

TRABAJO FIN DE GRADO EN INGENIERIA ELECTRONICA INDUSTRIAL



INTERFAZ DE CONTROL CAN-BLUETOOTH PARA ROBOTS LEGO-NXT

AUTOR: Manuel Alejandro Fernández Caballero

TUTOR: Alejandro Linares Barranco

CURSO ACADÉMICO: 2016/2017





INTERFAZ DE CONTROL CAN-BLUETOOTH PARA ROBOTS LEGO-NXT

AUTOR: Manuel Alejandro Fernández Caballero

TUTOR: Alejandro Linares Barranco

CURSO ACADEMICO: 2016/2017

Dpto. Arquitectura y Tecnología de Computadores

Escuela Politécnica Superior

Universidad de Sevilla

Sevilla, Junio 2017



INDICE DE CONTENIDOS

DOCUMENTO 1. MEMORIA.....	6
CAPITULO 1. INTRODUCCIÓN	7
I. Resumen.....	7
II. Antecedentes	8
a. Historia y actualidad de la robótica.....	8
b. Lego Mindstorms NXT	10
c. Bluetooth.....	14
d. Lenguaje Basic	15
CAPITULO 2. MARCO NORMATIVO LEGAL	18
I. Legislación de ámbito europeo	18
II. Legislación de ámbito nacional	18
III. Legislación de ámbito autonómico	19
IV. Legislación de ámbito local	19
V. Normas	19
CAPITULO 3. TECNOLOGIAS EMPLEADAS.....	21
I. Bloque NXT.....	21
a. Servomotores.....	23
b. Sensores.....	24
c. Comunicaciones	32
II. HC-05.....	32
III. VISUAL STUDIO EXPRESS	34
a. Introducción	34
b. Descripción	34
c. Características	36
IV. MICROCONTROLADOR 8051.....	37
V. BUS-CAN	39
a. Introducción	39
b. Características	41
CAPITULO 4. DESARROLLO	43
I. PLACA C8051F040-BT	43
a. Introducción	43



b.	Características	44
c.	Programación del microcontrolador C8051F040.	45
d.	Diseño e implementación.....	46
i.	Reloj del sistema	49
ii.	Timer 2	49
iii.	UART0	50
iv.	CAN	54
v.	Puertos de Entrada/Salida	59
vi.	Interrupciones	62
II.	APLICACIÓN DE CONTROL PARA NXT	65
a.	Protocolo de comunicación del NXT.	65
b.	Diseño e implementación de Interfaz de control con Visual Studio	66
i.	Inserción y configuración de controles	67
ii.	Asignación de métodos y procedimientos a eventos.....	69
CAPITULO 5. CONCLUSIONES		88
I.	Conclusiones	88
CAPITULO 6. BIBLIOGRAFIA		89
I.	Bibliografía	89
DOCUMENTO 2. ANEXOS		90
ANEXO 1. MANUAL HC-05		91
ANEXO 2. MANUAL PLACA 8051		105
ANEXO 3. COMANDOS DIRECTOS LEGO NXT		120
DOCUMENTO 3. PLIEGO DE CONDICIONES.....		133
I.	Descripción general del proyecto.....	134
a.	Alcance general del presente proyecto.....	134
II.	Condiciones generales	135
a.	Condiciones generales de índoles facultativas.....	135
i.	Técnico director de obra (tutor).....	135
ii.	Constructor o instalador (Alumno)	136
iii.	Verificación de los documentos del proyecto	137
iv.	Plan de seguridad y salud en el trabajo.....	137
v.	Ampliación del proyecto por causas imprevistas o de fuerza mayor.....	138
vi.	Prorroga por causa de fuerza mayor	138
vii.	Condiciones generales de ejecución de los trabajos.....	138



b.	Condiciones generales de índole económicas	138
i.	Composición de los precios unitarios.....	138
ii.	Precio de contrata, importe de contrata.....	140
iii.	Precio contradictorio.....	140
iv.	Condiciones generales de índole legal	141
III.	Condiciones particulares del equipo industrial.....	141
a.	Prescripciones sobre materiales	141
b.	Prescripción en cuanto a la ejecución por unidad de obra	142
i.	Medidas para asegurar la compatibilidad entre los diferentes productos, elementos y sistemas constructivos que componen la unidad de obra	142
ii.	Características técnicas	143
iii.	Normas de aplicación	143
iv.	Criterio de mediciones en proyecto	143
v.	Proceso de ejecución.....	143
vi.	Condiciones previas.....	143
vii.	Ensayos y pruebas de servicio.....	144
viii.	Condiciones de terminación.....	144
ix.	Garantías de calidad.....	145
c.	Listado de materiales empleados.....	145
i.	Hardware.....	145
ii.	Software	145
d.	Requisitos	145
i.	Requisitos del hardware.....	145
ii.	Requisitos del software	145
DOCUMENTO 4.	MEDICIONES Y PRESUPUESTO	146
I.	PRESUPUESTO	147



DOCUMENTO 1

MEMORIA



CAPITULO 1. INTRODUCCIÓN

I. Resumen

Este proyecto trata de elaborar una interfaz de control CAN-Bluetooth para robots Lego NXT en el contexto de la titulación de Grado en Ingeniería Electrónica Industrial con este TFG para desarrollar con más profundidad las competencias de las asignaturas de Informática y Comunicaciones Industriales, y de Robótica Industrial.

Para este proyecto el primer paso es crear una aplicación de control en Visual Basic para PC, la cual envía la información a la UART de un microcontrolador 8051, debidamente programado para enviar la información recibida por la UART al CAN, a través de un cable RS-232.

Este microcontrolador transmite dicha información a otro microcontrolador 8051 a través del Bus-CAN, este segundo microcontrolador posee un dispositivo Bluetooth conectado a la UART de dicho microcontrolador permitiendo la comunicación inalámbrica con nuestro robot NXT. Esta comunicación es bidireccional, por lo que ambos microcontroladores están programados para enviar información de la UART al CAN y viceversa; a la vez de realizar la comunicación entre ambos microcontroladores a través de Bus-CAN.

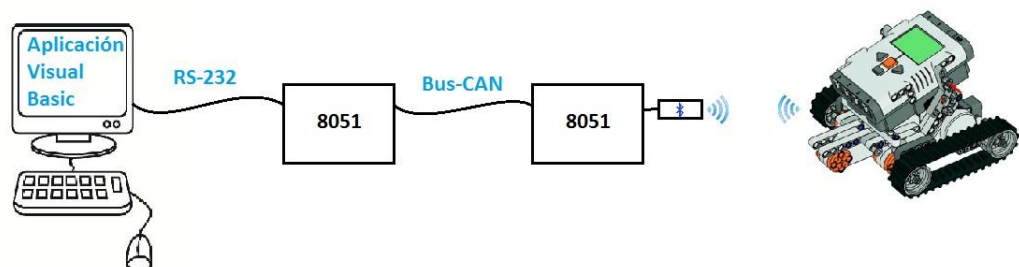


Figura 1: Esquema de comunicación industrial entre PC y Lego Mindstorms NXT.



II. Antecedentes

a. Historia y actualidad de la robótica

Por siglos, el ser humano ha construido máquinas que imitan partes del cuerpo humano. Los antiguos egipcios unieron brazos mecánicos a las estatuas de sus dioses; los griegos construyeron estatuas que operaban con sistemas hidráulicos, los cuales eran utilizados para fascinar a los adoradores de los templos.

El inicio de la robótica actual puede fijarse en la industria textil del siglo XVIII, cuando Joseph Jacquard inventa en 1801 una máquina textil programable mediante tarjetas perforadas. Luego, la Revolución Industrial impulsó el desarrollo de estos agentes mecánicos. Además de esto, durante los siglos XVII y XVIII en Europa fueron construidos muñecos mecánicos muy ingeniosos que tenían algunas características de robots. Jacques de Vaucansos construyó varios músicos de tamaño humano a mediados del siglo XVIII. En 1805, Henri Maillardert construyó una muñeca mecánica que era capaz de hacer dibujos. La palabra robot se utilizó por primera vez en 1920 en una obra llamada "Los Robots Universales de Rossum", escrita por el dramaturgo checo Karel Capek. Su trama trataba sobre un hombre que fabricó un robot y luego este último mata al hombre. La palabra checa 'Robota' significa servidumbre o trabajado forzado, y cuando se tradujo al inglés se convirtió en el término robot. Luego, Isaac Asimov comenzó en 1939 a contribuir con varias relaciones referidas a robots y a él se le atribuye el acuñamiento del término Robótica y con él surgen las denominaciones "Tres Leyes de Robótica" que son las siguientes:

1. Un robot no puede actuar contra un ser humano o, mediante la inacción, que un ser humano sufra daños.
2. Un robot debe de obedecer las órdenes dadas por los seres humanos, salvo que estén en conflictos con la primera ley.



3. Un robot debe proteger su propia existencia, a no ser que esté en conflicto con las dos primeras leyes.

Son varios los factores que intervienen para que se desarrollaran los primeros robots en la década de los 50's. La investigación en inteligencia artificial desarrolló maneras de emular el procesamiento de información humana con computadoras electrónicas e inventó una variedad de mecanismos para probar sus teorías. Las primeras patentes aparecieron en 1946 con los muy primitivos robots para traslado de maquinaria de Devol. También en ese año aparecen las primeras computadoras. En 1954, Devol diseña el primer robot programable. En 1960 se introdujo el primer robot "Unimate", basada en la transferencia de artículos.

1961 Un robot Unimate se instaló en la Ford Motors Company para atender una máquina de fundición de troquel. En 1966 Trallfa, una firma noruega, construyó e instaló un robot de pintura por pulverización. En 1971 El "Standford Arm", un pequeño brazo de robot de accionamiento eléctrico, se desarrolló en la Standford University. En 1978 Se introdujo el robot PUMA para tareas de montaje por Unimation, basándose en diseños obtenidos en un estudio de la General Motors.

Actualmente, el concepto de robótica ha evolucionado hacia los sistemas móviles autónomos, que son aquellos que son capaces de desenvolverse por sí mismos en entornos desconocidos y parcialmente cambiantes sin necesidad de supervisión. En los setenta, la NASA inicio un programa de cooperación con el Jet Propulsión Laboratory para desarrollar plataformas capaces de explorar terrenos hostiles.

En la actualidad, la robótica se debate entre modelos sumamente ambiciosos, como es el caso del IT, diseñado para expresar emociones, el COG, también conocido como el robot de cuatro sentidos, el famoso SOUJOURNER o el LUNAR ROVER, vehículo de turismo con control

remotos, y otros mucho más específicos como el, un helicóptero robot de uso militar, el guardia de tráfico japonés ANZEN TARO o los robots mascotas de Sony.

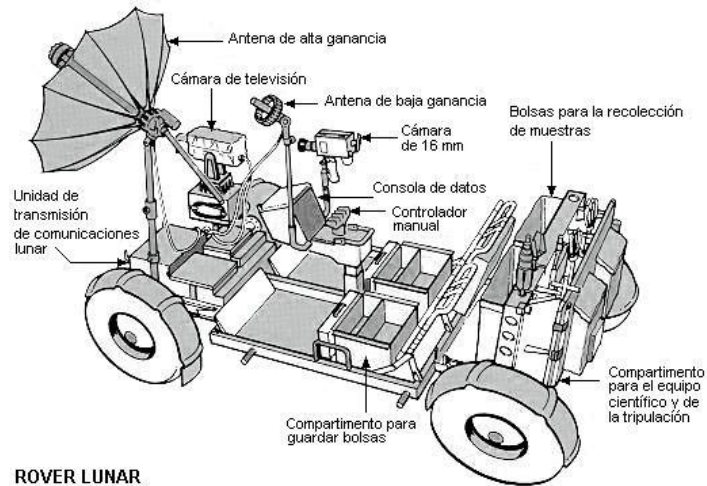


Figura 2: Rover Lunar

En general la historia de la robótica la podemos clasificar en cinco generaciones: las dos primeras, ya alcanzadas en los ochenta, incluían la gestión de tareas repetitivas con autonomía muy limitada. La tercera generación incluiría visión artificial, en lo cual se ha avanzado mucho en los ochenta y noventa. La cuarta incluye movilidad avanzada en exteriores e interiores y la quinta entraría en el dominio de la inteligencia artificial en lo cual se está trabajando actualmente.

b. Lego Mindstorms NXT

En los últimos años, LEGO ha invertido mucho en el campo de la robótica. La línea LEGO MINDSTORMS ha revolucionado el concepto de las personas con respecto a los LEGO. LEGO Mindstorms es un juego de robótica para niños, el cual posee elementos básicos de las teorías robóticas, como la unión de piezas y la programación de acciones, en forma interactiva. Este robot fue comercializado por primera vez en septiembre de 1998. La filosofía apunta a un deseo de hacer que la línea de productos LEGO MINDSTORMS amplíe el círculo de edad de



los fans de LEGO, para que lo utilicen tanto expertos en el campo de la robótica, como los principiantes en una edad joven.

LEGO ha dedicado un gran esfuerzo para crear un producto didáctico adaptado a las diferentes etapas de crecimiento del niño. La fuerza de la línea LEGO Mindstorms es que ofrece múltiples alternativas para construir un robot diseñado de acuerdo a su propia imaginación y sus deseos. La nueva generación de LEGO MINDSTORMS se basa en el éxito de la “Robotics Invention System” (en español RIS: Sistema de Invención Robotizado), reconocido en todo el mundo, pero es más rápido y fácil de usar. En pocas horas se puede construir y programar un robot de LEGO. Un claro ejemplo de este éxito en el mundo didáctico y de la robótica es la FLL (First Lego League), es un concurso mundial para estudiantes de primaria y secundaria organizado por la organización FIRST y LEGO Education. Los equipos del concurso también trabajan con los Lego Mindstorms kits para construir pequeños robots autónomos que deben cumplir con ciertos objetivos en campos de piezas Lego dadas. Cada año se centra en un tema distinto de ciencia o tecnología, por lo que todas las pruebas del concurso se centran en este tema. Su objetivo es fomentar el aprendizaje de las disciplinas STEM (acrónimo en inglés de Ciencia, Tecnología, Ingeniería y Matemáticas), mediante la metodología conocida como educación STEM. El interés de los usuarios se ha visto estimulado por las nuevas técnicas y la comercialización de sensores aún más eficiente. Hitechnic ha firmado un convenio de colaboración con LEGO que le permite producir y vender accesorios de hardware certificados por el grupo LEGO. Este acuerdo representa una apertura por parte de LEGO hacia la colaboración con otras empresas y para la transformación de software en código abierto. También Pall Myer Smith, ejecutivo de LEGO, mostró su entusiasmo por esta colaboración, porque lo considera una oportunidad de demostrar el potencial de la plataforma LEGO y aumentar la capacidad del producto para satisfacer las necesidades del usuario. En un artículo de la revista PC Magazine “LEGO Mindstorms NXT se ha definido como un producto líder en la tecnología, capaz de combinar la creatividad con la ciencia”.



Con respecto a su historia, LEGO Mindstorms fue uno de los resultados de la fructífera colaboración entre LEGO y el MIT. Esta asociación se emplea como ejemplo de relación entre la industria y la investigación académica que resulta muy beneficiosa para ambos socios. La línea Mindstorms no fue el primer fruto de esta relación, aunque si el más exitoso. Con anterioridad, LEGO se había interesado por el lenguaje de programación Logo. Fruto de este interés nació en 1986 Lego TC Logo, creado por Resnick y Steve Ocko. Lego TC Logo era un sistema en el que se programaba en una computadora que estaba conectada por un cable a una construcción LEGO que contaba con motores, luces y sensores. Aunque alcanzó un relativo éxito comercial, según Resnick el sistema imponía restricciones tanto físicas como imaginativas. El paso de programar una computadora que se conectaba a una construcción LEGO a programar un bloque de esa construcción era una idea natural que se estudió durante largo tiempo. Desde principios de los años 90 se empezó a investigar esta posibilidad. Hubo que esperar un lustro hasta que las condiciones fueran las apropiadas y decidieran empezar seriamente el desarrollo de lo que acabaría siendo el bloque RCX, un bloque de Lego que contaba con un microcontrolador, y que constituye el corazón del producto Mindstorms. De esta forma las construcciones LEGO pasaban de ser estructuras estáticas a máquinas dinámicas que interactúan con el mundo.

La primera versión salió al mercado con un precio de 200 dólares. Incluía 717 componentes, entre ellos el bloque RCX.



Figura 3: Bloque RCX



Tras su lanzamiento se vendieron 80.000 unidades en tres meses. Además, la comunidad de aficionados a la robótica, un público adulto, acogió con interés este nuevo producto. Este interés imprevisto del público adulto hizo que las ventas triplicaran las expectativas, así como numerosas páginas web de intercambio de ideas.

Además del bloque RCX, existieron otros bloques programables, los cuales gradualmente se fueron desarrollando hasta lograr la versión definitiva de la versión NXT. A partir de 1998 se comercializó el inicio de la línea con el robot Cybermaster. A principios de 2004, debido a los malos resultados de LEGO del año anterior, que registró unas pérdidas de unos 188 millones de euros, cundió el rumor de que abandonaría la línea Mindstorms y volvería a su mercado tradicional. Sin embargo, en enero de 2006 LEGO anunció la versión Mindstorms NXT, de última generación, con un microprocesador de 32 bits que puede programarse desde un ordenador, el cual empezó a comercializar en junio de ese mismo año.



Figura 4: Bloque NXT

El paquete básico incorpora diversos sensores, servomotores y el procesador. Resulta bastante fácil realizar pequeños montajes robóticos gracias a los componentes Lego y además Mathwork ofrece soporte para poder ser utilizado sobre Simulink de manera fácil. El problema de este sistema es el precio, el paquete inicial con todos estos sensores tiene un precio superior a los 380 €. Por lo tanto no cumple uno de los requerimientos principales del presente proyecto consistente en llevar a cabo una plataforma robótica de bajo coste.



c. Bluetooth

Bluetooth proviene de la palabra escandinava “Blåtand” que significa “hombre de tez oscura” pero en los tiempos que corren el significado original se ha perdido y ahora se asocia a las comunicaciones inalámbricas, un estándar global que posibilita la transmisión de voz, imágenes y en general datos entre diferentes dispositivos en un radio de corto alcance y lo que le hace más atractivo, muy bajo coste. Los principales objetivos que este estándar quiere lograr son:

- Facilitar las comunicaciones entre equipos.
- Eliminar cables y conectores entre aquéllos.
- Facilitar el intercambio de datos entre los equipos.



Figura 5: Logo Bluetooth

Bluetooth funciona bajo radio frecuencias pudiendo atravesar diferentes obstáculos para llegar a los dispositivos que tenga a su alcance.

Opera bajo la franja de frecuencias 2.4 – 2.48 GHz o como también es conocida como “Banda ISM” que significa “Industrial, Scientific and Medical” que es una banda libre para usada para investigar por los tres organismos anteriores. Pero esto tiene sus consecuencias, ya que al ser libre puede ser utilizada por cualquiera y para ello, para evitar las múltiples interferencias que se pudieran introducir (microondas, WLANs, mandos, etc.) Bluetooth utiliza una técnica denominada salto de frecuencias.

El funcionamiento es ir cambiando de frecuencia y mantenerse en cada una un “slot” de tiempo para después volver a saltar a otra diferente. Es conocido que entre salto y salto el tiempo que transcurre es muy



pequeño, concretamente unos 625 microsegundos con lo que al cabo de un segundo se puede haber cambiado 1600 veces de frecuencia.

Cuando coinciden más de un dispositivo Bluetooth en un mismo canal de transmisión se forma lo que se llaman “piconets” que son redes donde hay un maestro que es el que gestiona la comunicación de la red y establece su reloj y unos esclavos que escuchan al maestro y sincronizan su reloj con el del maestro.

Dicho alcance puede variar según la potencia a la que se transmite (a mayor potencia mayor consumo y menor autonomía para el dispositivo) y el número de repetidores que haya por el medio. Así el alcance puede estar entre los 10 y 100 metros de distancia (los repetidores provocan la introducción de distorsión que puede perjudicar los datos transmitidos).

Bluetooth puede conectar muchos tipos de aparatos sin necesidad de un solo cable, aportando una mayor libertad de movimiento. Por esta razón ya se ha convertido en una norma común mundial para la conexión inalámbrica. En el futuro, es probable que sea una norma utilizada en millones de teléfonos móviles, PC, ordenadores portátiles y varios tipos de aparatos electrónicos, como por ejemplo:

- Domótica (activación de alarmas, subida de persianas, etc.).
- Sector automovilístico (comunicación con otros vehículos).
- Medios de pago.

En una comunicación Bluetooth se pueden alcanzar tasas de transmisión de datos de 720 kbps (1 Mbps de capacidad bruta) con un rango óptimo de 10 metros (como se ha comentado antes 100 metros con repetidores).

d. Lenguaje Basic

El lenguaje de programación BASIC (Beginner's All purpose Symbolic Instruction Code) nació en el año 1964 como una herramienta destinado a principiantes, buscando una forma sencilla de realizar programas, empleando un lenguaje casi igual al usado en la vida ordinaria (en inglés), y con instrucciones muy sencillas y escasas.



Los autores fueron los científicos John G. Kemeny (Budapest, 1926 – USA1992) y Thomas E. Kurtz (Illinois 1928) su trabajo original se llamó True BASIC.

Inicialmente, Visual Basic fue pensado para ser un producto muy táctico. Microsoft tenía varias iniciativas en el desarrollo que lideraba Visual Basic 1.0, todas fueron pensadas para convertirse en las herramientas de programación a largo plazo, estratégicas, gráficas y orientadas a objetos.

La evolución del BASIC por los años 70 fue escasa, dado el auge que tomaron en aquella época lenguajes de alto nivel como el FORTRAN y el COBOL. En 1978 se definió una norma para unificar los Basics existentes creándose la normativa BASIC STANDARD.

Con la aparición de los primeros ordenadores personales, dedicados comercialmente al usuario particular, allá por la primera mitad de los ochenta, el BASIC resurgió como lenguaje de programación pensado para principiantes, y muchos de estos pequeños ordenadores domésticos lo usaban como único sistema operativo (Sinclair, Spectrum, Amstrad)

Con la aparición del Quick-BASIC de Microsoft, una versión ya potente del BASIC, que corregía casi todos los defectos de las versiones pasó prácticamente inadvertida, a no ser porque las últimas versiones del sistema operativo MS-DOS incluían una versión de Quick-BASIC algo recortada (Q-Basic). Esta versión del popular BASIC ya es un lenguaje estructurado, lo que permite crear programas modularmente, mediante subrutinas y módulos, capaz de crear programas ya competitivos con otros lenguajes de alto nivel.

Sin embargo algo había en el BASIC que tentaba a superarse: su gran sencillez de manejo. Si a esto se le añade el entorno gráfico Windows, el aprovechamiento al máximo de las posibilidades de Windows en cuanto a intercambio de información, de sus librerías, de sus drivers y controladores, manejo de bases de datos, etc. el producto resultante puede ser algo que satisfaga todas las necesidades de programación en el entorno Windows. La suma de todas estas cosas es VISUAL - BASIC.



Esta herramienta conserva del BASIC de los años 80 únicamente su nombre y su sencillez, y tras su lanzamiento al mercado, la aceptación a nivel profesional hizo borrar por fin el "mal nombre" asociado a la palabra BASIC.

El lenguaje BASIC se suele enseñar a los programadores principiantes porque es fácil de utilizar y de comprender y porque, en sus versiones más recientes, contiene muchos de los conceptos fundamentales de otros lenguajes considerados más complejos y técnicamente más potentes, como Pascal, C, C++ o JAVA.

En el año 2001 se comercializó la versión 6.0 de este producto. Desde su salida al mercado, cada versión supera y mejora la anterior. Dados los buenos resultados a nivel profesional de este producto, y el apoyo prestado por el fabricante para la formación de programadores, Visual-Basic se ha convertido en la primera herramienta de desarrollo de aplicaciones en entorno Windows.



CAPITULO 2. MARCO NORMATIVO LEGAL

I. Legislación de ámbito europeo

- Directiva 2002/96/CE del Parlamento Europeo y del Consejo, de 27 de Enero de 2003, sobre residuos de aparatos eléctricos y electrónicos.
- Directiva 2004/108/CE del Parlamento Europeo y del Consejo, de 15 de diciembre de 2004, relativa a la aproximación de las legislaciones de los Estados miembros en materia de compatibilidad electromagnética y por la que se deroga la Directiva 89/336/CEE.

II. Legislación de ámbito nacional

- Real Decreto 208/2005, de 25 de Febrero, sobre aparatos eléctricos y electrónicos y la gestión de sus residuos.
- Real Decreto 338/2010, de 19 de marzo, por el que se modifica el Reglamento de la Infraestructura para la calidad y seguridad industrial, aprobado por el Real Decreto 2200/1995, de 28 de diciembre.
- Real Decreto 411/1997, de 21 de marzo, por el que se modifica el Real Decreto 2200/1995, de 28 de diciembre, por el que se aprueba el Reglamento de la Infraestructura para la Calidad y Seguridad Industrial.
- Real Decreto 2200/1995, de 28 de diciembre, por el que se aprueba el Reglamento de la Infraestructura para la Calidad y la Seguridad Industrial.
- Real Decreto 486/1997, de 14 de abril, por el que se establecen las disposiciones mínimas de seguridad y salud en los lugares de trabajo.
- Ley 21/1992, de 16 de julio, de Industria.
- Ley 22/2011, de 28 de julio, de residuos y suelos contaminados.



III. Legislación de ámbito autonómico

- Decreto 73/2012, de 22 de marzo, por el que se aprueba el Reglamento de Residuos de Andalucía.
- Decreto 218/1999, de 26 de Octubre, por el que se aprueba el Plan Director Territorial de Gestión de Residuos Urbanos de Andalucía.
- Decreto 6/2012, de 17 de enero, por el que se aprueba el Reglamento de Protección contra la Contaminación Acústica en Andalucía, y se modifica el Decreto 357/2010, de 3 de agosto, por el que se aprueba el Reglamento para la Protección de la Calidad del Cielo Nocturno frente a la contaminación lumínica y el establecimiento de medidas de ahorro y eficiencia energética.

IV. Legislación de ámbito local

No se han encontrado documentos legislativos locales referentes al presente proyecto.

V. Normas

- UNE-EN 60950 Seguridad de los equipos de Tecnologías de la información.
- UNE-EN 55024 Características de inmunidad.
- UNE-EN 55022 Características de las perturbaciones radioeléctricas.
- UNE-EN 61000-4-14:2001/A1:2005 Compatibilidad electromagnética.
- UNE-EN ISO 10218 Robots y dispositivos robóticos.
- Normativa legal ISO 10218:1992. Creada en 1992 por el Organismo Internacional de Estandarización (ISO-92).
- Normativa americana ANSI/RIA R15.06-1992. Realizada por el Instituto Nacional de Normalización de los Estados Unidos (ANSI).



- Normativa europea EN 775 y española UNE-EN 775. Consiste en una adaptación de la norma ISO 10218:1992 por el Comité Europeo de Normalización (CEN).



CAPITULO 3. TECNOLOGIAS EMPLEADAS

I. Bloque NXT

LEGO Mindstorms es una línea de productos LEGO que combina los ladrillos tradicionales con sensores, actuadores, ladrillos programables y piezas de LEGOTECHNIC (como ejes, vigas y engranajes) para la construcción de robots y otros sistemas automatizados. Pueden modelarse todo tipo de sistemas electromecánicos con la línea Mindstorms.

El primer producto lanzado por LEGO en este campo fue el RCX (1998), el predecesor del actual NXT (2006): ambos definidos como ladrillo o bloque programable.

El bloque RCX tiene tres versiones oficiales. 1.0, 1.5 y 2.0, las cuales presentan mejoras en el software sin verse afectado mayormente el hardware que se vende con el bloque, sin embargo, la parte electrónica de los bloques no es compatible, ya que las tres versiones poseen distintas regulaciones de voltaje, pero aun así no afecta el hardware que posee el bloque. Su microcontrolador interno es Hitachi H8/3292, que funciona a 5 voltios y una velocidad aproximada de 16 MHz, siendo esa su velocidad máxima para la serie de Hitachi H8/3000. Posee una memoria ROM de 16 kb, una memoria RAM externa de 32 kb y posee un decodificador Análogo Digital que permite transformar las distintas entradas de energía en bits. El rendimiento de RCX es significativamente más bajo que el del NXT (calidad de sensores, velocidad, modo de comunicación, procesador...), pero su mayor desventaja comparado con el lego Mindstorms NXT es la baja capacidad de mantener hilos de proceso, es decir, no puede ejecutar dos instrucciones al mismo tiempo, a pesar que el programador o usuario compruebe que si puede, pero la velocidad de proceso impide distinguir el retardo producido.

Respecto a las entradas y salidas, posee tres conectores que permite capturar la información que proviene de los distintos sensores (en la parte superior de la pantalla de LCD, son de color gris y se distinguen por los números 1, 2 y 3). Las salidas del bloque RCX son para energizar los



motores que se pueden conectar al robot y así darle movimiento. El voltaje que provee es de 9 volts, haciendo que cada motor que se conecte al bloque puede moverse acorde a las instrucciones del programa (en la parte inferior de la pantalla de LCD, son de color y se distinguen por las letras A, B y C). También hay una pantalla LCD que muestra el estado de batería, el estado de puertos de E/S, el programa en ejecución y otra información (contador, temporizador, valores registrados por un sensor,...).

En la parte delantera del bloque RCX, el LEGO Mindstorms trae un puerto infrarrojo que le permite la comunicación con el computador para transferir el firmware y los programas.

La alimentación eléctrica del bloque es mediante 6 baterías AA de 1,5 volts, las cuales se conectan en la parte posterior del bloque. Las baterías se conectan en paralelo y proporcionan energía tanto al bloque como a los motores que se conectan al mismo bloque.



Figura 6: Bloque RXC con anotaciones

El bloque NXT es una versión mejorada a partir del LEGO Mindstorms RCX, que generalmente se considera la predecesora y precursora de los bloques programables de LEGO. Debido a la comercialización de los bloques programables, LEGO vendió la generación NXT en dos versiones: Retail Version y Education Base Set. Una ventaja de la versión educacional es que se incluían las baterías recargables y el cargador, pero esta misma versión debía comprar el software según el tipo de licencia: Personal, Sala de clases, Sitio.



El microcontrolador que posee es un ARM7 de 32 bits (AT91SAM7S256 Atmel), que incluye 256 Kb de memoria Flash y 64 Kb de RAM externa, la cual a diferencia del bloque RCX, posee mayores capacidades de ejecución de programas, evitando que los procesos inherentes de varios paquetes de datos colisionen y produzcan errores y un posible error en la ejecución del software.

En el bloque de NXT existen cuatro entradas para los sensores, que son incompatibles con las entradas de RCX; sin embargo, el paquete de NXT incluye el adaptador para que los sensores de RCX sean compatibles con NXT. Las salidas son tres localizadas en la parte posterior del bloque, para que la conexión de los motores y partes móviles tengan mejor acceso.

El bloque NXT puede comunicarse con el computador mediante la interfaz de USB o mediante Bluetooth. Además con la interfaz Bluetooth puede comunicarse con otros robots que estén en las cercanías.



Figura 7: Bloque NXT con sensores y actuadores

a. Servomotores.

Cada bloque NXT puede usar tres servomotores diferentes con un peso de 60 gramos cada uno, que permite moverse al robot. Los servomotores NXT están dotados de un sensor de rotación interno que permite controlar el movimiento con precisión. Estos motores pueden configurarse para que giren en un sentido un número determinado de



vueltas o a una velocidad determinada, la precisión que se puede conseguir es de 1° . También se pueden sincronizar varios motores a la misma velocidad; por ejemplo, utilizando bloque “mover” en el software de LEGO Mindstorms NXT del que hablaremos en el siguiente capítulo.



Figura 8: Vista externa e interna del servomotor NXT

Además de los motores especialmente diseñados para el NXT se pueden utilizar los motores Technic antiguos (motores RCX) y los nuevos motores Power Function (PF). Los motores PF son parte del nuevo sistema eléctrico de LEGO Technic. Estos motores vienen a sustituir a los antiguos motores Technic, ofreciendo un sistema de fácil montaje con las piezas Technic sin studs.

b. Sensores.

LEGO Mindstorms NXT es un sistema más abierto que el anterior, el RCX. Ofrece de entrada más sensores y la posibilidad de utilizar los sensores antiguos. Por otra parte, ofrece por medio de su web toda la información necesaria para desarrollar nuevos periféricos para el NXT. Consecuencia de ello es la oferta de nuevos sensores por parte de terceras empresas.

El NXT está provisto de 4 puertos para sensores con interfaz analogía y digital. Son compatibles con I2C y uno de ellos (el 4) es de alta velocidad.

Los sensores originales de LEGO que pueden encontrarse tanto en el set educativo como en el comercial son 4:

- Sensor contacto:

Este distingue entre dos estados, reconociendo cuando está presionado el botón. Tiene un orificio en la parte frontal que permite encajar ejes y otras piezas con la misma sección. Si se desea utilizar más señores de contacto que los puertos disponibles en el NXT, es posible utilizar un multiplexor. En la siguiente figura puede verse cuál es su funcionamiento interno:

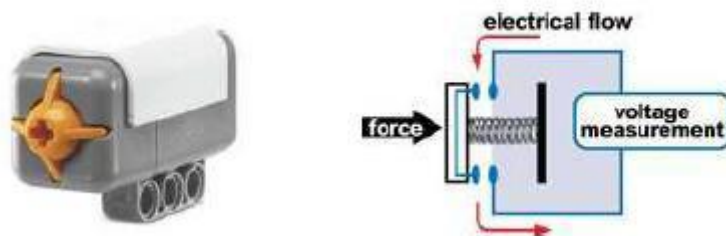


Figura 9: Sensor de contacto y descripción de su funcionamiento

- Sensor de luz:

Es uno de los dos sensores que aporta al robot el sentido de la vista (el otro será el sensor de ultrasonidos). El sensor de luz permite distinguir entre un ambiente iluminado y uno oscuro, midiendo la intensidad de luz presente. Tiene una sensibilidad superior al sensor de luz anterior y mide la luz con mayor precisión de 0 (oscuridad) hasta 100 (alto brillo). Permite apagar la luz infrarroja localizada bajo el sensor vía software, de tal modo que únicamente mida la luz ambiente de su entorno.

También es capaz de distinguir los colores midiendo la intensidad de la luz reflejada sobre la superficie de color (gracias a esta función un robot equipado con sensor de luz puede seguir una línea de color).



Otra característica del sensor consiste en calcular la distancia de la superficie de apoyo: en primer lugar mediante la emisión de una luz roja y, a continuación, midiendo la cantidad de luz reflejada. La extrema sensibilidad a la luz y la transferencia de una superficie a otra representa un obstáculo importante en el uso de este tipo de sensor. No se debe utilizar en habitaciones muy iluminadas debido a que el sensor no puede detectar las variaciones reales, y siempre estará en torno al valor máximo permitido.



Figura 10: Sensor de luz

- **Sensor de sonido:**
Este sensor permite programar robots que responden ante un nuevo estímulo como lo es el sonido. Mide el nivel de ruido en decibelio (dB) y en dBA (unidad de nivel sonoro medido con un filtro previo que quita parte de las baja y las muy altas frecuencias) para identificar las diferencias de tono y timbre. El sensor mide en dB con sensibilidad a todos los sonidos, teniendo en cuenta también los que no son perceptibles por el oído humano. Y en dBA la sensibilidad del sensor se adapta a la del oído humano.
Teniendo en cuenta la multiplicidad de sonidos audibles y la complejidad de los niveles de presión acústica, la lectura del sensor se muestra por pantalla del NXT en porcentaje:
 - 4-5%: habitación en silencio.
 - 5-10%: alguien que habla en la distancia.

- 10-30%: conversación normal o música con el volumen bajo.
- 30-100%: alguien que grita o música en volumen alto.



Figura 11: Sensor de sonido

- Sensor de ultrasonido:

Este es un sensor capaz de medir distancias (en pulgadas o centímetros) por medio de ultrasonidos. Emite ultrasonidos y espera su retorno (el cual se produce si encuentra algún obstáculo intermedio). Basándose en el tiempo transcurrido entre la emisión y la recepción puede determinar la distancia.

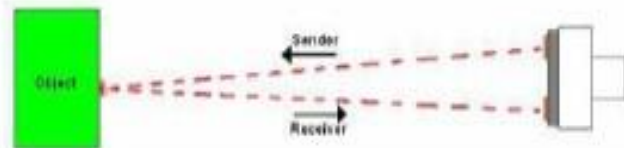


Figura 12: Funcionamiento del sensor de ultrasonidos

Con este sensor es más fácil montar robots que eviten obstáculos sin tener que chocar contra ellos o encontrar objetos sin tener que tocarlos previamente. Los mejores resultados los obtiene cuando el objeto está en frente, aunque también puede medir distancias a objetos en una posición lateral. Y las principales dificultades las encuentra con las superficies curvilíneas o pequeñas.

El rango de visibilidad de los sensores varía de 0 a 255 centímetros con una precisión de ± 3 centímetros. Se debe prestar especial atención a si hay más sensores ultrasónicos



que funcionan en la misma habitación ya que podrían interferir.



Figura 13: Sensor de ultrasonidos

Además de estos 4 sensores originales, existen varias empresas que comercializan sensores para el LEGO MINDSTORMS. Algunas de ellas han llegado a un acuerdo con LEGO que les permite acceder a los encapsulados oficiales de LEGO, con lo que la apariencia exterior es la misma que la de los sensores oficiales. La oferta va creciendo y en este momento el campo que abarca es muy amplio. Entre ellos se encuentra el giroscopio de HiTechnic que se utiliza en el NXTWay, que presentamos a continuación:

- **Giroscopio:**

Este sensor detecta la rotación alrededor de un eje y devuelve el valor de la velocidad angular en grados por segundo. El giroscopio puede llegar a medir hasta $\pm 360^\circ$ por segundo. La frecuencia de rotación se puede detectar alrededor de 300 veces por segundo. El eje de medición es el plano vertical, con el giroscopio como podemos observar en la siguiente

f
i
g
u
r
a
:

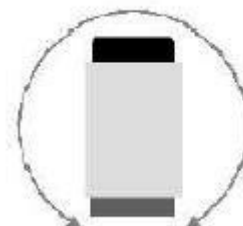


Figura 14: Giroscopio y eje de rotación

Otro sensor muy utilizado de Hitechnic, es el acelerómetro que permite medir la aceleración en diferentes ejes. Actualmente los sensores acelerómetros disponibles para el NXT son en todos los casos de tres ejes. Y todos ellos utilizan un protocolo de comunicaciones I2C compatible con el NXT.

- **Acelerómetro:**

Este sensor contiene un acelerómetro que mide la aceleración en los ejes x, y y z. Mide aceleraciones que se encuentren en el rango de -2g a +2g con una escala de aproximadamente 200 unidades por cada g (-200 a 200), es decir, con una sensibilidad de aproximadamente $0,05 \text{ m/s}^2$. El valor de la aceleración en cada eje se actualiza alrededor de 100 veces por segundo. Los tres ejes se asignan de la siguiente forma:



Figura 15: Acelerómetro y ejes

Otras empresas que comercializan sensores para LEGO Mindstorms son:

- **Codatex:** es una empresa austriaca que desarrolla elementos de "Identificación por radiofrecuencia" (RFID). Ha firmado un acuerdo con LEGO en virtud del cual accede a los encapsulados originales de LEGO Mindstorms NXT. Su producto estrella es el sensor identificador de radiofrecuencia RFID:



Figura 16: Sensor RFID

- Mindsensors: es una empresa que está desarrollando sensores y otros periféricos. Estos sensores no ocultan los componentes electrónicos que quedan a la vista y su oferta es muy variada. Es una de las empresas que en su momento comercializaban periféricos para el RCX. Entre sus productos destaca el sensor de distancia por infrarrojos (IR) para detección de obstáculos:



Figura 17: Sensor de distancia IR

- TecnoStuff: desarrolla sensores compatibles tanto para el NXT como para el RCX. Para el NXT dispone de un sensor de presión de aire (para sistemas neumáticos), un sensor de movimiento (detecta el movimiento de personas), un sensor de llama IR que detecta la llama de luz infrarroja (de calor radiante) y otro de llama UV.

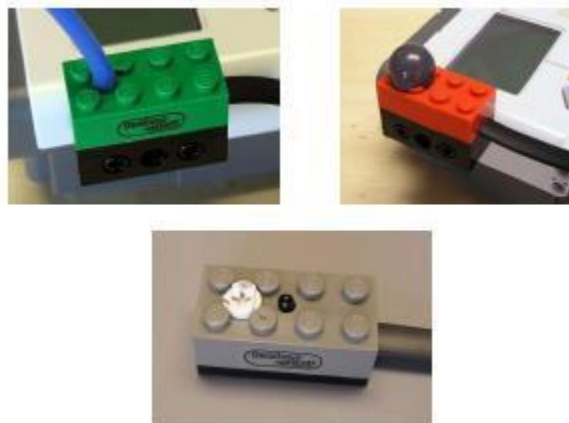


Figura 18: Sensor de presión, de movimiento y de llama IR

- Vernier: es una empresa que desarrolla sistemas de adquisición de datos para educación y que ha desarrollado un adaptador para poder utilizar sus sensores con el NXT. Con este adaptador se pueden utilizar hasta 30 sensores diferentes del catálogo de Vernier. El adaptador de Vernier está encapsulado del mismo modo que los originales de LEGO.



Figura 19: Adaptador para sensores Vernier



c. Comunicaciones

El bloque de NXT puede comunicarse con el computador mediante la interfaz de USB que posee, la cual ya viene en la versión 2.0. Además, puede comunicarse con otros robots en las cercanías, siendo posible establecer comunicaciones entre un NXT maestro y hasta tres esclavos. Para ello posee una interfaz Bluetooth que es compatible con la Clase II v 2.0. Esta conectividad con Bluetooth no tan solo permite conectarse con otros bloques, sino también con ordenadores, palms, teléfonos móviles, y otros aparatos con esta interfaz de comunicación. Dentro de las posibilidades de conexión se encuentran:

- Conectar hasta tres dispositivos distintos.
- Buscar y conectarse a otros dispositivos que posean Bluetooth.
- Recordar dispositivos con los cuales se ha conectado anteriormente para conectarse más rápidamente.
- Establecer el bloque NXT como visible o invisible para el resto de los dispositivos.

II. HC-05

Como conector Bluetooth se eligió en principio el modelo HC-06 con el cual se realizaba la comunicación en sentido PC-NXT pero había problemas a la hora de recibir los datos del NXT, tras varias pruebas e investigación se comprobó que el dispositivo HC-06 solo puede ejercer como dispositivo esclavo, por tanto había que buscar un dispositivo que se pudiese configurar como modo maestro y fue así como se eligió el modelo HC-05. A diferencia del HC-06 que solo dispone de 4 pines (GND, VCC, RX y TX) el HC-05 puede configurarse como maestro o esclavo además de otras muchas características que pueden modificarse a través de comandos AT como se verá más adelante. El HC-05 se trata de un conector Bluetooth bastante económico. Dispone de un regulador de 3.3 v por lo que se puede conectar directamente a los 5 v de la placa del 8051.



Figura 20: Dispositivo HC-06

El módulo de Bluetooth HC-05 es el que ofrece una mejor relación de precio y características, ya que es un módulo Maestro-Esclavo, quiere decir que además de recibir conexiones desde una PC o Tablet, también es capaz de generar conexiones hacia otros dispositivos Bluetooth.

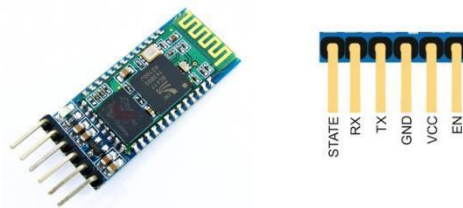


Figura 21: Dispositivo HC-05 y pines

Como se observa en la figura previa, solo dispone de 6 pines, de los cuales solo se utilizarán 5, por un lado VCC que se conectará a los 5V del 8051, por otro GND que irá al pin GND del 8051, después TXD que irá conectado al pin RXD de la placa y RXD que irá al pin TXD de la placa y por último el HC-05 tiene un modo de comandos AT que debe activarse mediante un estado alto en el PIN34 (Pin “KEY”) mientras se enciende (o se resetea) el módulo. Una vez que estamos en el modo de comandos AT, podemos configurar el módulo Bluetooth y cambiar parámetros como el nombre del dispositivo, password, modo maestro/esclavo, se puede configurar incluso para que se conecte a ciertos dispositivos indicándole la dirección de Bluetooth de este, si queremos que este visible o no, etc...

En el Anexo 1 se incluye un manual completo del HC-05, donde aparecen todas sus características y comandos AT.



III. VISUAL STUDIO EXPRESS

a. Introducción

Para el desarrollo de nuestra aplicación en lenguaje Basic se ha elegido Microsoft Visual Studio Express Edition que es un programa de desarrollo en entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows desarrollado y distribuido por Microsoft Corporation. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros. Es de carácter gratuito y es proporcionado por la compañía Microsoft Corporation orientándose a principiantes, estudiantes y aficionados de la programación web y de aplicaciones, ofreciéndose dicha aplicación a partir de la versión 2005 de Microsoft Visual Studio.



Figura 22: Logo Visual Studio

b. Descripción

Visual Studio Express permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .net 2002, se incorpora la versión Framework 3.5, Framework 4.0 y Framework 4.5 para las ediciones 2005, 2008, 2010 y 2012). Así se pueden crear aplicaciones que se intercomunican entre estaciones de trabajo, páginas web y dispositivos móviles. Cabe destacar que estas ediciones son iguales al entorno de desarrollo comercial de Visual Studio



Professional pero sin características avanzadas. Las ediciones que hay dentro de cada suite son:

- Visual Studio 2013 Express for Web
- Visual Studio 2013 Express for Windows
- Visual Studio 2013 Express for Windows Desktop

Visual Estudio Express consta de los siguientes productos separados:

- Visual Basic Express
- Visual Web Developer Express
- Visual C++ Express
- Visual C# Express
- SQL Server Express
- Express for Windows Phone

Visual Basic Express proporciona un entorno de desarrollo totalmente funcional a los programadores noveles y aficionados que desean generar aplicaciones de formularios Windows Forms, aplicaciones cliente de Windows Presentation Foundation, aplicaciones de explorador de Windows Presentation Foundation, aplicaciones de consola y bibliotecas de clases. Visual Basic Express es la elección perfecta para los programadores noveles que están interesados en aprender a programar en el lenguaje Visual Basic.

Adicionalmente, Microsoft ha puesto gratuitamente a disposición de todo el mundo una versión reducida de Microsoft SQL Server llamada **SQL Server Express Edition** cuyas principales limitaciones son que no soporta bases de datos superiores a 10 GB de tamaño, únicamente utiliza un procesador y 1 Gb de RAM y no cuenta con el Agente de SQL Server.

En el pasado se incluyeron los siguientes productos, actualmente desaparecidos en versiones como Visual Studio Express 2005, 2008 y 2012:



- Visual InterDev.
- Visual J++.
- Visual FoxPro.
- Visual SourceSafe.

c. Características

Para la creación de la interfaz del usuario con el robot se usará Visual Studio. Esta permite aprovechar las ventajas de Windows con diseñadores XAML, un IDE productivo y una gran variedad de lenguajes de programación, incluidos C#, Visual Basic y C++. Permite elegir entre Windows Presentation Foundation (WPF), Windows Forms y Win32 para dirigir la aplicación al escritorio de Windows con la tecnología adecuada para la aplicación y sus conocimientos.

Para este proyecto se usará el Windows Forms que se incluye en la versión gratuita Visual Studio Express 2013 para escritorio de Microsoft. El Windows Forms es la interfaz de programación de aplicación gráfica (API) que se incluye como parte de Microsoft .NET Framework. Esta proporciona acceso a los elementos de la interfaz de Microsoft Windows nativas envolviendo la API de Windows existente en código administrado. La programación del código se hará con el lenguaje de programación Visual Basic (VB).

Requisitos del sistema:

- Sistemas operativos compatibles:
 - Windows 7 SP1 (x86 y x64)
 - Windows 8 (x86 y x64)
 - Windows 8.1 (x86 y x64)
 - Windows Server 2008 R2 SP1(x64)
 - Windows Server 2012 (x64)
 - Windows Server 2012 R2 (x64)
- Requisitos del hardware:
 - Procesador a 1,6 GHz o superior.



- 1GB de RAM (1,5 GB si se ejecuta en una máquina virtual)
- 5GB de espacio disponible en el disco duro.
- Disco duro de 5400 RPM
- Tarjeta de video compatible con DirectX 9 con resolución de pantalla de 1024 x 768 o más.

IV. MICROCONTROLADOR 8051.

Éste microcontrolador está basado en la Arquitectura Harvard (es decir, existen espacios de direcciones separados para código y datos). Aunque originariamente fue diseñado para aplicaciones simples, se permite direccionar 64 KB de ROM externa y 64 KB de RAM por medio de líneas separadas, chip select, para programa y datos.

Adicionalmente, el microcontrolador contiene una memoria interna, dividida en dos partes: los SFR y memoria de propósito general. Los SFR (Special Function Registers), son los registros proporcionados por el microcontrolador, y tienen asignadas direcciones en esta memoria interna. El acceso a esta memoria interna es más rápido que el acceso a la memoria externa, pero es de tamaño limitado. Parte de esta memoria interna además se usa como pila durante las llamadas a función y el proceso de interrupciones.



Figura 23: Intel 8051

Una característica particular del 8051 es la inclusión de una unidad de proceso booleano que permite que operaciones de nivel de bit lógica booleana se ejecuten directa y eficientemente en registros internos. Esto ha hecho que el 8051 sea muy popular en aplicaciones de control industrial.



Otra característica muy valorada es que tiene cuatro conjuntos separados de registros. A menudo se usa esta característica para reducir la latencia de interrupción. (La rutina que maneja la interrupción declara usar otro conjunto de registros, evitándose de esta manera tener que salvar en la pila los registros originales).

La familia de $\mu\text{C-8051}$ es variada, y se encuentra en diversas presentaciones, la selección de uno o de otro tipo de microcontrolador dependerá principalmente de las necesidades a satisfacer. El 8051 está basado en los microprocesadores de 8 bits, contiene internamente un CPU de 8bits, 3 puertos de entrada y salida paralelos, un puerto de control, el cual a su vez contiene; un puerto serie, dos entradas para Timer/Contador de 16 bits, dos entradas para interrupciones externas, las señales de RD y WR para la toma o almacenamiento de datos externos en RAM, la señal de PSEN para la lectura de instrucciones almacenadas en EPROM externa. Gracias a estas tres señales el $\mu\text{C} - 8051$ puede direcciones 64 K de programa y 64K de datos separadamente, es decir, un total de 128Kb. Además cuenta con 128 bytes de memoria RAM interna.

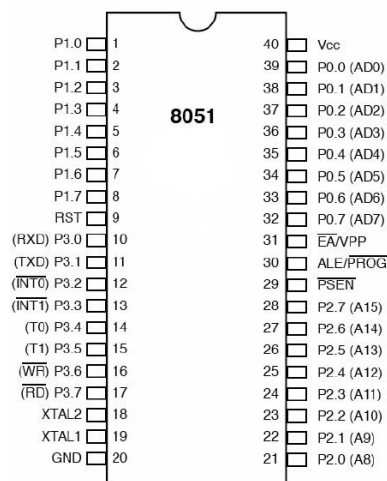


Figura 24: PINOUTS del microcontrolador 8051

Además el $\mu\text{C-8051}$ puede generar la frecuencia (Baud Rate) de Transmisión/Recepción de datos por el puerto serie de manera automática partiendo de la frecuencia del oscilador general, por medio de la programación del Timer 1. Dicha frecuencia de transmisión puede ser cambiada den cualquier momento con solo cambiar el valor de



almacenado en el control o también se puede duplicar o dividir la frecuencia con solo escribir sobre el bit 7 (SMOD) del registro de control (PCON).

El 8051 contiene las siguientes características:

- 1 CPU de 8 bits como parte central.
- 32 líneas bidireccionales de entrada y salida (4 puertos).
- 128 bytes de memoria RAM.
- 2 Controladores / Timers de 16 bits.
- 1 UART completa.
- 5 estructuras de interrupción con dos niveles de prioridad.
- 1 circuito de reloj.
- 64 Kbytes de espacio para programa.
- 64 Kbytes de espacio para datos.

V. BUS-CAN

a. Introducción

Ante el incremento del número de dispositivos electrónicos en sistemas como el de los automóviles, control de motores, domótica, telemetría, etc... , las necesidades de cableado y su complejidad aumentaron. Pronto se vio la posibilidad de conectar todos los dispersivos a un bus que debía de ser fiable, robusto, de alta inmunidad al ruido, etc. Además, el bus debía poder permitir altas velocidades de transmisión en entornos difíciles por la temperatura, vibraciones, interferencias, etc.

CAN ("Controler Area Network"), es un bus serie patentado por la compañía Robert Bosch (1982). Inicialmente se pensó en el cómo bus de campo, pero donde realmente encontró utilidad fue en el sector del automóvil, para interconectar el bus de confort, seguridad, etc. Fue diseñado para permitir la comunicación fiable entre centralitas electrónicas basadas en microprocesador, ECUs ("Electronic Control Unit") y reducir cableado. En Europa se ha convertido en un estándar

“de facto”, con carácter internacional y documentado por normas ISO (ISO-11898).

El bus CAN es un protocolo serie asíncrono del tipo CSMA/CD (“Carrier Sense Multiple Access with Collision Detection”). El bus es un medio compartido (multiplexado), se trata de un protocolo “Multicast”, es decir, todo el mundo puede hablar (de uno en uno) y escuchar. “CSMA” quiere decir que cada nodo de la red debe monitorizar el bus y si detecta que no hay actividad puede enviar un mensaje, y “CD” si 2 nodos de la red comienzan a transmitir un mensaje, ambos detectan colisión. Un método de arbitración basado en prioridades resuelve el conflicto.

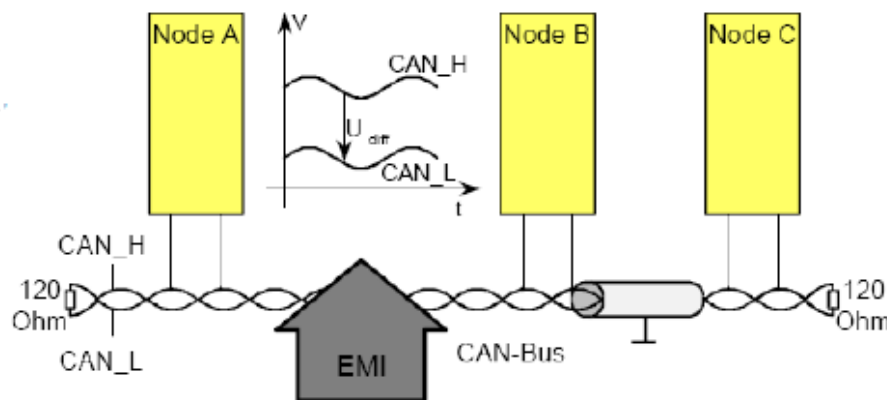


Figura 25; Capa física del BUS CAN

Se utilizan un par de cables trenzados (bus diferencial) para conseguir alta inmunidad a las interferencias electromagnéticas (EMIs), en algunos casos puede ir apantallado. Como la impedancia característica de esta línea es del orden de 120Ω , se emplean resistencias de este valor en ambos extremos del bus para evitar ondas reflejadas y evitar así que el bus se convierta en una antena.

La familia C8051F04x cuenta con una red de área de control (CAN). El CAN es un controlador que permite la comunicación en serie utilizando el protocolo CAN. El controlador CAN consiste en una CAN Core, RAM Mensaje (separado de la RAM CIP-51), una máquina de estado controlador de mensajes y registros de control. CAN es un controlador



de protocolo y no proporciona controladores de la capa física, es decir, transmisores-receptores.

b. Características

- CAN: Controller Area Network (red de área de control).
- Protocolo de comunicación serie tiempo real con un nivel alto de seguridad, bajo coste, bus de datos multiplexado (todos los nodos comparten el medio).
- Los mensajes son globales, cada receptor identificará el mensaje y lo procesará si la información le es útil o va destinada a él.
- Su funcionalidad cubre desde aplicaciones que requieran una red de comunicación de alta velocidad hasta aquellas de bajo coste.
- CAN actualmente predomina en los controles embebidos en automóviles, sistemas de control de motores, domótica, telemetría, etc.
- La figura muestra la descomposición de capas de la especificación CAN frente a las capas del modelo de referencia OSI.

Las características principales del bus CAN son:

- Prioridad en el envío y recepción de mensajes.
- Garantizar la sincronización de datos.
- Multicast en recepción sincronizada.
- Amplia consistencia de datos en el sistema.
- Sistema Multi-master.
- Detección de errores e identificación de los mismos.
- Retransmisión automática de los datos corruptos tan pronto como el bus vuelva a estar disponible.
- Distinción entre errores temporales y errores permanentes en el bus causados por un nodo defectuoso, los cuales de manera autónoma se excluyen de la red para permitir su funcionamiento.

El controlador CAN opera a velocidades de bits de hasta 1 Mbit/segundo, aunque esto puede ser limitado por la capa física elegida



para transmitir datos sobre el bus CAN. El controlador CAN cuenta con 32 mensajes objetos que se pueden configurar para transmitir o recibir datos. Los datos de entrada, el mensaje objetos y sus máscaras de identificadores se almacenan en la memoria RAM mensaje CAN. Todas las funciones de protocolo para la transmisión de datos y filtrado de la aceptación se realiza mediante el controlador CAN, de esta manera, se necesita un ancho de banda mínimo de la CPU para utilizar la comunicación CAN. El CIP-51 configura el controlador CAN, accede a los datos recibidos, y pasa los datos para su transmisión a través de los Registros de funciones especiales (SFR) en el CIP-51.





b. Características

El C8051F040-TB, altamente integrado, de señal mixta de 8 bits del microcontrolador (MCU) cuenta con dos sub-sistemas ADC de alta velocidad separados (ya sea 10/12-bit, 100 kSPS u 8 bits, 500 kSPS). Estos microcontroladores analógicos también incluyen dos DAC de 12 bits separados y un ± 60 VPGA (amplificador de ganancia programable). El C8051F04X contiene un monitor en el chip VDD, temporizador de vigilancia y un oscilador de reloj. Con estos componentes los microcontroladores de esta familia tienen soluciones autónomas de System-on-a-Chip. Todos los periféricos analógicos y digitales se pueden activar / desactivar y configurar mediante firmware usuario. La memoria flash se puede reprogramar incluso en circuito, proporcionando almacenamiento de datos no volátil, y que también permite actualizaciones del 8051 firmware.

Hemos seleccionado el microcontrolador C8051F040-TB de la empresa Silicon Labs, para el desarrollo del presente proyecto, porque se ha tenido en cuenta tanto los requisitos hardware como las herramientas y útiles de las que disponemos. A parte de lo expuesto en el párrafo anterior, se ha seleccionado porque es el microcontrolador que reúne todas las características necesarias para la realización del proyecto y para los objetivos del mismo.

Las características principales del microcontrolador son las que se muestran a continuación:

- Microcontrolador compatible 8051 con pipeline de alta velocidad (hasta 25 MIPS).
- Controlador bus CAN 2.0B con 32 mensajes, cada uno con su máscara de identificación.
- Interfaz de depuración integrada y de alta velocidad.
- ADC de 12 bits, 100ksps, 8 canales, ganancia programable y con multiplexor analógico integrado.



- ADC de 8 bits, 500ksps, 8 canales, ganancia programable y multiplexor analógico integrado.
- 2x DCA de 12 bits
- Amplificador diferencial de entrada de alto voltaje (60v pico a pico) y ganancia programable.
- 64K de memoria Flash integrada.
- 4096 + 256 bytes de RAM interna.
- Interfaz de Memoria RAM externa de hasta 64Kb de espacio direccionable.
- SMBus / I2C, 2xUART, SPI integrados.
- 5 timers de propósito general.
- PCA: contador / timer programable con 6 capturadores / comparadores y Watch Dog.
- Watch-Dog timer, Vdd Monitor y Sensor de temperatura.
- 64 puertos I/O tolerantes 5v.

c. Programación del microcontrolador C8051F040.

Para la programación del 8051 se utiliza el lenguaje de programación C, este lenguaje es un lenguaje de propósito general asociado al sistema operativo UNIX.

Es un lenguaje de medio nivel. Trata con objetos básicos como caracteres, números...; también con bits y direcciones de memoria. Se utiliza para la programación de sistemas. El lenguaje C consta del lenguaje C propiamente dicho y extensiones en forma de macros y un amplio conjunto de librerías predefinidas.

La compilación del software y su introducción en la placa C8051F040 se hace a través del entorno de desarrollo de Silabs cuyo aspecto se puede ver en la siguiente figura:

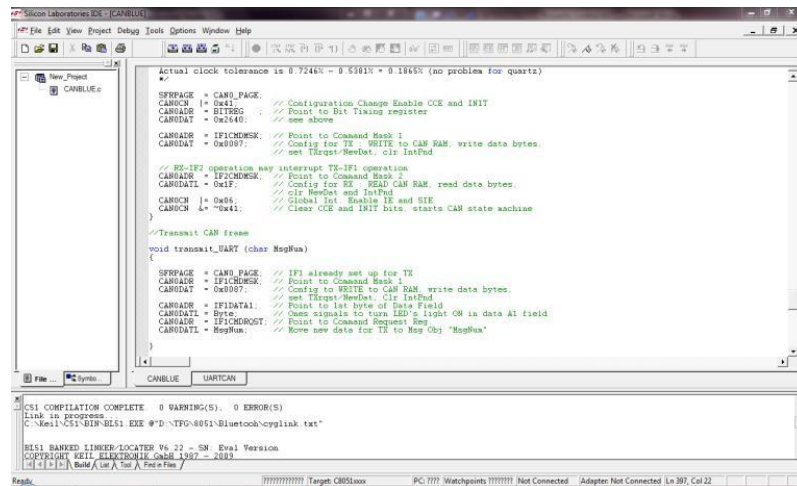


Figura 27: Entorno de desarrollo de Silabs

Usando este entorno se escribe el software necesario, se verifica que esté libre de errores y se compila cargándolo a la plataforma física, en este caso la placa C8051F040, conectando el puerto JTAG de la placa a un puerto USB del ordenador a través del USB Debug Adapter como se observa en la siguiente figura. Todas las características físicas de la placa y la descripción de todos los pines vienen descritas en el Anexo 2.



Figura 28: Debug Adapter.

d. Diseño e implementación.

Para la comunicación de nuestro PC con el robot NXT se ha utilizado la comunicación a través del BUS-CAN entre dos placas C8051F040. El flujo de información hasta el robot NXT sería la siguiente:



1. Comunicación serie desde el PC hasta la UART de una de las placas a través de un cable RS-232.
2. Transmisión de la información de la UART a través del BUS-CAN hasta la siguiente placa.
3. Recepción de la información a través del BUS-CAN y transmisión a la UART de la segunda placa.
4. Envío de la información recibida en la UART a través de un dispositivo Bluetooth (HC-05), el cual se conecta con nuestro robot NXT, este dispositivo se configura mediante la aplicación creada en VB como se verá más adelante.

Para poder realizar este proceso se han tenido que programar ambas placas para que realicen dos procesos, enviar la información recibida por la UART a través del BUS-CAN y viceversa

El programa se divide en cinco bloques distintos:

- **Bloque de declaración de variables, registros y funciones:**

Es el primer bloque y permite la definición de las diversas variables, los registros y funciones que se van a emplear, además de las bibliotecas a utilizar.

- **Bloque principal:**

Es el bloque donde se encuentra el bucle en el que se mantiene el programa mientras no haya ninguna interrupción, y donde se inicializan todas las funciones de nuestro programa.

- **Bloque Set Up:**

Bloque en el que se definen los pines físicos de la tarjeta que se utilizarán como entrada y salida, además de la configuración de los osciladores y la UART. Este bloque solo se ejecuta una vez, al iniciarse el programa por primera vez.



- **Bloque Can:**

Este bloque configura e inicializa el CAN, además contiene las funciones que pasan la información de una placa a otra y la cargan en la UART.

- **Bloque interrupciones:**

En este bloque encontramos las dos interrupciones a través de las cuales funciona nuestro programa, una para el CAN y otra para la UART.

Las funciones desarrolladas en los bloques anteriores son las siguientes:

FUNCIONES INTERNAS DEL MICROCONTROLADOR C8051F040-TB		
MÓDULOS	ELEMENTOS	CONFIGURACIÓN
Reloj (Bloque Set Up)	Cristal externo 22.1184 Mhz.	SYSCLK = 22.1184 Mhz.
Temporizador (Bloque Set Up)	Timer 2	Configurado en temporizador de auto-recarga, el modo de temporizador de 16 bits. La interrupción se produce una vez que ha terminado la cuenta. Frecuencia Timer 2: SYSCLK = 22.1184 Mhz.
UART (Bloque Set Up)	UART0	Configurado: Modo 1 Velocidad de transmisión variable de 8 bits. Utiliza el Timer 2 para la transmisión y recepción de datos. Noveno bit ignorado; RX habilitado.
Puertos (Bloque Set Up)	XBR0 XBR1 XBR2 XBR3 P0MDOUT P1MDOUT	Configuración de los puertos para habilitar los dispositivos que vamos a utilizar en el presente proyecto.
CAN (Bloque CAN)	Mensajes Objetos. Transmisión. Recepción.	Configuración Mensaje Objeto: Recepción y transmisión de datos.
Interrupciones (Bloque Interrupciones)		Interrupción de la UART y CAN

Tabla 1: Módulos internos del microcontrolador C8051F040-TB



i. Reloj del sistema

En el módulo del reloj se ha optado por un cristal externo por su precisión y una velocidad relativamente alta para satisfacer los requerimientos de tiempos de los buses que se deberán emplear así como, las tareas que deberá desempeñar el microcontrolador. Dicha selección, nos permite cierto margen, en un futuro, a la hora de ampliar o mejorar el prototipo del presente proyecto.

REGISTRO	CONFIGURACIÓN
OSCICN	0x80
CLKSEL	0x00
OSCXCN	0x67
CLKSEL	0x01

Tabla 2: Registros y configuración del reloj del microcontrolador.

ii. Timer 2

El microcontrolador C8051F040-TB incluye 5 contadores/temporizadores (Timers):

- Timer 0 y 1 son contadores/temporizadores de 16 bits compatibles con los encontrados en la norma 8051. El Timer 0 y el Timer 1 son casi idénticos y tienen cuatro modos principales de funcionamiento.
- El Timer 2, el Timer 3 y el Timer 4 son contadores/temporizadores de 16 bits de auto-recarga y captura. Se suelen usar con el ADC, DAC de la generación de onda cuadrada o para uso de propósito general. Estos temporizadores pueden ser usado para medir intervalos de tiempo, contar eventos externos y generar peticiones de interrupción periódicas. Los temporizadores 2, 3, y 4 son idénticos y no sólo ofrecen 16 bits de auto-recarga y la captura, también tienen la capacidad de producir un ciclo de trabajo del 50% de onda cuadrada en un pin de puerto externo.



MODOS TIMER 0 Y TIMER 1	MODOS TIMER 2, TIMER 3 Y TIMER 4
13-bit del contador/temporizador	16-bit del contador/temporizador con auto-recarga.
16-bit del contador/temporizador	16-bit del contador/temporizador con captura.
8-bit del contador/temporizador con auto-recarga	Conmutar salida.
Dos 8-bit contador/temporizador (solo para Timer 0)	

Tabla 3: Modos de funcionamientos del Timer 0 y 1.

Los temporizadores 2, 3 o 4 pueden utilizarse para generar velocidades de transmisión para UART0. En el desarrollo del presente proyecto hemos utilizado el Timer 2 con la siguiente configuración:

REGISTRO	CONFIGURACIÓN
TMR2CN	0x00
TMR2CF	0x08
RCAP2	- ((long)SYSTEMCLOCK/BAUDRATE/16);
TMR2	RCAP2
TR2	1

Tabla 4: Registros de configuración del Timer 2

iii. UART0

UART0 es un puerto serie mejorado con la detección de errores de trama y el hardware de reconocimiento de dirección.

UART0 puede operar en los modos asíncronos o semidúplex full-dúplex síncronos y la comunicación multiprocesador es totalmente compatible. Cuando la UART0 recibe los datos, estos se almacenan temporalmente en un registro de la explotación, lo que permite a la UART0 iniciar la



recepción de un segundo byte de datos de entrada antes que el programa haya terminado de leer el byte de datos anterior.

UART0 se accede a través del registro SFR control Serie (SCON0) y Buffer de datos de serie (SBUF0). EL SBUF0 sirve tanto para transmitir como recibir registros. La lectura del registro SCON0, accede al registro de recepción y la escritura en el registro SCON0, accede al registro de transmisión. UART0 puede funcionar en modo de sondeo o de interrupción.

UART0 tiene dos fuentes de interrupciones:

- Transmitir una bandera de interrupción, TI0 (SCON0.1) se establece cuando la transmisión de un byte de datos está completa.
- Recepción de una bandera de interrupción, RI0 (SCON0.0) se establece cuando la recepción de un byte de datos se ha completado.

Las banderas de interrupción de la UART0 no se borran por hardware sino que se deben borrar manualmente por software.

UART0 proporciona cuatro modos de funcionamiento, uno síncrono y tres asíncrono, seleccionados mediante el establecimiento de bits de configuración en el registro SCON0. Estos cuatro modos ofrecen diferentes velocidades de transmisión y protocolos de comunicación.

MODO	SINCRONIZACIÓN	RELOJ	BITS DATOS	BITS START/STOP
0	Síncrono	SYSCLK/12	8	None
1	Asíncrono	Timer 1, 2, 3 ó 4 desbordarse	8	1 Start, 1 Stop
2	Asíncrono	SYSCLK/32 ó SYSCLK/64	9	1 Start, 1 Stop
3	Asíncrono	Timer 1, 2, 3 ó 4 desbordarse	9	1 Start, 1 Stop

Tabla 5: Modos de funcionamientos de la UART



Para el desarrollo del presente proyecto hemos configurado la UART0 con el modo de funcionamiento 1 velocidad de transmisión variable de 8 bits.

Este modo proporciona un estándar de comunicación asíncrona, full-duplex utilizando un total de 10 bits por byte de datos:

- Un bit de inicio.
- Ocho bits de datos (LSB primero).
- Un bit de parada.

Los datos se transmiten desde el pin TX0 y se recibe en el pin RX0.

En recepción, los ocho bits de datos se almacenan en SBUF0 y el bit de parada entra en RB80 (SCON0.2).

La transmisión de datos se inicia cuando una instrucción escribe un byte de datos en el registro SBUF0.

El TI0 es la Bandera de interrupción de transmisión (SCON0.1) está fijado en el final de la transmisión.

La recepción de los datos puede comenzar en cualquier momento después de habilitar el bit de recepción REN0 (SCON0.4), para habilitarlo se ajusta a 1 lógico.

Después de la parada, fin de la transmisión, se reciben los bits, el byte de datos se cargará en el registro SBUF0 si ocurren las siguientes condiciones:

- RI0 debe ser 0 lógico.
- SM20 es 1 lógico.
- El bit de parada debe ser 1 lógico.

Si se cumplen estas condiciones, los ocho bits de datos se almacenan en SBUF0, el bit de parada se almacena en RB80 y la bandera RI0 está se establece.

Si no se cumplen estas condiciones, SBUF0 y RB80 no se cargarán y la bandera RI0 no se establecerá.

Una interrupción se producirá cuando esté activado cualquiera de las banderas de interrupción, ya se TI0 o RI0.



La velocidad de transmisión generada en el modo 1 es una función de desbordamiento del temporizador, que se muestra en las siguientes ecuaciones:

- i. Usando el Timer1.
 - Cuando $SMOD0 = 0$: $Mode1_BaudRate = 1/32 \times Timer1_OverflowRate$.
 - Cuando $SMOD0 = 1$: $Mode1_BaudRate = 1/16 \times Timer1_OverflowRate$.
 - $Timer1_OverflowRate = T1CLK (256-TH1)$.
- ii. Usando los Timers 2, 3 ó 4.
 - $Mode1_BaudRate = 1/16 \times Timer234_OverflowRate$.
 - $Timer234_OverflowRate = TnCLK (65536-RCAPn)$.

UART0 puede usar el Timer 1 operando en 8-Bit Modo Auto-Recarga o los Timers de 2, 3 ó 4 en el modo Auto-recarga para generar la velocidad de transmisión (tenga en cuenta que los relojes TX y RX se seleccionan por separado).

En el presente proyecto la UART0 utiliza el Timer 2.

En cada caso de desbordamiento del temporizador (un vuelco de todo los 0xFF para el temporizador 1, 0xFFFF para temporizadores 2, 3 y 4 a cero) se envía un reloj a la lógica de la velocidad de transmisión.

Los Temporizadores 1, 2, 3 y 4 se seleccionan como fuente de velocidad de transmisión de bits en el registro SSTA0.

El reloj de la velocidad de transmisión se selecciona mediante los bits S0TCLK1 y S0TCLK0, y el reloj de la velocidad de recepción se selecciona mediante los bits S0RCLK1 y S0RCLK0.

REGISTRO	CONFIGURACIÓN
SCON0	0x50
SSTA0	0x15

Tabla 6: Registros de selección de la velocidad de transmisión



iv. CAN

El controlador CAN se inicializa usando los siguientes pasos:

1. Configurar la página de SFR a la CAN0_PAGE (página 0x1).
2. Poner a '1' los bits INIT y CCE del registro CAN0CN.
3. Configurar los parámetros temporales en el Registro de Bits de Tiempo (Bit Timing Register) y el registro de extensión del BRP.
4. Inicializar cada mensaje objeto o poner sus bits de MsgVal a NO VALIDO.
5. Borrar el bit INIT a "0".

Temporización.

El reloj del controlador CAN (fsys) deriva del reloj del 8051 (SYSCLK).

Para mayor precisión se requiere un cristal como reloj externo.

La siguiente tabla muestra los parámetros para realizar los cálculos:

PARÁMETRO	VALOR	DESCRIPCIÓN
Reloj del sistema CIP-51 (SYSCLK).	22.1184MHz.	Oscilador externo en el "Modo de oscilador de cristal". Un cristal de cuarzo 22.1184 MHz se conecta entre XTAL1 y XTAL2.
Controlador CAN reloj del sistema (fsys).	22.1184MHz.	Derivado de SYSCLK.
CAN período de reloj (tsys).	45.211ns.	Derivado de 1/fsys.
CAN time quantum (tq).	45.211ns.	Derivado de tsys x BRP1,2
CAN longitud del bus.	10m.	Retardo de la señal 5ns/m entre nodos CAN.
Tiempo de retardo de propagación.	400ns.	2 x (retardo de bucle de transmisor-receptor + retardo de la línea de autobús).

Tabla 7: Reloj del controlador CAN



Registros CAN.

Los registros CAN se clasifican en 4 grupos:

1. Registros de protocolo del controlador CAN: Control CAN, interrupciones, control de errores, estado del bus, de prueba modos.

Se utilizan para configurar el controlador CAN, alarmas de proceso, el estado del bus monitor y colocar el controlador en modo de prueba.

Los registros de protocolo del controlador CAN son accesibles mediante CIP-51 MCU SFR para un método ordenado para facilitar su consulta y análisis y algunos se pueden acceder directamente al abordar la SFR en el mapa SFR CIP-51 para mayor comodidad.

Los registros son los siguientes:

- Registro control (CAN0CN).
- Registro de Estado (CAN0STA).
- Registro contador de errores.
- Bit Timing Register: Temporización de bit.
- Baud Rate Prescaler Extension Register: extensión del preescalado de reloj.

A los registros CAN0STA y CAN0CN se puede acceder a través de CIP-51 MCU SFR. Todos los demás se acceden indirectamente utilizando la dirección CAN con un método ordenado a través CAN0ADR, CAN0DATH y CAN0DATL.

2. Registro de la interface de los mensajes de objetos: Sirven para configurar 32 mensajes de objetos, para enviar y recibir datos hacia y desde los mensajes de objeto.

En los conjuntos de registros IF1 e IF2 (registros de petición de comando CAN) se indica el número de mensaje a transmitir a la cola de mensajes CAN.



Se acceden vía indirecta a través de CAN0ADR, CAN0DATH y CAN0DATL. Se dividen en 2 grupos y sirven para acceder a los mensajes objeto.

Con estos mensajes se pueden enviar datos, recibir datos, y configurar al controlador para que filtre mensajes.

CAN0ADR se puede incrementar automáticamente (de 0x08 a 0x12 para el grupo 1 de registros de interface y de 0x20 a 0x2A para el segundo grupo). Se autoincrementa al hacer una lectura o escritura del registro CAN0DATL => Operar primero con el H.

3. Registros de manejo de Mensajes: Son de lectura únicamente y dan información al 8051 sobre los mensajes de objetos (flags de válido, transmisión pendiente, flags de datos nuevos) y de interrupciones (estado de interrupción o qué mensaje ha provocado la interrupción).

Se acceden usando el direccionamiento indirecto igual que el grupo anterior.

4. Registros Especiales del 8051 (SFRs): Dispone de 6 registros mapeados en memoria RAM directa en la página 1. Estos registros permiten controlar el protocolo CAN y dar acceso al mapeo indirecto del resto de registros del controlador CAN. Los registros son: CAN0CN, CAN0STA, CAN0ADR, CAN0DATH y CAN0DATH.



Los mensajes CAN.

La trama de los mensajes CAN es la siguiente:

Campo	Bits	Propósito
Start-of-frame	1	Indica el inicio de una trama
Identifier	11	Identificador único de los datos
Remote transmission request (RTR)	1	Debe ser un nivel <i>dominante</i> si se emplea
Identifier extension bit (IDE)	1	Debe ser un nivel <i>dominante</i> si se emplea
Reserved bit (r0)	1	Debería ser <i>dominante</i> , pero se aceptan ambos valores.
Data length code (DLC)	4	Número de bytes que transporta (0-8 bytes)
Data field	0-64	0-8 bytes del mensaje
CRC	15	CRC para la comprobación
CRC delimiter	1	Debe ser obligatoriamente Recesivo
ACK slot	1	El transmisor emite un nivel <i>recesivo</i> y cualquier nodo puede responder con un bit <i>dominante</i>
ACK delimiter	1	Debe ser obligatoriamente Recesivo
End-of-frame (EOF)	7	Debe ser obligatoriamente Recesivo

Figura 29: Trama de los mensajes CAN 1

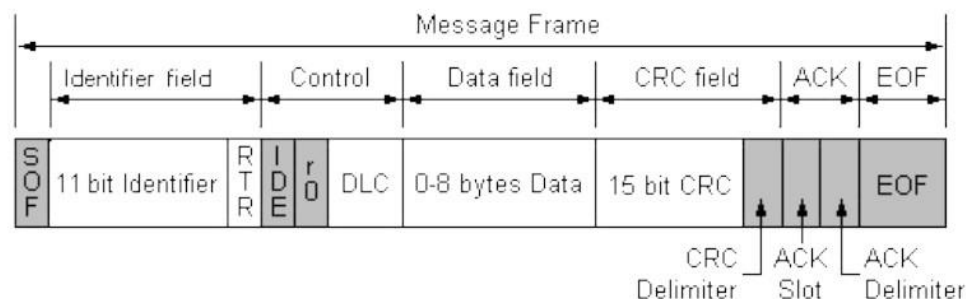


Figura 30: Trama de los mensajes CAN 2

Las características de los mensajes son:

- **Enrutado de Información:** En los sistemas CAN los nodos no necesitan hacer uso de ninguna información acerca de la configuración del sistema (por ejemplo, la dirección de la estación).
- **Flexibilidad del sistema:** Los nodos pueden añadirse a la red CAN sin ningún tipo de cambio en el software o hardware de los nodos o en la capa de aplicación del sistema.



- **Enrutado de Mensajes:** El contenido de un mensaje se nombra por su ID (identificador). El identificador no indica el destino de un mensaje, pero describe el significado de los datos, por lo que todos los nodos de la red son capaces de decidir a través del filtrado de mensajes cuándo se requiere una actuación sobre los datos o cuándo no.
- **Multicast:** Como consecuencia del concepto de filtrado de mensajes, ningún nodo puede recibir y simultáneamente actuar sobre el mismo mensaje.
- **Consistencia de Datos:** En una red CAN se garantiza que un mensaje será aceptado simultáneamente por todos los nodos o por ninguno. Esta consistencia de datos se alcanza gracias al multicast y el manejo de los errores.
- **Bit Rate:** La velocidad de un Sistema CAN puede variar de una implementación a otra. De todas maneras, en un sistema CAN dado, la velocidad es fijada y uniforme en toda la RED.
- **Prioridad:** El ID de un mensaje define una prioridad estática durante el acceso al bus.
- **Petición de Datos Remota:** A través de una trama remota un nodo puede requerir que otro nodo responda con otro mensaje con el ID correspondiente. Tanto la Trama de Datos como la Trama Remota se nombran por medio del mismo ID.
- **Multimaestro:** Cuando el bus está disponible, cualquier nodo puede empezar la transmisión de un mensaje. La unidad con el mensaje de mayor prioridad será la que gane el acceso al bus. Este sistema de acceso al bus tiene entre otras características que no consume tiempo (lo que se traduce en un mayor ancho de banda) ya que todos los nodos transmiten su identificador a la vez que sondean el bus. Si alguno encuentra un bit dominante cuando el bit correspondiente del ID del mensaje que quiere transmitir es recesivo, automáticamente deja el proceso de arbitraje y esperará



a la siguiente etapa de arbitraje del bus para intentar transmitir su mensaje, mientras que recolecta los datos del bus.

v. Puertos de Entrada/Salida

La familia de dispositivos C8051F04x tienen una amplia gama de recursos digitales que están disponibles a través de los cuatro Puertos E/S más bajos: P0, P1, P2 y P3. Cada uno de los pines en P0, P1, P2 y P3, se puede definir como un Propósito General I / O (GPIO) pin o puede ser controlado por un periférico digital o función (como UART0 o INT1 por ejemplo). Los controles de diseño del sistema asignan funciones digitales a los pins, limitado solamente por el número de pines disponibles. Esta flexibilidad de asignación de recursos se logra mediante el uso de una “Crossbar” según prioridades. El estado de un pin de puerto I / O siempre se puede leer en su registro de datos asociado con independencia de que ese pin se ha asignado a un periférico digital o se comporta como GPIO. Los pines en los puertos 1, 2 y 3 se pueden utilizar como entradas analógicas a ADC2 (C8051F040/1/2/3 solamente), voltaje comparadores analógicos, y ADC0, respectivamente.

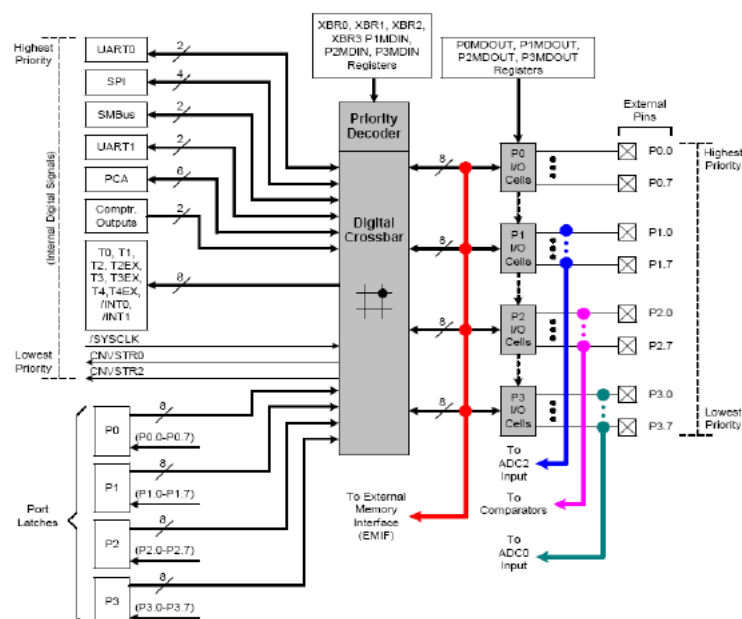


Figura 31: Puertos de entrada / salida 1





Debido a que los registros del Crossbar afectan el PINOUT de los periféricos del dispositivo, se configuran normalmente en el código de inicialización del sistema antes de que los periféricos se configuren ellos mismos. Una vez configurado, los registros del Crossbar normalmente se dejan solos.

Los controladores de salida de los puertos del 0 al 3 permanecen deshabilitados hasta que el Crossbar se habilita estableciendo XBARE. El modo de salida de cada pin del puerto se puede configurar para ser open drain o push-pull. En la configuración Push-Pull, escribir un 0 lógico en el bit asociado en el registro Puerto de datos hará que el pin de puerto sea conducido a GND, y escribir un 1 lógico hará que el pin de puerto sea conducido a VDD. En la configuración de open drain, escribir un 0 lógico en el bit asociado en el registro Puerto de datos hará que el pin de puerto sea conducido a GND, y un 1 lógico hará que el pin de puerto asuma un estado de alta impedancia. La configuración de open drain es útil para evitar la contención entre dispositivos en sistemas en los que el pasador de Puerto participa en una interconexión compartida en la que múltiples salidas están conectadas al mismo cable físico (como la señal de SDA en una conexión de SMBus).

Los modos de salida de los pines del puerto 0 al 3 se determinan mediante los bits en los registros PnMDOUT.

REGISTRO	CONFIGURACIÓN
XBR0	0x04
XBR1	0x00
XBR2	0x40
XBR3	0x80
P0MDOUT	0x01
P1MDOUT	0x40

Tabla 8: Configuración de los pines del puerto 0 al 3



vi. Interrupciones

Las características de las interrupciones son las siguientes:

- Soporta hasta 20 fuentes de interrupción diferentes (más el Reset) con 2 niveles de prioridades.
- Cada interrupción tiene asociado uno o más flags en algún registros especial. Cuando se produce una interrupción el flag se pone a 1.
- Cuando se activa una interrupción se produce un LCALL a la rutina de servicio de interrupción. El código de esta rutina acabará con RETI.
- Hay un bit de enable global de interrupciones (EA) y enables para cada interrupción.
- El flag de interrupción debe ser puesto a cero en la rutina de interrupción.
- El software puede simular cualquier interrupción poniendo a 1 el flag correspondiente.
- Hay 2 interrupciones externas /INT0 e /INT1, mapeadas a algún puerto digital, y que pueden configurarse como activas por nivel, flanco, nivel alto o bajo.
- El tiempo de respuesta ante una interrupción depende del estado de la CPU en el momento de la interrupción. Lo más rápido son 5 ciclos de reloj. Y lo más lento son 18 ciclos si estaba ejecutando un RETI y posteriormente una DIV. Cuando se escribe o lee la Flash la CPU se para (stall) y no ve las interrupciones.

Las fuentes de interrupción son:

- Reset.
- /INT0.
- Timer0 Overflow.
- /INT1.



- Timer1 Overflow.
- UART0.
- Timer2 Overflow.
- SPI0.
- SMB0.
- ADC0 rango de comparación.
- PCA.
- Comparador 0.
- Comparador 1.
- Comparador 2.
- Timer3 Overflow.
- ADC0 fin de conversión.
- Timer4 Overflow.
- ADC2 rango de comparación.
- ADC2 fin de conversión.
- CAN.
- UART1.



Interrupt Source	Interrupt Vector	Priority Order	Pending Flag	Bit addressable?	Cleared by HW?	SFRPAGE (SFRPGEN = 1)	Enable Flag	Priority Control
Reset	0x0000	Top	None	N/A	N/A	0	Always Enabled	Always Highest
External Interrupt 0 (/INT0)	0x0003	0	IE0 (TCON.1)	Y	Y	0	EX0 (IE.0)	PX0 (IP.0)
Timer 0 Overflow	0x000B	1	TF0 (TCON.5)	Y	Y	0	ET0 (IE.1)	PT0 (IP.1)
External Interrupt 1 (/INT1)	0x0013	2	IE1 (TCON.3)	Y	Y	0	EX1 (IE.2)	PX1 (IP.2)
Timer 1 Overflow	0x001B	3	TF1 (TCON.7)	Y	Y	0	ET1 (IE.3)	PT1 (IP.3)
UART0	0x0023	4	RI0 (SCON0.0) TI0 (SCON0.1)	Y		0	ES0 (IE.4)	PS0 (IP.4)
Timer 2	0x002B	5	TF2 (TMR2CN.7)	Y		0	ET2 (IE.5)	PT2 (IP.5)
Serial Peripheral Interface	0x0033	6	SPIF (SPI0CN.7) WCOL (SPI0CN.6) MODF (SPI0CN.5) RXOVN (SPI0CN.4)	Y		0	ESPI0 (EIE1.0)	PSPI0 (EIP1.0)
SMBus Interface	0x003B	7	SI (SMB0CN.3)	Y		0	ESMB0 (EIE1.1)	PSMB0 (EIP1.1)
ADC0 Window Comparator	0x0043	8	AD0WINT (ADC0CN.2)	Y		0	EWADC0 (EIE1.2)	PWADC0 (EIP1.2)
Programmable Counter Array	0x004B	9	CF (PCA0CN.7) CCFn (PCA0CN.n)	Y		0	EPCA0 (EIE1.3)	PPCA0 (EIP1.3)
Comparator 0	0x0053	10	CP0FIF/CP0RIF (CPT0CN.4/5)			1	CP0IE (EIE1.4)	PCP0 (EIP1.4)
Comparator 1	0x005B	11	CP1FIF/CP1RIF (CPT1CN.4/5)			2	CP1IE (EIE1.5)	PCP1 (EIP1.5)
Comparator 2	0x0063	12	CP2FIF/CP2RIF (CPT2CN.4/5)			3	CP2IE (EIE1.6)	PCP2 (EIP1.6)
Timer 3	0x0073	14	TF3 (TMR3CN.7)			1	ET3 (EIE2.0)	PT3 (EIP2.0)
ADC0 End of Conversion	0x007B	15	ADC0INT (ADC0CN.5)	Y		0	EADC0 (EIE2.1)	PADC0 (EIP2.1)
Timer 4	0x0083	16	TF4 (TMR4CN.7)			2	ET4 (EIE2.2)	PT4 (EIP2.2)
ADC2 Window Comparator	0x0093	17	AD2WINT (ADC2CN.0)			2	EWADC2 (EIE2.3)	PWADC2 (EIP2.3)
ADC2 End of Conversion	0x008B	18	ADC2INT (ADC1CN.5)			2	EADC1 (EIE2.4)	PADC1 (EIP2.4)
CAN Interrupt	0x009B	19	CAN0CN.7		Y	1	ECAN0 (EIE2.5)	PCAN0 (EIP2.5)
UART1	0x00A3	20	RI1 (SCON1.0) TI1 (SCON1.1)			1	ES1 (EIE2.6)	PS1 (EIP2.6)

Figura 33: Interrupciones



II. APLICACIÓN DE CONTROL PARA NXT

a. Protocolo de comunicación del NXT.

La longitud máxima que permite el NXT en los paquetes enviados por Bluetooth es de 64 bytes. Todos los mensajes deben incluir dos bytes al inicio para indicar la longitud de la trama. Estos dos bytes no se tienen en cuenta para indicar la longitud del paquete. En la figura 34 se muestra la estructura de una trama de datos del NXT.

Length, LSB	Length, MSB	Command Type	Command	Bytes 4 to N-1: Message	Byte N: End
----------------	----------------	-----------------	---------	----------------------------	----------------

Figura 34: Trama de datos para enviar comandos Bluetooth al NXT.

En la siguiente figura se detalla la función de cada byte en la trama:

Byte 0	LSB (Less Significant Byte). Dígito menos significativo para indicar la longitud del paquete.
Byte 1	MSB (Most Significant Byte). Dígito más significativo para indicar la longitud del paquete.
Byte 2	Tipo de comando 0x00: Comando directo, se requiere respuesta 0x01: Comando de sistema, se requiere respuesta 0x02: Comando de respuesta 0x80: Comando directo, no se espera respuesta 0x81: Comando de sistema, no se espera respuesta
Byte 3	Comando. Byte que indica que se ha de hacer con los datos: Open, Read, Write, Delete data, y Direct communication con el NXT.
Byte 4 a N-1	Estos bytes aportan información adicional.
Byte N	0x00 que indica el final de paquete.

Figura 35: Especificación de la función de cada byte de la trama y los valores que pueden tomar según lo que deban indicar.



Especialmente interesante resulta la posibilidad de dar órdenes directas al robot mediante comandos directos sin necesidad de que éste ejecute ningún programa en el robot. De esta forma es posible controlar remotamente el movimiento del robot. Un ejemplo simple de comando directo es el que nos devuelve como respuesta el nivel de batería del NXT.

Byte 0	0x00 o 0x80
Byte 1	0x0B

Figura 36: Trama de bytes correspondiente a un comando directo para solicitar el nivel de batería del NXT.

Byte 0	0x02
Byte 1	0x0B
Byte 2	Status
Bytes 3-4	Voltaje en mV

Figura 37: Trama de bytes correspondiente a la respuesta del NXT indicando su voltaje.

En el Anexo 3 se incluye una descripción más detallada del protocolo de comunicación y todos los comandos directos soportados por el NXT.

b. Diseño e implementación de Interfaz de control con Visual Studio.

Cuando creamos una aplicación de Windows Forms estamos creando una aplicación de escritorio para Windows basada en formulario (formada por ventanas).

Este Framework de .NET nos ofrece herramienta para diseñar fácilmente las interfaces de usuario y añadirle funcionalidad usando el lenguaje de programación Visual Basic.

Este proceso de creación de aplicaciones Windows Forms se basa en estos pasos:



i. Inserción y configuración de controles.

El primer paso consistirá en ir añadiendo controles a nuestro interfaz. Dispondremos de una amplia variedad.

Los controles que nosotros usaremos en nuestro proyecto serán los siguientes:

- **Label:** se utilizan para mostrar texto o imágenes que el usuario no puede editar. Se utilizan para identificar objetos en un formulario.
- **Button:** permite al usuario hacer clic en el para ejecutar una acción. Puede mostrar tanto texto como imágenes. Cuando se hace clic en el botón, da la sensación de que se ha presionado y soltado.
- **ComboBox:** Se utiliza para mostrar datos en un cuadro combinado desplegable. De forma predeterminada, aparece en dos partes: la parte superior es un cuadro de texto que permite al usuario escribir un elemento de la lista. La segunda parte es un cuadro de la lista que muestra una lista de elementos, de los cuales el usuario puede seleccionar uno.
- **TextBox:** se utiliza para obtener entradas del usuario o para mostrar texto. Se utiliza generalmente para el texto que se puede editar, aunque también puede configurarse como control de solo lectura. Los cuadros de texto pueden mostrar varias líneas, ajustar el texto al tamaño del control y agregar formato básico. Este control permite un único formato para el texto que se muestra o escribe en el control.
- **NumericUpDown:** Muestra un único valor numérico que el usuario puede aumentar o reducir haciendo clic en los botones de arriba y abajo del control.
- **GroupBox:** muestra un marco alrededor de un grupo de controles con un título opcional.



- **Timer:** componente que genera un evento a intervalos definidos por el usuario.
- **SerialPort:** representa un recurso de puerto serie.

La interfaz gráfica creada proporciona un entorno visual sencillo, tal y como se puede apreciar en la figura. A través de ella podremos establecer la comunicación a través del puerto serie y controlar el robot desde nuestro ordenador.

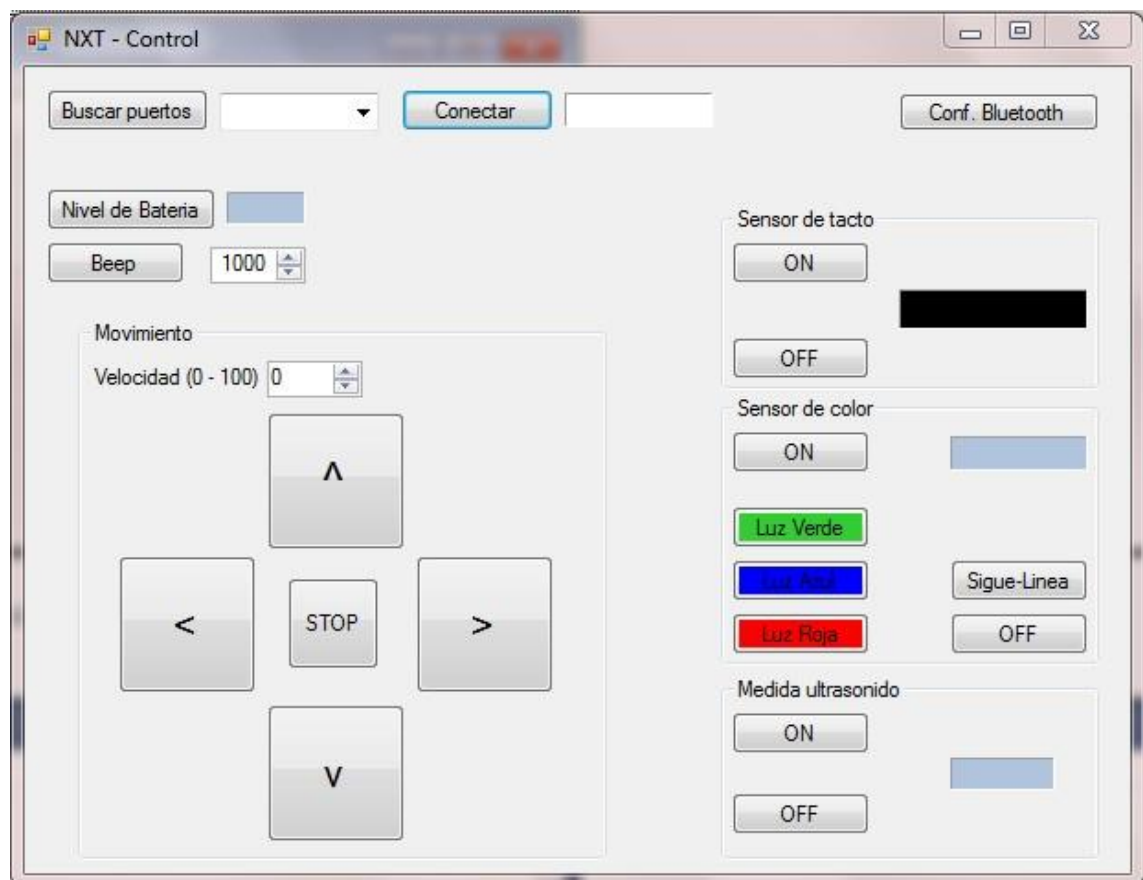


Figura 38: Interfaz de control para NXT



ii. Asignación de métodos y procedimientos a eventos.

Ahora tenemos que asignar métodos y procedimientos a determinados eventos de nuestros controles, como pueden ser el hacer clic sobre un botón o mantenerlo pulsado. Registraremos y daremos nombre a esos eventos para después añadirles funcionalidad mediante código VB. Introducimos ahora el concepto de página code-behind (código trasero). Cada formulario de Windows Forms tendrá dos vistas fundamentales, una vista del interfaz gráfico que hemos visto en el apartado anterior, y que se representa y edita gráficamente, y una vista el código de dicho formulario (Code-behind).

El code-behind controlará el comportamiento de la vista del interfaz mediante código, por ejemplo, inicializado todos los controles del interfaz, cambiando de valor algún campo o redireccionando directamente a otro formulario.

Las funciones de nuestros controles serán:

- **Establecer la conexión a través del puerto serie con el microcontrolador.**

Para ello pulsamos sobre “Buscar puertos” y en el desplegable de la izquierda nos aparecen todos los puertos activados en nuestro ordenador, seleccionamos el puerto donde hemos conectado el microcontrolador y pincharemos en “Conectar”.

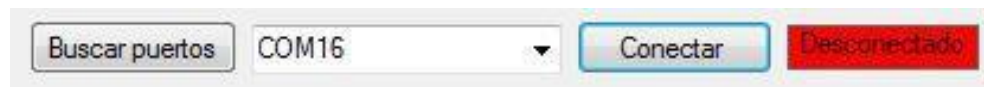


Si marcamos la opción de “Conectar” y se realiza la conexión con el puerto aparecerá en el display indicado que se ha conectado con dicho puerto, y el botón que antes tenía la función de conectar ahora pasa a tener la función de “Desconectar”.





Pulsando en “Desconectar” cerramos el puerto elegido y podemos elegir otro.



El código que se ejecutara será:

```
Private Sub BuscarPuertos_Click(sender As Object, e As EventArgs) Handles
    BuscarPuertos.Click 'Función para buscar puertos serie activados.
        ComboPuertos.Items.Clear()

        For Each PuertoDisponible As String In My.Computer.Ports.SerialPortNames
            'Busca cada uno de los puertos serie disponibles en el ordenador
            ComboPuertos.Items.Add(PuertoDisponible)
        Next
        If ComboPuertos.Items.Count > 0 Then 'Si hay puertos serie utilizándose los
            muestra en el combobox
            ComboPuertos.Text = ComboPuertos.Items(0)
        Else
            MessageBox.Show("NINGUN PUERTO ENCONTRADO") ' si no hay
            ninguno disponible te indica que no se ha encontrado ningún puerto
            ComboPuertos.Items.Clear()
        End If
    End Sub

Private Sub Conexion_Click (ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Conexion.Click 'Función para establecer la
    conexión con un puerto serie.

    Dim byteOut(15) As Byte 'Variable que almacena la trama para enviar a
    través del puerto.

    Try
        If Conexion.Text = "Conectar" Then 'Si pulsamos conectar.

            With SerialPort 'Definir los parámetros de la conexión del puerto serie.

                .PortName = ComboPuertos.Text
                .BaudRate = 9600
                .Parity = IO.Ports.Parity.None
                .DataBits = 8
                .StopBits = IO.Ports.StopBits.One
                .ReadTimeout = 300
                .WriteTimeout = 300

            End With

            SerialPort.Open() 'Abre el puerto serie elegido.
            EstadoConexion.Text = "Conectado" 'Emitimos el mensaje de conexión.
            EstadoConexion.BackColor = Drawing.Color.LawnGreen
            Velocidad.Text = "25" 'Se establece una velocidad por defecto al iniciar
            la conexión.
            Conexion.Text = "Desconectar" 'El pulsador para conectar ahora sirve
            para desconectar.
            TimerBateria.Enabled = True 'Se activa el timer para la pedida del
            nivel de batería.
            Call Bateria_Click(sender, e) 'Llamamos a la función de la batería
            para pedir nada más iniciar el nivel de batería del NXT.
```



```

Elseif Conexion.Text = "Desconectar" Then 'Si pulsamos desconectar.

    SerialPort.Close() 'Cerramos el puerto.
    EstadoConexion.Text = "Desconectado" 'Emitimos el mensaje de
                                         Desconexión.

    EstadoConexion.BackColor = Drawing.Color.Red
    Conexion.Text = "Conectar" 'El pulsador vuelve a tener la función
                                de conectar.

    TimerBateria.Enabled = False 'Desactivamos el timer de la batería
                                para que no se envíe ninguna trama mientras el puerto está cerrado.
    MarcadorBateria.BackColor = Drawing.Color.LightSteelBlue
                                'Reiniciamos el marcador de la batería.

    MarcadorBateria.Text = ""
    Velocidad.Text = "0" 'Eliminamos la velocidad por defecto.
End If

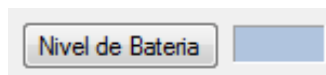
Catch ex As Exception
    MsgBox("NO SE PUDO CONECTAR CON EL PUERTO " &
CStr(SerialPort.PortName)) 'Si no se puede conectar con el puerto elegido nos
                                llega un mensaje de error.
End Try

End Sub

```

- **Lectura del nivel de batería de nuestro NXT:**

Al establecerse la comunicación con el puerto elegido, automáticamente se activa un Timer, el cual cada 10 segundos manda una trama preguntando por el nivel de batería del NXT y mostrando el valor a través de un display:



También disponemos de un pulsador el cual pregunta por el nivel de batería independientemente del Timer.

El código que se ejecutará será el siguiente:

```

Private Sub Bateria_Click(sender As Object, e As EventArgs) Handles Bateria.Click
'Función para enviar la trama para pedir el nivel de batería del NXT y devolver el valor.

    Dim byteOut(3) As Byte 'Variable para almacenar la trama a enviar.
    Dim byteIn(6) As Byte 'Variable que almacena la trama recibida.
    Dim Voltage, t As Integer
    t = 0

    Try
        byteOut(0) = &H2 '2 bytes en mensaje de salida.
        byteOut(1) = &H0 '0 para NXT
        byteOut(2) = &H0 '&H0 = esperamos respuesta; &H80 = no esperamos
                            respuesta.
        byteOut(3) = &HB '$HB = Comando de lectura de batería.
        SerialPort.Write(byteOut, 0, 4) 'Trama enviada por el puerto serie.
    End Try
End Sub

```




While SerialPort.BytesToRead = 0 'Bucle que espera a que se reciban datos para no mostrar inconcluencias en el display.

End While

byteIn(0) = SerialPort.ReadByte 'Numero de bytes en el mensaje recibido.
byteIn(1) = SerialPort.ReadByte 'Debe ser cero para el NXT.
For t = 2 **To** 1 + byteIn(0) 'Leemos el resto del mensaje y lo almacenamos en nuestra variable de recepción.

byteIn(t) = SerialPort.ReadByte()

Next

Voltage = byteIn(5)+byteIn(6)*256 'El voltaje tiene el byte bajo en el 5 y el byte alto en el 6.

MarcadorBateria.Text = Voltage / 1000 'Muestra el voltaje.

If MarcadorBateria.Text > 5 **Then** 'Cambiamos el color del display en función del voltaje.

MarcadorBateria.BackColor = Drawing.Color.LawnGreen

Elseif MarcadorBateria.Text < 5 **And** MarcadorBateria.Text > 2 **Then**

MarcadorBateria.BackColor = Drawing.Color.Orange

Elseif MarcadorBateria.Text < 2 **Then**

MarcadorBateria.BackColor = Drawing.Color.Red

End If

Catch ex As Exception

TimerBaterly.Enabled = **False** 'Se desactiva el timer en caso de error para que no siga enviando la trama.

MsgBox("ERROR EN LA LECTURA DEL NIVEL DE BATERIA") 'Muestra en pantalla un mensaje de Error.

End Try

End Sub

Private Sub TimerBaterly_Tick(sender As Object, e As EventArgs) **Handles**

TimerBaterly.Tick

Call Bateria_Click(sender, e) 'Este Timer está establecido para que llame a la función de lectura de batería cada 10s.

End Sub

• Alarma de búsqueda:

Se ha establecido un pulsador con un marcador numérico asociado, al pinchar el pulsador se envía una trama que le indica al NXT que debe emitir un tono con la frecuencia indicada en nuestro marcador (se puede utilizar en caso de pérdida de vista del NXT):



El código que se ejecutará será el siguiente:

Private Sub Alarma_Click(sender As Object, e As EventArgs) **Handles** Alarma.Click

Dim byteOut(7) As Byte 'Variable que almacena la trama a enviar.

Dim freq As Int16 'Variable de Frecuencia.

Dim i As Integer = 0 'Contador

Dim f As Integer = 0 'Contador



```
Try
    byteOut(0) = &H6
    byteOut(1) = &H0
    byteOut(2) = &H80
    byteOut(3) = &H3
    freq = Frecuencia.Value

    Do Until i = 100 Or f = 2
        byteOut(4) = freq - (freq / 256) * 256
        byteOut(5) = freq / 256
        byteOut(6) = &H64
        byteOut(7) = &H0
        SerialPort.Write(byteOut, 0, 8)
        i = i + 1
    If i = 100 Then
        i = 0
        Do Until i = 100
            byteOut(4) = (freq / 2) - ((freq / 2) / 256) * 256
            byteOut(5) = (freq / 2) / 256
            byteOut(6) = &HC8
            SerialPort.Write(byteOut, 0, 8)
            i = i + 1
        Loop
        i = 0
        f = f + 1
    End If
Loop

Catch ex As Exception
    MsgBox(ex.ToString)
End Try

End Sub
```

'6 bytes en el byte enviado
'siempre 0 para el NXT
'&H0 = esperamos respuesta; &H80 = no esperamos respuesta.
'PLAYTONE
'Byte bajo de frecuencia
'Byte alto de frecuencia
'Duración de byte bajo = 100 ms
'Duración de byte bajo

- **Controlar los motores:**

Lo haremos a través de los 4 pulsadores que tenemos para dirigir el robot en las cuatro direcciones y el pulsador de parada. Al pulsar en cada uno de ellos mandamos una trama diferente a través del CAN para que actúe sobre ellos. Hay que tener en cuenta que en cuanto se produce la conexión con el puerto serie se establece una velocidad por defecto de 25 (25% de la velocidad total de los





motores) pudiéndose cambiar esta en cualquier momento desde el cuadro numérico de “Velocidad”.

Por ejemplo manteniendo pulsado el botón “Λ” se ejecutara el siguiente código que envía la trama necesaria para activar los dos motores en el mismo sentido, al dejar de pulsar el botón se manda automáticamente el código de parada.

Public Sub Adelante_MouseDown(sender **As Object**, e **As EventArgs**) **Handles** Adelante.MouseDown

Dim byteOut(13) **As Byte** ' Variable para enviar el comando por el puerto serie.

Try

byteOut(0) = &HC 'LSB, Cantidad de bytes a enviar

byteOut(1) = &H0 'MSB

byteOut(2) = &H80 ' &H0 = Si queremos respuesta, &H80 = Si no queremos respuesta.

byteOut(3) = &H4 'tipo de comando que indica que se van a controlar los motores.

byteOut(4) = &H1 'Puerto de motores con el que se va a interactuar. En este caso Puerto B.

byteOut(5) = **CByte**(Velocidad.Text) ' Velocidad de salida en un rango de 0 a 100.

byteOut(6) = &H5 ' Modo Motor ON con regulación y sincronización entre los dos motores.

byteOut(7) = &H1 ' Ajusta la velocidad en función de la carga.

byteOut(8) = &H0 ' Sincronización del radio de giro, en este caso no es necesario.

byteOut(9) = &H20 ' Activa la alimentación del motor en el puerto especificado.

byteOut(10) = &H0 ' Los siguientes 4 bytes son a 0.

byteOut(11) = &H0

byteOut(12) = &H0

byteOut(13) = &H0

SerialPort1.Write(byteOut, 0, 14) ' Envío los bytes del comando a través del puerto serie.

byteOut(4) = &H2 ' Puerto de motores a interactuar, puerto C.

SerialPort1.Write(byteOut, 0, 14) ' Como el resto de bytes se mantienen iguales vuelvo a enviar el comando.

Catch ex As Exception

MsgBox("EL MOTOR NO RESPONDE")

End Try

End Sub

Private Sub Adelante_MouseUp(sender **As Object**, e **As MouseEventArgs**) **Handles** Adelante.MouseUp

Call Parada_Click(sender, e) ' Al dejar de mantener pulsado llama a la subrutina de parada.

End Sub



El mismo código se utilizaría para el botón “v”, solo cambiaría el byte de la velocidad por el siguiente:

```
byteOut(5) = CByte(256 - Velocidad.Text)
```

A 256 le restamos la velocidad establecida para que la velocidad en este motor sea negativa.

Igual para los comandos de “<” y “>”, lo que cambiaría sería:

- Para ambos motores:

```
byteOut(6) = &H7
```

Se sincronizan ambos motores y no les afecta la carga externa en comparación de los comandos “Adelante” y “Atrás”.

- Para un motor:

```
byteOut(5) = CByte((Velocidad.Text) / 2)
```

Dividimos la velocidad por la mitad para que los giros vayan a una velocidad menor que la establecida para ir hacia adelante o hacia atrás.

- Para el otro motor:

```
byteOut(5) = CByte(256 - (Velocidad.Text) / 2)
```

El segundo motor además de ir a la mitad de la velocidad tiene también la velocidad negativa para poder realizar el giro sobre si mismo.

Y por último tenemos el botón de “STOP”, el cual manda el siguiente código que frena los motores deteniendo su alimentación.

```
Public Sub Parada_Click(sender As Object, e As EventArgs) Handles Parada.Click
```

```
Dim byteOut(13) As Byte
```

```
Try
```

```
byteOut(0) = &HC ' LSB, Cantidad de bytes a enviar
byteOut(1) = &H0 ' MSB
byteOut(2) = &H80 ' &H0 = Si queremos respuesta, &H80 = Si no queremos
                    respuesta.
byteOut(3) = &H4 ' Tipo de comando que indica que se van a
                    controlar los motores.
byteOut(4) = &H1 ' Puerto de motores con el que se va a interactuar. En este
                    caso Puerto B.
byteOut(5) = &H0 ' Velocidad de salida en un rango de 0 a 100.
```



```
byteOut(6) = &H0 ' Modo freno.
byteOut(7) = &H1 ' Ajusta la velocidad en función de la carga.
byteOut(8) = &H0 ' Sincronización del radio de giro, en este caso no es
                  necesario.
byteOut(9) = &H0 ' Desactiva la alimentación del motor en el puerto especificado.
byteOut(10) = &H0 ' Los siguientes 4 bytes son a 0.
byteOut(11) = &H0
byteOut(12) = &H0
byteOut(13) = &H0
SerialPort1.Write(byteOut, 0, 14) ' Envío los bytes del comando a través del
                                   puerto serie.

byteOut(4) = &H2 ' Puerto de motores a interactuar, puerto C.
SerialPort1.Write(byteOut, 0, 14) ' Envío los bytes del comando a través del
                                   puerto serie.

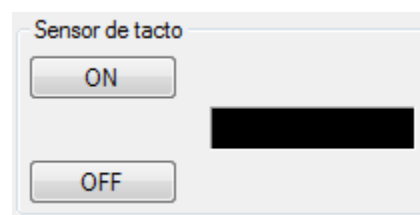
Catch ex As Exception
    MsgBox("ERROR STOP")
End Try

End Sub
```

- **Utilidad para el sensor de contacto:**

Este sensor es un sensor muy simple ya que se trata de un Switch que cuando se pulsa manda un 1 y si no manda un 0, este sensor puede utilizarse como final de carrera o programarse como pulsador para alguna aplicación en especial. Para ver un poco su funcionamiento se ha programado para que cuando se pulse se cambie de color una barra que aparece en la aplicación y se encienda un LED del sensor de color (que veremos más adelante).

El diseño del control de este sensor es el siguiente:



Al pulsar sobre "ON" se activa un Timer que cada 300ms realiza la comprobación para ver si el sensor se ha pulsado o no, si se ha pulsado, la barra que aparece en negro cambia a blanco y se enciende el LED Verde del sensor de color, cuando se deja de pulsar se apaga el LED y la barra vuelve a ponerse en negro. Al pulsar sobre "OFF" desactivamos el Timer y por consiguiente deja de hacerse la comprobación. El código utilizado es el siguiente:



```

Private Sub TimerTacto_Tick(sender As Object, e As EventArgs) Handles
TimerTacto.Tick
Dim byteOut(6) As Byte
Dim byteIn(17) As Byte
Dim g As Integer = 0

Try
    'SET INPUT MODE
    byteOut(0) = &H5 'Cantidad de bytes que va a enviar
    byteOut(1) = &H0 'Siempre 0
    byteOut(2) = &H80 '&H0 = se espera respuesta &H80 = no se espera respuesta
    byteOut(3) = &H5 'Comando a enviar (SETINPUTMODE)
    byteOut(4) = &H1 'Puerto
    byteOut(5) = &H1 'tipo de sensor (SWITCH)
    byteOut(6) = &H20 'Modo del sensor = BOOLEAN MODE
    SerialPort.Write(byteOut, 0, 7)

    'GET INPUT VALUES
    byteOut(0) = &H3 'Cantidad de bytes que va a enviar
    byteOut(1) = &H0 'Siempre 0
    byteOut(2) = &H0 '&H0 = se espera respuesta &H80 = no se espera respuesta
    byteOut(3) = &H7 'Comando a enviar (GETINPUTVALUES)
    byteOut(4) = &H1 'Puerto
    SerialPort.Write(byteOut, 0, 5)

    byteIn(0) = SerialPort.ReadByte 'Numero de bytes en la respuesta
    byteIn(1) = SerialPort.ReadByte 'Debe ser 0 para el NXT.
    For g = 2 To 1 + byteIn(0) 'Leemos el resto del mensaje.
        byteIn(g) = SerialPort.ReadByte()
    Next

    If byteIn(14) = 1 Then 'Si se pulsa
        Señal.BackColor = Drawing.Color.White 'La barra se vuelve blanca
        Call LVerde_Click(sender, 'Se enciende el LED Verde
        Else 'Si no se pulsa
            Señal.BackColor = Drawing.Color.Black 'La barra se vuelve negra
            Call Apagarluz_Click(sender, e) 'Se apaga el LED
        End If

    Catch ex As Exception
        TimerTacto.Enabled = False 'En caso de error desactivamos el Timer
        MsgBox("ERROR EN LA LECTURA") 'En caso de error mostramos este mensaje
    End Try
End Sub

Private Sub SensorTacto_Click(sender As Object, e As EventArgs) Handles
SensorTacto.Click
TimerTacto.Enabled = True 'Activamos el Timer
End Sub

Private Sub ApagarTacto_Click(sender As Object, e As EventArgs) Handles
ApagarTacto.Click
TimerTacto.Enabled = False 'Desactivamos el Timer
End Sub

```

- **Utilización del sensor de color:**

El sensor de color es un sensor que tiene varios usos, entre ellos distinguir entre 6 colores y emitir 3 colores distintos.



El control que hemos diseñado para este sensor es el siguiente:



- Al pinchar sobre el pulsador “ON” el sensor entra automáticamente En modo lectura de colores y va mostrando los diferentes colores en el display, en esta función se activa un timer que realiza la lectura de color cada 300ms. El código utilizado es el siguiente:

```
Private Sub SensorLuz_Click(sender As Object, e As EventArgs) Handles  
SensorLuz.Click
```

```
Dim byteOut(6) As Byte  
Dim byteIn(17) As Byte  
Dim g As Integer = 0  
TimerColor.Enabled = True 'Activamos el timer que llama a la función de  
                             lectura de color cada 300ms.  
  
Try  
  
    'SET INPUT MODE  
    byteOut(0) = &H5 ' Cantidad de bytes que va a enviar  
    byteOut(1) = &H0 ' Siempre 0  
    byteOut(2) = &H80 ' &H0 = se espera respuesta &H80 = no se espera  
                      respuesta  
    byteOut(3) = &H5 ' Comando a enviar (SETINPUTMODE)  
    byteOut(4) = &H2 ' Puerto  
    byteOut(5) = &HD ' tipo de sensor  
    byteOut(6) = &H0 ' Modo del sensor = RAWMODE  
    SerialPort.Write(byteOut, 0, 7)  
  
    'GET INPUT VALUES  
    byteOut(0) = &H3 ' Cantidad de bytes que va a enviar  
    byteOut(1) = &H0 ' Siempre 0  
    byteOut(2) = &H0 ' &H0 = se espera respuesta &H80 = no  
                      se espera respuesta  
    byteOut(3) = &H7 ' Comando a enviar (GETINPUTVALUES)  
    byteOut(4) = &H2 ' PUERTO  
    SerialPort.Write(byteOut, 0, 5)  
  
    byteIn(0) = SerialPort.ReadByte 'Numero de bytes en la respuesta  
    byteIn(1) = SerialPort.ReadByte 'Debe ser 0 para el NXT.  
    For g = 2 To 1 + byteIn(0) 'Leemos el resto del mensaje.  
        byteIn(g) = SerialPort.ReadByte()  
    Next
```



```

Select Case byteIn(14) 'En el byte 14 se encuentra el numero asociado
                                                                al color
Case 1 'Color Negro
    Color.Text = "Negro" 'Texto a mostrar en el display
    Color.ForeColor = Drawing.Color.White 'Color del texto a mostrar
    Color.BackColor = Drawing.Color.Black 'Color de fondo del display
Case 2 'Color Azul
    Color.Text = "Azul"
    Color.ForeColor = Drawing.Color.White
    Color.BackColor = Drawing.Color.Blue
Case 3 'Color Verde
    Color.Text = "Verde"
    Color.ForeColor = Drawing.Color.Black
    Color.BackColor = Drawing.Color.LimeGreen
Case 4 'Color Amarillo
    Color.Text = "Amarillo"
    Color.ForeColor = Drawing.Color.Black
    Color.BackColor = Drawing.Color.Yellow
Case 5 'Color Rojo
    Color.Text = "Rojo"
    Color.ForeColor = Drawing.Color.White
    Color.BackColor = Drawing.Color.Red
Case 6 'Color Blanco
    Color.Text = "Blanco"
    Color.ForeColor = Drawing.Color.Black
    Color.BackColor = Drawing.Color.White

End Select

Catch ex As Exception

    TimerColor.Enabled = False 'En caso de error desactivamos el timer
    MsgBox("ERROR EN LA LECTURA DE COLOR") 'En caso de error
                                                mostramos este mensaje.

End Try

End Sub

Public Sub TimerColor_Tick(sender As Object, e As EventArgs) Handles
TimerColor.Tick
    Call SensorLuz_Click(sender, e) 'Llamamos a esta función cada 300ms
End Sub

```

- Al pinchar sobre los pulsadores “Luz Verde”, “Luz Azul” o “Luz Roja” lo que hacemos es mandar al NXT una trama que le indica al sensor que emita una luz a través del LED que posee, los tres colores a emitir son evidentemente el verde, el azul y el rojo, el código utilizado es el siguiente:

```

Private Sub LVerde_Click(sender As Object, e As EventArgs) Handles
LVerde.Click
    Dim byteOut(6) As Byte

    Try

```




```
'SET INPUT MODE
byteOut(0) = &H5           ' Cantidad de bytes que va a enviar
byteOut(1) = &H0           ' Siempre 0
byteOut(2) = &H80         ' &H0 = se espera respuesta &H80 = no se espera
                             respuesta
byteOut(3) = &H5           ' Comando a enviar (SETINPUTMODE)
byteOut(4) = &H2           ' Puerto
byteOut(5) = &HF           ' Emitir Luz Verde
byteOut(6) = &H0           ' Modo RAWMODE
SerialPort.Write(byteOut, 0, 7)

Catch ex As Exception
MsgBox("FALLO EN LA LAMPARA")
End Try
End Sub
```

Para los colores Azul y Rojo enviamos el mismo código, lo único que varía es el *byteOut(5)* donde para el color azul tiene el valor *&H10* y para el color rojo tiene el valor *&HE*.

- Al pinchar sobre el pulsador “Sigue-línea” se ejecutará un programa el cual hace que nuestro robot siga un rastro marcado por una línea negra, en el caso de no encontrar la línea negra realiza un barrido en su busca, el programa es el siguiente:

```
Private Sub Siguelinea_Click(sender As Object, e As EventArgs) Handles
Siguelinea.Click
Dim i, t As Integer           'Contadores
Velocidad.Text = "40"         'Se establece una velocidad
KeyPreview = True             'Establecemos que el formulario reciba todos los
                               elementos

Try
Call SensorLuz_Click(sender, e) 'Realizamos lectura de color

If Color.Text = "Negro" Then 'Mientras el color sea negro el robot avanza
Do While Color.Text = "Negro"
Call Adelante_MouseDown(sender, e)
Call SensorLuz_Click(sender, e)
Loop
i = 0                           'Inicializamos los contadores
t = 0

Elseif Color.Text <> "Negro" Then 'Si el color no es negro se realiza un
                                hasta encontrarlo

Do Until i = 5 Or Color.Text = "Negro"
Call Derecha_MouseDown(sender, e)
Call SensorLuz_Click(sender, e)
i = i + 1
Loop

Do Until t = 10 Or Color.Text = "Negro"
Call Izquierda_MouseDown(sender, e)
```



```
        Call SensorLuz_Click(sender, e)
        t = t + 1
    Loop

End If

If ModifierKeys = Keys.Control Then 'Si la tecla pulsada es CONTROL
                                     desactivamos el Sigue-línea
    Call Apagarluz_Click(sender, e)
Else 'Si no, se para levemente el robot y se activa un timer para
                                     volver a llamar al Sigue-línea
    Call Parada_Click(sender, e)
    TimerBucle.Enabled = True
End If

Catch ex As Exception
    MsgBox("NO SE ENCUENTRA LINEA A SEGUIR")
End Try

End Sub

Private Sub TimerBucle_Tick(sender As Object, e As EventArgs) Handles
TimerBucle.Tick 'Pasados 100ms se desactiva el timer y vuelve a
                                                         llamarse al Sigue-línea
    TimerBucle.Enabled = False
    Call Siguelinea_Click(sender, e)
End Sub
```

- Al pinchar sobre el pulsador “OFF” lo que hacemos es detener los timers que están activos, los motores, el Sigue-línea, apagamos el sensor de luz indicando al NXT que no hay ningún sensor conectado y reinicializar el display, el código utilizado es el siguiente:

```
Public Sub Apagarluz_Click(sender As Object, e As EventArgs) Handles
Apagarluz.Click
    Dim byteOut(6) As Byte

    TimerColor.Enabled = False 'Detenemos el timer para la lectura de color
    TimerBucle.Enabled = False 'Detenemos el timer para el siguelinea.

    Call Parada_Click(sender, e) 'Llamamos a la función de parada de
                                motores.

    Try
        'SET IMPUT MODE
        byteOut(0) = &H5 ' Cantidad de bytes que va a enviar - 2
        byteOut(1) = &H0 ' Siempre 0
        byteOut(2) = &H80 ' &H0 = se espera respuesta &H80 = no se espera
                                respuesta

        byteOut(3) = &H5 ' Comando a enviar (GETINPUTVALUES)
        byteOut(4) = &H2 ' Puerto
        byteOut(5) = &H0 'NO SENSOR, con este comando apagamos el
                                sensor de color.

        byteOut(6) = &H0
        SerialPort.Write(byteOut, 0, 7)
```



```
'Iniciando el display
Color.Text = ""
Color.BackColor = Drawing.Color.LightSteelBlue
Color.ForeColor = Drawing.Color.Gray

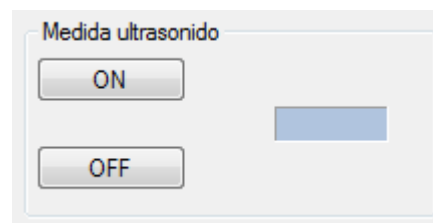
Catch ex As Exception
    MsgBox("ERROR")
End Try
End Sub
```

- **Medición de distancia a través del sensor de ultrasonido:**

Este sensor usa un protocolo de comunicación I2C con el NXT, por ello para obtener la lectura que realizar varios pasos:

- El primer paso es configurar el modo de funcionamiento del sensor a través de la trama SETINPUTMODE, a través del cual le indicamos al NXT el tipo de sensor que vamos a conectar, el modo de funcionamiento de dicho sensor y el puerto al que lo vamos a conectar.
- El segundo paso es indicar que queremos leer 2 bytes a través del bus I2C del registro 0x42 que es donde se encuentra la lectura realizada por el sensor de ultrasonido, la trama utilizada es LSWRITE.
- El tercer paso es esperar a que el dispositivo tenga listos los bytes de lectura, para ello utilizamos la trama LSGETSTATUS.
- Por últimos realizamos la lectura del registro a través de la trama LSREAD.

Para el control de este proceso se ha creado el siguiente grupo de pulsadores y displays:



Al pulsar sobre “ON” llamamos a la función “Medicionultrasonido” que incluye todas las tramas necesarias para realizar la lectura y mostrarla en el display, dentro de esta función también se activa un Timer que



cada 300ms llama a esta función y así tenemos una lectura continua del sensor.

Si mientras está activado el sensor de ultrasonidos se maneja el robot para desplazarlo este quedara bloqueado si encuentra algún obstáculo a una distancia menor de 20cm.

Y por último al pulsar sobre “OFF” llamamos a la función “ParadaUltrasonido” con la cual detenemos el Timer y limpiamos el display, dejando así de realizar lecturas sobre el sensor.

El código utilizado para este proceso es el siguiente:

```
Private Sub Medicionultrasonido_Click(sender As Object, e As EventArgs) Handles
Medicionultrasonido.Click 'Función que envía las tramas encargadas de activar el
                           sensor de ultrasonidos y leer la distancia.

    Dim byteOut(8) As Byte
    Dim byteIn(22) As Byte
    Dim Distancia, g As Integer
    g = 0

    Try

        'SET INPUT MODE      Establecemos el modo de funcionamiento del sensor de
                           ultrasonidos.

        byteOut(0) = &H5
        byteOut(1) = &H0
        byteOut(2) = &H80
        byteOut(3) = &H5           'Comando a enviar (SETINPUTMODE)
        byteOut(4) = &H0           'Puerto en el que estaría conectado el sensor
                                   (0 = puerto 1 del NXT)
        byteOut(5) = &HB           'Tipo de sensor (LOWSPEED_9V)
        byteOut(6) = &H0           'Modo del sensor (RAWMODE)
        SerialPort.Write(byteOut, 0, 7)

        'LS WRITE      Indicamos al sensor conectado en el puerto 1 que queremos leer 2
                           bytes en el registro 0x42.

        byteOut(0) = &H7
        byteOut(1) = &H0
        byteOut(2) = &H80
        byteOut(3) = &HF           'Comando a enviar (LSWRITE)
        byteOut(4) = &H0           'Puerto en el que estaría conectado el sensor
                                   (0 = puerto 1 del NXT)
        byteOut(5) = &H2           'Longitud de bytes que se quieren escribir en el bus I2C.
        byteOut(6) = &H2           'Longitud de bytes que se quieren recibir por el bus I2C.
        byteOut(7) = &H2           'Dirección del dispositivo I2C, para el NXT es 2.
        byteOut(8) = &H42         'Dirección del registro donde se guarda la lectura en el NXT.
        SerialPort.Write(byteOut, 0, 9)

        'LS GET STATUS      Esperamos hasta que haya 1 byte disponible para leer.

        byteOut(0) = &H3
        byteOut(1) = &H0
        byteOut(2) = &H0           'Si queremos respuesta.
        byteOut(3) = &HE           'Comando LSGETSTATUS
```



```
byteOut(4) = &H0 'Puerto 1
SerialPort.Write(byteOut, 0, 5)

byteIn(0) = SerialPort.ReadByte 'Número de bytes a recibir
byteIn(1) = SerialPort.ReadByte '0 para NXT
For g = 2 To 1 + byteIn(0) 'Leemos el resto del mensaje
    byteIn(g) = SerialPort.ReadByte()
Next

'LS READ Realizamos la lectura del registro.
byteOut(0) = &H3
byteOut(1) = &H0
byteOut(2) = &H0 'Indicamos que queremos respuesta
byteOut(3) = &H10 'Comando LSREAD
byteOut(4) = &H0 'Puerto 1
SerialPort.Write(byteOut, 0, 5)

byteIn(0) = SerialPort.ReadByte 'Número de bytes a recibir
byteIn(1) = SerialPort.ReadByte '0 para NXTXT
For g = 2 To 1 + byteIn(0) 'Leemos el resto del mensaje
    byteIn(g) = SerialPort.ReadByte()
Next

Distancia = byteIn(6) 'La distancia recogida por el sensor se encuentra en el
                        byte 6
Medida.Text = Distancia 'Mostramos la distancia en el display

If Distancia < 30 Then 'Si tenemos en marcha el robot a la vez que está leyendo
la distancia, el robot se detendrá si la distancia con un obstáculo es menor a 20cm.

    Call Parada_Click(sender, e) 'Llamamos a la función de Parada.

End If

TimerUltraSonic.Enabled = True 'Activamos el timer del ultrasonidos que
                                preguntará por la distancia cada 300ms

Catch ex As Exception
    MsgBox("ERROR EN LA MEDICION") 'Mensaje a lanzar en caso de error.
End Try

End Sub

Private Sub ParadaUltrasonido_Click(sender As Object, e As EventArgs) Handles
ParadaUltrasonido.Click 'Se detiene la lectura de distancia a través del sensor US.
    TimerUltraSonic.Enabled = False 'Detenemos el timer.
    Medida.Text = "" 'Ponemos el display a 0.
End Sub

Private Sub TimerUltraSonic_Tick(sender As Object, e As EventArgs) Handles
TimerUltraSonic.Tick 'Timer del US, Llama a la función de lectura por US cada 300ms.
    Call Medicionultrasonido_Click(sender, e)

End Sub
```



- **Configuración del Bluetooth:**

Para la realización de este proyecto hemos tenido que utilizar un Bluetooth HC-05 para poder configurarlo como maestro y así poder conectarse con nuestro NXT y enviar y recibir tramas sin problemas, para ello hemos tenido que configurarlo mediante comandos AT, con estos comandos podemos programarlo en modo maestro o esclavo, el baudrate, el nombre, la contraseña, el modo de conexión para que trabaje con uno o varios dispositivos, la dirección del dispositivo con el cual queremos que trabaje, etc...

Para entrar en la configuración del Bluetooth se ha creado el siguiente pulsador:



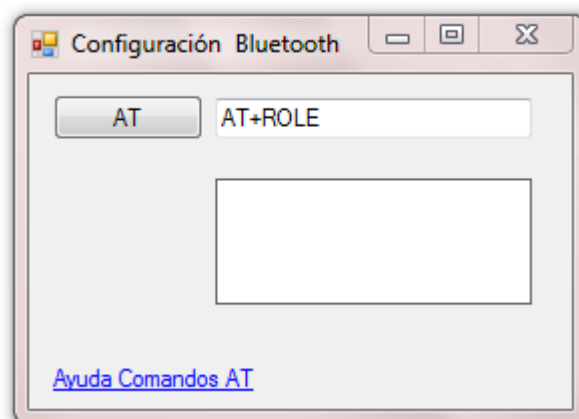
El cual llama a la función "ButtonBluetooth" con el siguiente código:

```
Private Sub ButtonBluetooth_Click(sender As Object, e As EventArgs) Handles  
ButtonBluetooth.Click
```

```
    CAT.Show()  
    TimerBatería.Enabled = False
```

```
End Sub
```

Esta función abre una nueva ventana para poder comunicarnos con el dispositivo Bluetooth, al abrir esta aplicación se desactiva el Timer de la batería para evitar problemas:





En el cuadro de texto se escribe el comando AT que queremos ver o cambiar de nuestro Bluetooth y cuando pulsamos el pulsador “AT” se envía dicha información y lo que recibimos se muestra en el cuadro de abajo. También se ha creado un hipervínculo el cual al ser pulsado te abre una página web con las instrucciones y comandos necesarios para programar el dispositivo Bluetooth.

Los comandos más utilizados para la configuración del Bluetooth son los siguientes (en el Anexo1 se incluye manual completo del HC-05):

- AT: comprobar la conexión.
- AT+NAME: ver el nombre por defecto.
- AT+ADDR: ver la dirección por defecto.
- AT+VERSION: ver la versión.
- AT+UART: ver la velocidad de transmisión.
- AT+ROLE: ver el rol del módulo Bluetooth (1 = master, 0 = esclavo).
- AT+RESET: Reset y salir del modo AT.
- AT+ORGL: restaurar a la configuración de fábrica.
- AT+PSWD: ver la contraseña por defecto

El código utilizado es el siguiente:

Public Class CAT

Private Sub ButtonAT_Click_1(sender As Object, e As EventArgs) Handles ButtonAT.Click

Dim i, t As Integer
Dim AT As String = TextBoxAT.Text 'Variable para guardar lo escrito en el cuadro de comandos.

Dim byteOut(30) As Byte
Dim bytein(30) As Byte
Dim respuesta As Integer 'Variable que va a guardar la longitud de la respuesta
Dim a As String = Len(AT) 'Variable que guarda la longitud del comando a enviar.

RespuestaAT.Text = String.Empty 'Limpiamos el cuadro de respuesta.

Try
For i = 1 To a 'Cargamos cada carácter del cuadro de comandos en una variable.
byteOut(i) = Asc(Mid(AT, i, 1)) 'Se transforma cada carácter en código ASCII para poder transmitirlos.

Next
byteOut(a + 1) = &HD 'Terminaciones necesarias para indicar final de carro y



```
byteOut(a + 2) = &HA          nueva línea en el comando
                                enviado.

NXT.SerialPort.Write(byteOut, 1, a + 2)      'Enviamos el comando

While NXT.SerialPort.BytesToRead = 0          'Esperamos a que haya respuesta
End While

respuesta = NXT.SerialPort.BytesToRead        'Guardamos el número de bytes que
                                                se van a recibir en una variable

For t = 1 To respuesta                        'Copiamos los bytes de respuesta en una variable
    bytein(t) = NXT.SerialPort.ReadByte

    RespuestaAT.Text = RespuestaAT.Text + Chr(bytein(t)) 'Se transforma cada
                                                            byte a Char y se van mostrando en el cuadro de respuesta.
Next

Catch ex As Exception
    MsgBox("FALLO EN COMANDOS AT") 'Muestra este mensaje en caso de error.
End Try

End Sub

Private Sub LinkAyuda_LinkClicked(sender As Object, e As
LinkLabelLinkClickedEventArgs) Handles LinkAyuda.LinkClicked 'Función que nos
permite acceder a un enlace de una página web para ayudar con la configuración del
Bluetooth.
    LinkAyuda.LinkVisited = True 'Activamos el link

    System.Diagnostics.Process.Start("http://www.naylampmechatronics.com/blog/24_config
uracion-del-modulo-bluetooth-hc-05-usa.html")
End Sub

End Class
```




CAPITULO 5. CONCLUSIONES

I. Conclusiones

Como conclusiones podemos decir que se ha logrado el objetivo de este proyecto, que es el control de un robot Lego NXT por Bluetooth a través del bus CAN del microcontrolador 8051.

Un punto importante de este proyecto es que, al haber creado una aplicación de control mediante VB, se puede utilizar cualquier elemento para la transmisión de los comandos siempre y cuando este elemento se comunique con el robot mediante Bluetooth. Y viceversa, al tener un sistema de comunicación bidireccional mediante Bluetooth podemos usar otros programas para crear una aplicación que controle el robot NXT.

A pesar de haber cumplido los objetivos, siempre puede mejorarse creando más comandos de control para el NXT y los diferentes sensores que este posee, ya que el NXT al ser un robot de Lego pueden crearse una gran cantidad de robots distintos y cada uno puede programarse con habilidades propias.



CAPITULO 6. BIBLIOGRAFIA

I. Bibliografía

1. Apuntes de la asignatura “Informática y Comunicaciones Industriales”.
2. <http://kio4.com/b4a/29bluetooth.htm>
3. <http://www.robotappstore.com/Knowledge-Base/NXT-File-Handling-over-Bluetooth--Introduction/25.html>
4. <http://www.extremenxt.com/vbfront.htm>
5. <http://drblackadder.net/Research/NXT%20I2C%20Communication/>
6. https://www.rcscomponents.kiev.ua/datasheets/hc_hc-05-user-instructions-bluetooth.pdf
7. <http://saber.patagoniatec.com/modulo-bluetooth-hc-06-bluetooth-arduino-slave-hc06-esclavo-iot/>
8. http://www.naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05-usa.html
9. http://www.aprenderaprogramar.es/index.php?option=com_content&view=category&id=37&Itemid=61
10. LEGO MINDSTORMS NXT Bluetooth Developer Kit.
11. LEGO MINDSTORMS NXT Hardware Developer Kit.
12. Manual C8051F04X de Silicon Laboratories.
13. Manual LEGONXT.
14. Wikipedia.