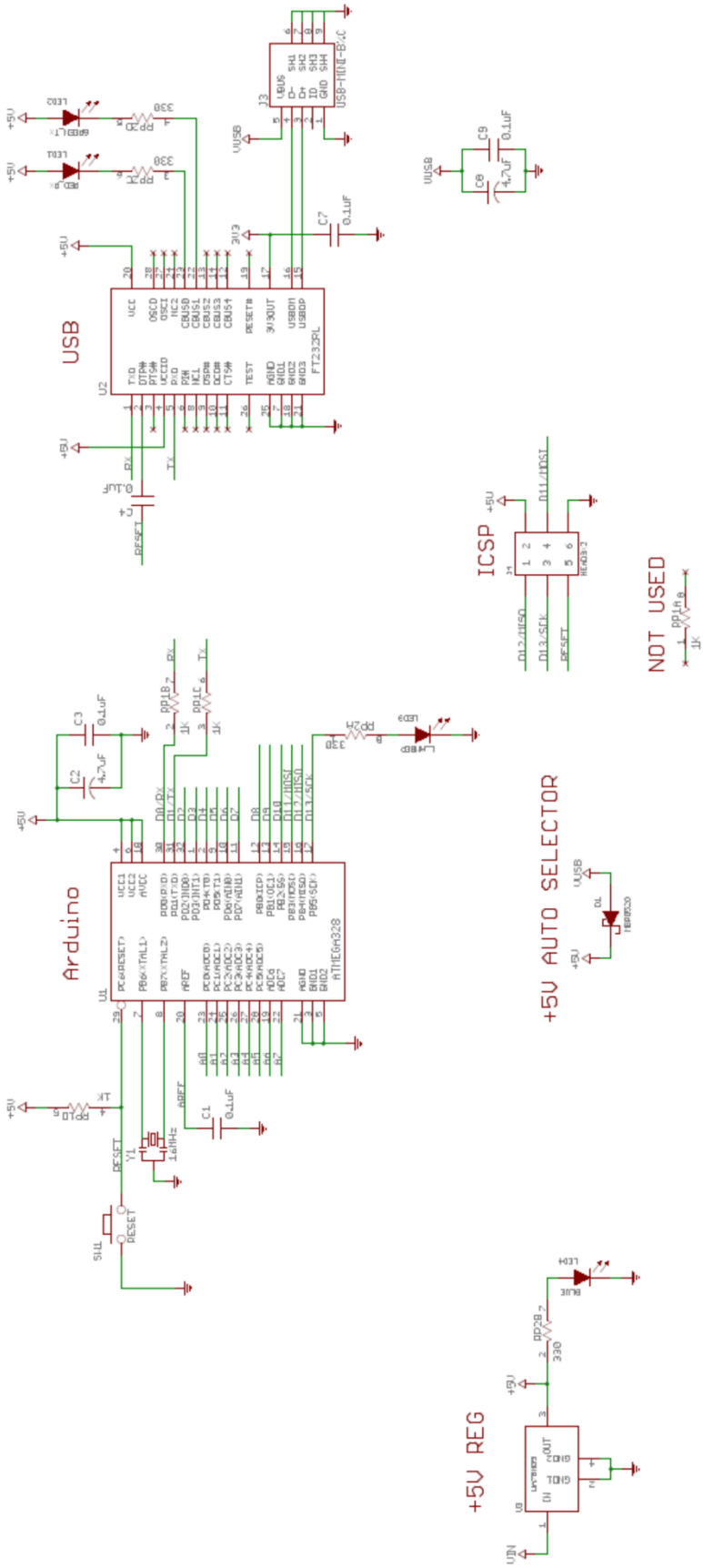
1.4. Sensor de sonido



1.5. Sensor de ultrasonidos

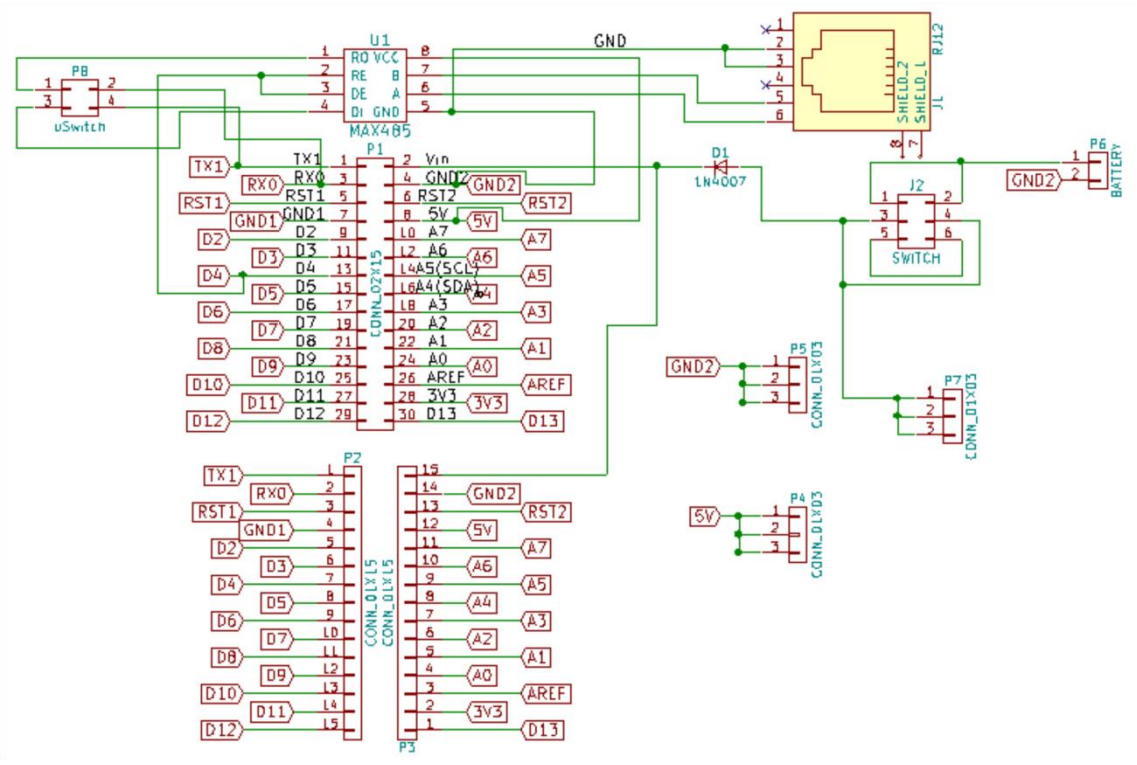


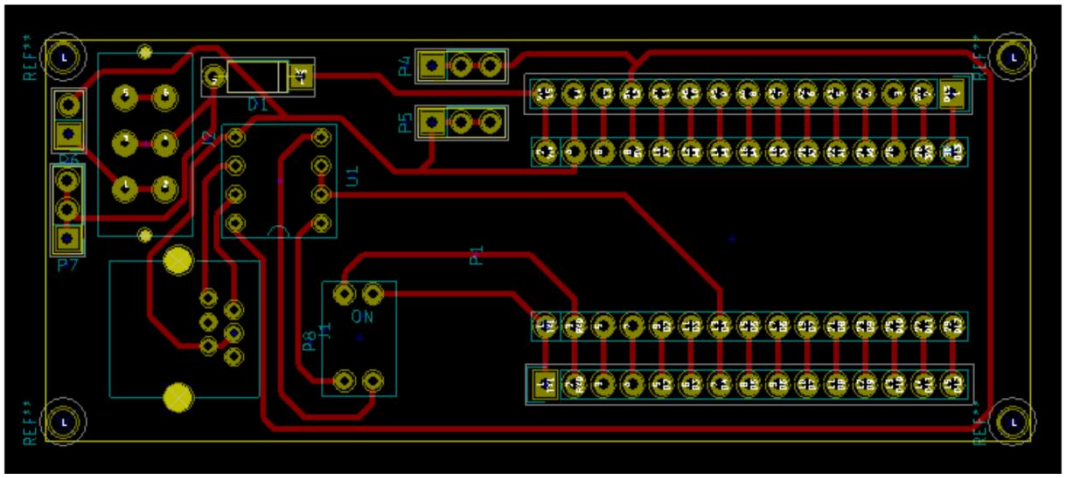
2. Documentación Arduino Nano



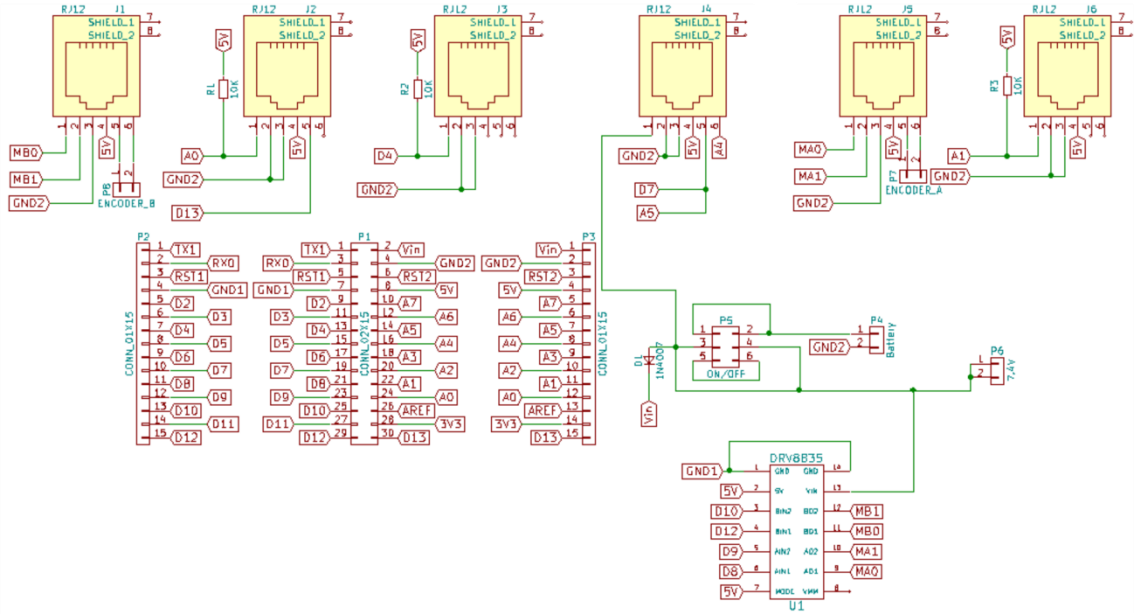
3. Placas de circuito impreso (PCB) desarrolladas

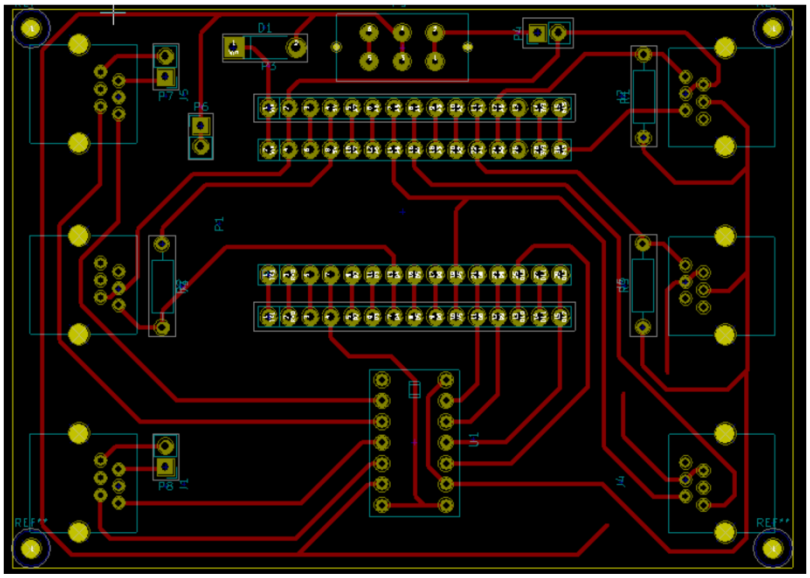
3.1. Adapatador Arduino Nano



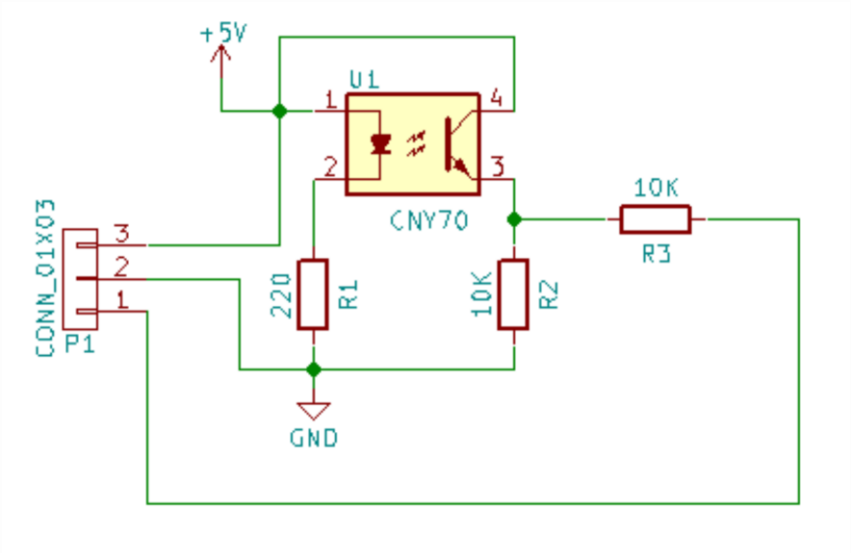


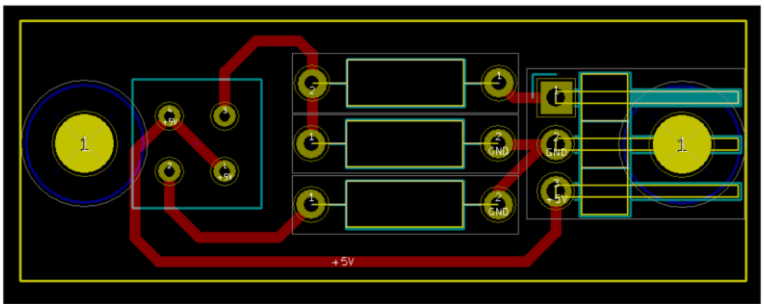
3.2. Sistema embebido



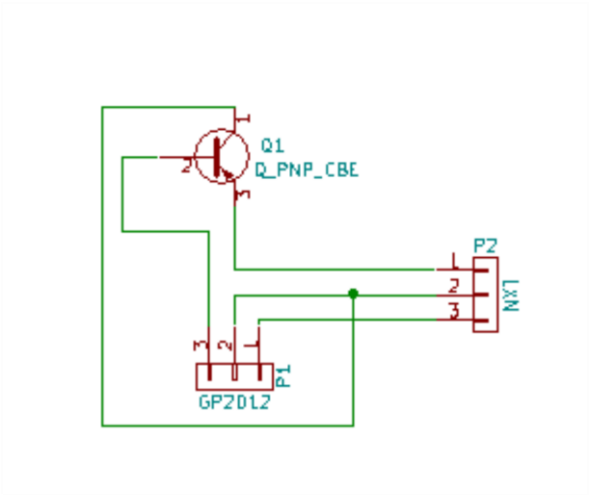


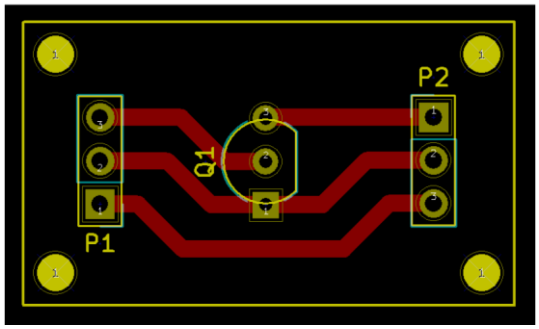
3.3. PCB adaptador CNY70



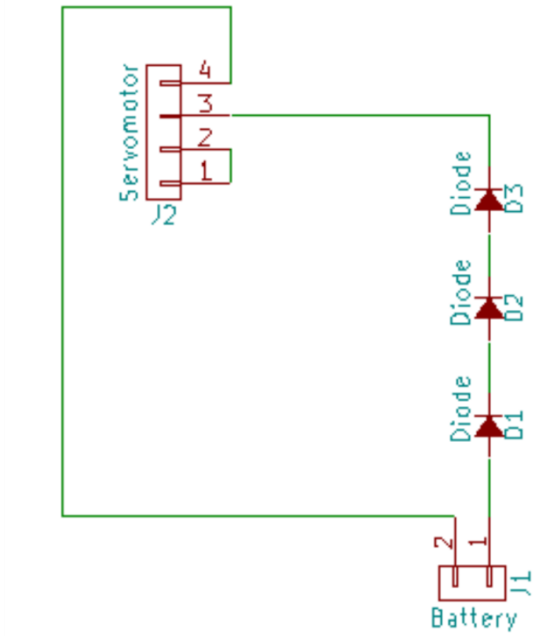


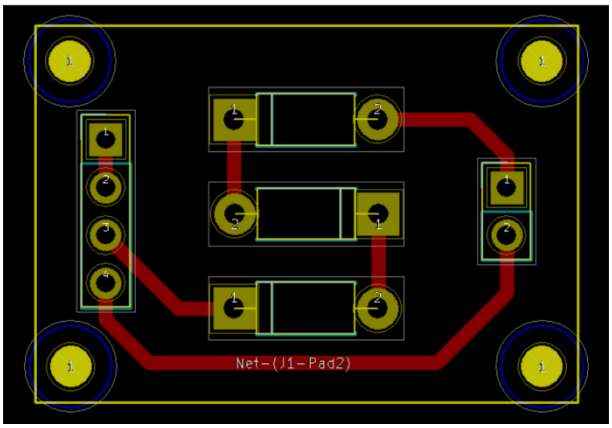
3.4. PCB adaptador GP2D12



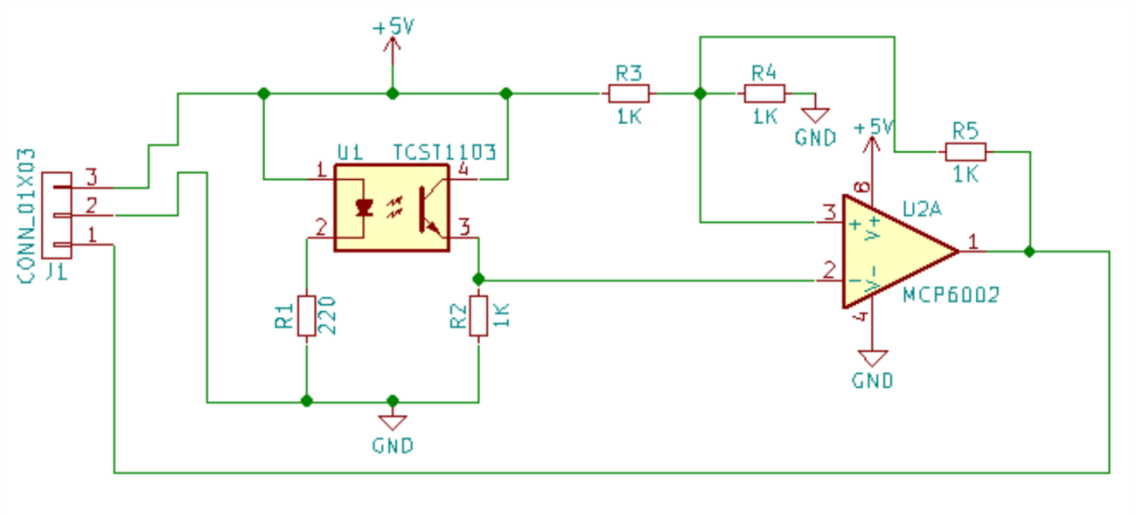


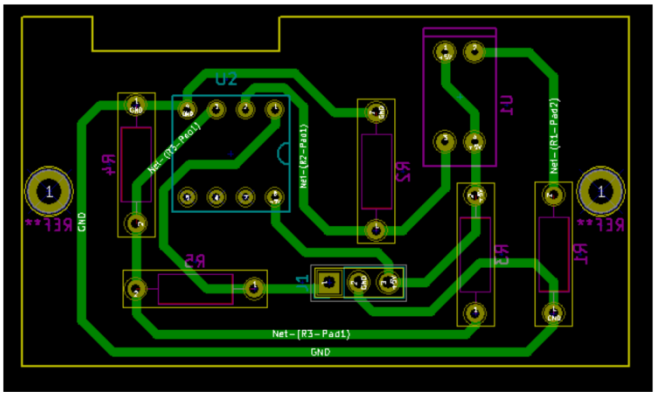
3.5. PCB adaptador servomotor



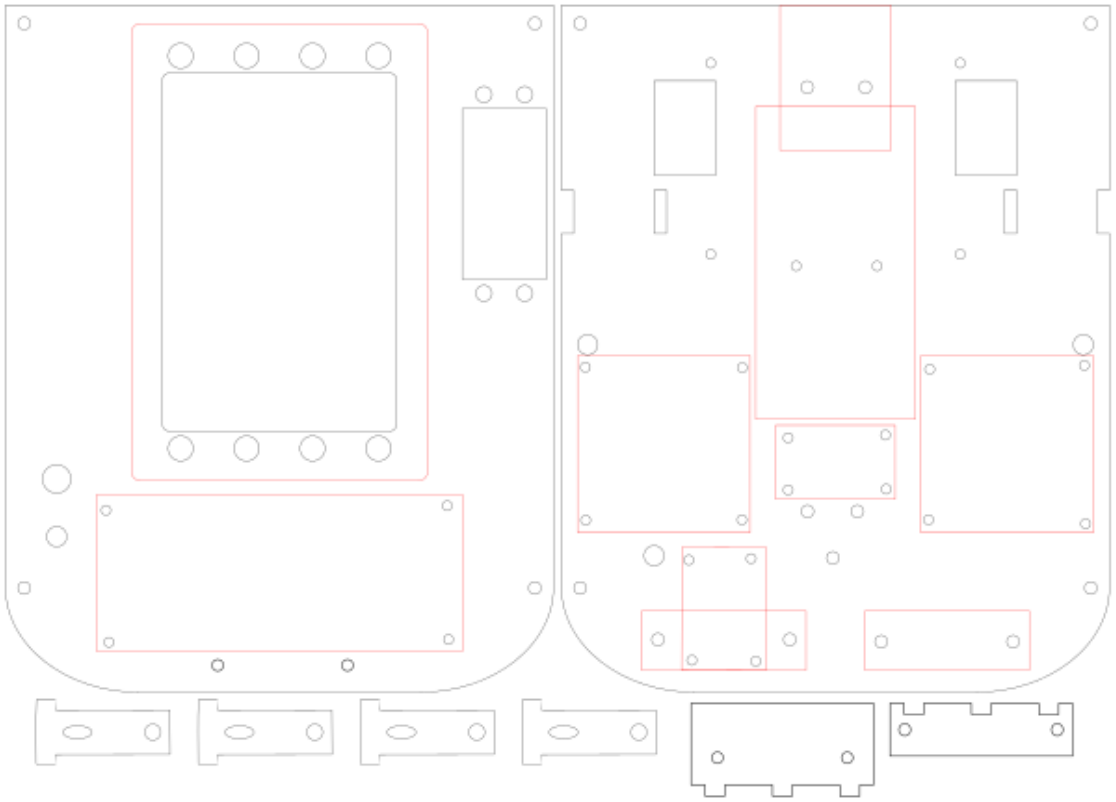


3.6. PCB adaptador encoder



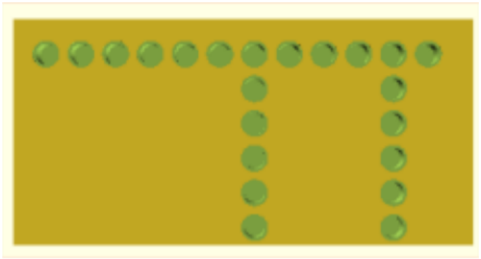


4. Diseño 2D



5. Diseño 3D

5.1. Chasis Robot B



5.2. Disco codificador

VI. Pliego de condiciones

En este apartado se van a evaluar los requisitos utilizados para la realización de este trabajo.

1. Requisitos hardware

- Kit LEGO Mindstorms NXT
- Placa Arduino Nano
- Sensor GP2D12 o similares características
- Sensores CNY70 o similares características
- Sensores TCST1103 o similares características
- Motores de corriente continua con escobillas capaces de trabajar a una tensión de 7.4 V con una corriente máxima de 1 A.
- Servomotor estándar de radiocontrol con una corriente máxima de
- Driver de motores puente en H
- Conectores RJ12 LEGO
- Impresora 3D con área de impresión de 20x15x10 o superior capaz de trabajar con PLA o ABS.
- Cortadora láser
- Baterías de 7.4 V con 2000 mAh de capacidad o superiores

2. Requisitos software

- Sistema operativo Windows 10 o similar
- IDE Arduino v1.6.0 o superior
- KiCad EDA v4.0.6 o superior
- Bricx Command Center v3.3.8.11 o superior
- Inkscape v0.91 o superior
- OpenSCAD v2015.03-2 o similar
- Notepad++ o similar

VII. Presupuesto

En este apartado se va a realizar una estimación del coste total que supone la realización de este Trabajo Fin de Grado.

El cálculo del coste total se va a desglosar en dos partes:

- Coste del material
- Coste de personal

1. Coste del material

Dentro del coste del material hay que diferenciar entre los costes del equipamiento hardware y software y los costes de los componentes y dispositivos utilizados.

Elemento	Precio (€)	Duración	Tiempo de uso	Coste (€)
Portátil HP Pavillion	699	4 años	6 meses	87,37
Impresora 3D	1600	4 años	1 meses	33,50
Impresora láser	100	4 años	6 meses	12,5
Total				133,37

Tabla 10. Costes de equipos

Elemento	Coste (€)
Licencia Microsoft Office 2016	220

Tabla 11. Costes de software

Elemento	Unidades	Precio unitario (€)	Coste (€)
Kit LEGO Mindstorms NXT	1	349,69	349,69
Placa Arduino Nano	2	20,00	40,00
GP2D12	1	16,32	16,32
CNY70	2	1,19	2,38
TCST1103	2	1,69	3,38
Motores DC	2	3,99	7,98
Driver DRV8835	1	5,43	5,43
Servomotor	1	9,99	9,99
Rueda loca	1	4,99	4,99
Ruedas	1	6,99	6,99
Módulo Keyes L9110 Fan	1	14,52	14,52
Baterías io-litio + cargador	2	7,70	15,4
Portabaterías	2	0,71	1,42
Conectores RJ12 LEGO	20	2,44	48,8
Resistencias	30	0,03	0,90
Diodos	10	0,05	0,50
Amp. Op. “rail to rail”	2	0,8	1,60

Plancha fibra de vidrio	1	8	8
Papel PCB	5	0,5	2,50
Conectores	10	0,59	5,90
Interruptores	2	2,4	4,8
Kit cables prototipado	1	5,49	5,49
DM	1	12	12
PLA	1	20	20
Transistores	1	0,14	0,14
MAX485	2	3,45	6,90
Corte por láser	60 min	1,21€/min	72,6
Material químico PCB	1	11,50	11,5
Total			680,12

Tabla 12. Costes de material

2. Coste de personal

El coste de mano de obra se ha dividido en las distintas fases de desarrollo del proyecto.

Concepto	Nº horas	Precio por hora (€)	Coste (€)
Estudio LEGO Mindstorms	30	30	900
Estudio Arduino	15	30	450
Selección de componentes	30	30	900
Diseño de PCB	40	30	1.200
Fabricación de PCB	25	30	750
Soldadura	20	30	600
Diseño 3D	10	30	300
Diseño 2D	15	30	450
Documentación	150	10	1.500
Total	335		7.050

Tabla 13. Costes de personal

3. Coste total

Concepto	Coste (€)
Costes de equipos	133,37
Costes de software	220,00
Costes de material	680,12
Costes de personal	7050,00
Total	8.083,49

Tabla 14. Coste total del trabajo

El coste total del trabajo ascendería a 8.083,49 €.

VIII. Bibliografía y referencias

[1] J. Pérez Tudela, "STEM, STEAM... ¿pero eso qué es? - ODITE: Observatorio de Innovación Tecnológica y Educativa", Odite.ciberespiral.org, 2016. [Online]. Disponible: http://odite.ciberespiral.org/comunidad/ODITE/recurso/stem-steam-pero-eso-que-es/58713dbd-414c-40eb-9643-5dee56f191d3

[2] "Crumble", Complubot.com, 2015.[Online].Disponible: http://complubot.com/inicio/proyectos/swr/crumble/

[3] Microbit.org, 2017. [Online]. Disponible: http://microbit.org/es/

[4] "Raspberry Pi 3 Model B - Raspberry Pi", Raspberry Pi, 2016. [Online]. Disponible: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[5] "Historia - Mindstorms LEGO.com", Lego.com, 2015. [Online]. Disponible: https://www.lego.com/es-es/mindstorms/history

[6] M. Pablo Turmero (2015), "Lego Mindstorms - Monografias.com", Monografias.com. [Online]. Disponible: http://www.monografias.com/trabajos105/lego-mindstorms/lego-mindstorms.shtml

[7] "Comparativa Mindstorms NXT vs. EV3-electric*Brick*s", Blog.electric*brick*s.com, 2014. [Online]. Disponible: http://blog.electric*brick*s.com/2014/07/comparativa-mindstorms-nxt-ev3/

[8] "CSC 126 Lab #13", Faculty.berea.edu. [Online]. Disponible: https://faculty.berea.edu/faculty/pearcej/CSC126/bottasks/L13-NXT-prog.html

[9] L. 9841, "LEGO NXT Intelligent *Brick* Set 9841 | *Brick* Owl - LEGO Marketplace", *Brick*owl.com, 2015. [Online]. Disponible: http://www.*brick*owl.com/catalog/lego-nxt-intelligent-*brick*-set-9841

[10] "LEGO® MINDSTORMS® EV3", Robotshop.com, 2016. [Online]. Disponible: http://www.robotshop.com/ca/en/lego-mindstorms-ev3-us.html

[11] "Historia", Arduino: Tecnología para todos, 2015. [Online]. Disponible: http://arduinodhtics.weebly.com/historia.html

[12] "Arduino - Home", Arduino.cc, 2015. [Online]. Disponible: https://www.arduino.cc/

[13] "Index of /DigitalControl.dir/LEGO MINDSTORMS NXT Hardware Developer Kit", Legolab.daimi.au.dk, 2015. [Online]. Disponible: http://www.legolab.daimi.au.dk/DigitalControl.dir/LEGO%20MINDSTORMS%20NXT%20Hardware%20Developer%20Kit/

[14] ”Programación Robot Mindstorm NXT – Ciclos Formativos IES Valle del Jerte – Plasencia”, Informatica.iesvalledeljerteplasencia.es, 2013. [Online]. Disponible: http://informatica.iesvalledeljerteplasencia.es/wordpress/programacion-robot-mindstorm-nxt/

[15] Bradley P.J., de la Puente J.A., Zamorano J., Ada User Guide for LEGMINDSTORMS NXT. Ada User Journal, 32, no. 3. September 2011.

[16] ”Mindstorms | Na miękko”, Namiekko.pl, 2015. [Online]. Disponible: http://namiekko.pl/tag/mindstorms/

[17] “Sensor lego a butia,” Sensor lego a butia - Proyecto Butiá, 2015. [Online]. Disponible: https://www.fing.edu.uy/inco/proyectos/butia/mediawiki/index.php/Sensor_lego_a_butia.

[18] ”LEGO NXT wire pinout”, Salvius the Robot, 2012. [Online]. Disponible: https://blog.salvius.org/2012/04/lego-nxt-wire-pinout.html

[19] López Perez, E., “Ingeniería en microcontroladores” [Online]. Disponible: http://www.electronica60norte.com/mwfls/pdf/rs-485.pdf

[20] ”LEGO Education - 9891 Sensor de rotacion - LEGO Education”, Electric*brick*s.com, 2017. [Online]. Disponible: https://www.electric*brick*s.com/lego-education-mindstorms-rcx-9891-sensor-rotacion-lego-education-p-303.html

[21]”— Использование датчика цвета (Color Sensor)”, Ks2.tom.ru, 2017. [Online]. Disponible: http://ks2.tom.ru/?p=3777

[22] “Sensor de tacto - 9843 | MINDSTORMS® | LEGO Shop”, Shop.lego.com, 2017. [Online]. Disponible: https://shop.lego.com/es-ES/Sensor-de-tacto-9843

[23] “Sensor de luz - 9844 | MINDSTORMS® | LEGO Shop”, Shop.lego.com, 2017. [Online]. Disponible: https://shop.lego.com/es-ES/Sensor-de-luz-9844

[24] “Sensor de sonido - 9845 | MINDSTORMS® | LEGO Shop”, Shop.lego.com, 2017. [Online]. Disponible: https://shop.lego.com/es-ES/Sensor-de-sonidos-9845

[25] “Sensor de ultrasonidos - 9846 | MINDSTORMS® | LEGO Shop”, Shop.lego.com, 2017. [Online]. Disponible: https://shop.lego.com/es-ES/Sensor-ultrasonico-9846

[26] "Interactive Servo Motor - 9842 | MINDSTORMS® | LEGO Shop", Shop.lego.com, 2014. [Online]. Disponible: https://shop.lego.com/en-AT/Interactive-Servo-Motor-9842

[27] NXT® motor internals, 2006. [Online]. Disponible: http://www.philohome.com/nxtmotor/nxtmotor.htm

[28] V. García (2009). "Automatismo para garaje con Arduino," Automatismo Electrónico, Arduino, micros, microcontroladores, proyectos técnicos. [Online]. Disponible: http://hispavila.com/total/3ds/atmega/automatismo1.html

[29] M. Gasperi and P. Hurbain (200, Extreme NXT: extending the LEGO MINDSTORMS NXT to the next level. Berkeley, CA: Apress, 2nd ed., 33, 2009

[30] "*Brick*Engineer: LEGO Design," *Brick*Engineer LEGO Design, 2008. [Online]. Disponible: http://www.*brick*engineer.com/pages/2008/09/05/lego-nxt-motor-wiring/

[31] "¿Qué es un encoder? ¿Cómo funciona? Y tipos de encoder que existen," ABM Industrial, 2013. [Online]. Disponible: http://www.abm-industrial.com/2013/02/07/que-es-un-encoder/

[32] "Lenguajes de programacion para NXT - electric*brick*s," -, 2009. [Online]. Disponible: http://blog.electric*brick*s.com/2009/09/lenguajes-de-programacion-para-nxt/

[33] "NXT-G: the development environment supplied with Lego Mindstorms, NXT-G," Génération Robots - Blog, 2016. [Online]. Disponible: http://www.generationrobots.com/blog/en/2015/09/nxt-g-the-development-environment-supplied-with-lego-mindstorms-nxt-g/

[34] "Mindstorms, Arduino y la conexión de sensores," Manejo del protocolo I2C usando ROBOTC, 2011. [Online]. Disponible: http://mindstormsyarduino.blogspot.com.es/2011/01/manejo-del-protocolo-i2c-usando-robotc.html

[35] "Programovací možnosti," ROBOTI | ČVUT - Katedra řídicí techniky. [Online]. Disponible: https://support.dce.felk.cvut.cz/roboti/index.php?id=programming

[36] Arduino Nano, 2017. [Online]. Disponible: https://store.arduino.cc/arduino-nano

[37] cny70, 2009. [Online]. Disponible: http://www.jmnlab.com/cny70/cny_70.html

[38] "Sensor de infrarrojos," Wiki de Robtica, 2016. [Online]. Disponible: http://wiki.robotica.webs.upv.es/wiki-de-robotica/sensores/sensores-proximidad/sensor-infrarrojos/.

[39] "Fotointerruptor Transmisivo, Fototransistor, Agujero Pasante, 3.1 mm, 1 mm, 60 mA, 6 V," Farnell element14, 2017. [Online]. Disponible: http://es.farnell.com/vishay/tcst1103/sensor-ptico-transmisivo/dp/1470057

[40] "Control Nunchuk Para Nintendo Wii Mando Wii U Consola - Bs. 133.330,00," Bs. 133.330,00 en Mercado Libre, 2017. [Online]. Disponible: https://articulo.mercadolibre.com.ve/MLV-461475282-control-nunchuk-para-nintendo-wii-mando-wii-u-consola-_JM.

[41] "ZX-NUNCHUK Wii-Nunchuk interface board" Robotshop, 2015. [Online]. Disponible: http://www.robotshop.com/media/files/PDF/inex-zx-nunchuck-datasheet.pdf

[42] "Motor con reductora y doble eje de salida," Motor con reductora y doble eje de salida - Complubot Shop, 2017. [Online]. Disponible: http://www.complubot.com/shop/index.php?id_product=1054&controller=product

[43] "L9110 Ventilator- / Propeller-Modul für Arduino," Roboter-Bausatz.de, 2017. [Online]. Disponible: https://www.roboter-bausatz.de/1189/l9110-ventilator/propeller-modul-fuer-arduino

[44] "Servo de RC tipo S3003," Servo de RC tipo S3003 - Complubot Shop, 2017. [Online]. Disponible: http://www.complubot.com/shop/index.php?id_product=928&controller=product

[45] "Servomotores" Teknomovo - teknomovo, 2016. [Online]. Disponible: http://teknomovo.mx/productos/servomotores.html

[46] Pita, M. (2007), "Familias Lógicas," Familias Lógicas (página 3) - Monografias.com. [Online]. Disponible: http://www.monografias.com/trabajos45/familias-logicas-electronica/familias-logicas-electronica3.shtml

[48] "Sharp distance sensors," ASkr's Projects, 2008. [Online].Disponible: http://www.askrprojects.net/lego/sharp.html

[49] "Configuraciones básicas del transistor", 2014. [Online]. Disponible: http://www.geocities.ws/pnavar2/transis2/colector.html

[50] Encoder - Robótica," Google Sites, 2010. [Online]. Disponible: https://sites.google.com/site/proyectosroboticos/encoder

[51] "Pololu - DRV8835 Dual Motor Driver Carrier," Pololu Robotics & Electronics, 2013. [Online]. Disponible: https://www.pololu.com/product/2135

[52] "Muebles de DM. Pero qué es el DM, y por qué se utiliza para hacer muebles.," Blog Mueblipedia.com, 2014. [Online]. Disponible: http://blog.mueblipedia.com/2013/04/08/muebles-de-dm-pero-que-es-el-dm-y-por-que-se-utiliza-para-hacer-muebles/

[53] "Sculpteo," Corte por láser definición, 2009. [Online]. Disponible: https://www.sculpteo.com/es/glosario/corte-por-laser-definicion/

[54] NXT Five Minute Bot, 2007. [Online]. Disponible: http://www.nxtprograms.com/five_minute_bot/

[55] "La impresión 3D: Que es y sus técnicas de impresión," Impresoras 3D, 2016. [Online]. Disponible: http://comunidad.iebschool.com/impresoras3d/2016/11/02/la-impresion-3d-que-es-y-sus-tecnicas-de-impresion/

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá