UNIVERSIDAD DE SONORA

DIVISIÓN DE INGENIERÍA

INGENIERÍA EN MECATRÓNICA

Tesis

Control de motores de corriente directa por medio de Zigbee

Presenta:

Carlos Alberto León Mercado

Víctor Hugo Benítez Baltazar
Director de Tesis

Hermosillo, Sonora **Octubre de 2012**

Universidad de Sonora

Repositorio Institucional UNISON

Contenido

Capítulo 1

1. Planteamiento del problema

En este capítulo se plantea el problema al cual se le diseñará una solución. Se justifica y se delimita el proyecto. Se establecen los objetivos además de una metodología experimental en el cual se desarrolla a grandes rasgos el proyecto de tesis.

1.1 Antecedentes

La Universidad de Sonora en conjunto con la Universidad Nacional Autónoma de México (UNAM) cuenta con un campo de pruebas de helióstatos (CPH) Fig. 1.1, el cual será utilizado como un prototipo de fuente alternativa de energía renovable. El CPH cuenta con un gran número de helióstatos los cuales deben ser controlados independientes uno de otro. Para ello se utiliza una demasiado cableado, para el control de cada uno lo cual resulta difícil al querer detectar algún problema de comunicación. Dada esta circunstancia es que se comenzó a investigar otra forma de comunicación que no requiriera tanta infraestructura de cableado para su funcionamiento como la comunicación inalámbrica.

Fig. 1.1 Campo de helióstatos.

1.2 Problema actual

El uso de la comunicación inalámbrica para la intercomunicación entre dos o más dispositivos a distancia surge como una solución a los problemas de espacio y costo. Utilizar una conexión alámbrica genera toda una infraestructura de cableado para la intercomunicación lo cual requiere de una mayor inversión económica comparado con una conexión inalámbrica. Actualmente, se desarrolló una propuesta de control inalámbrico en algunos heliostatos en el CPH. Sin embargo, en campo, dicho control no resultó viable debido a problemas de interferencia. Dado que la tecnología implementada fue adquirida a un proveedor, no es posible determinar donde se encuentra el problema.

El proyecto actual pretende servir de base para realizar pruebas en un entorno de desarrollo en el cual se pueda manipular la programación y encontrar solución al problema de comunicación.

1.3 Objetivo General

Controlar velocidad, dirección, arranque y paro de un motor de corriente directa por medio de un microcontrolador de 8 bits mediante una conexión inalámbrica Zigbee.

1.4 Objetivo Especifico

- Selección del microcontrolador.
- Diseño de una arquitectura de comunicación inalámbrica.
- Control de un motor de DC por medio de la arquitectura inalámbrica.
- Diseño de un prototipo demostrativo.

1.5 Justificación

En la Universidad de Sonora actualmente no existen antecedentes o documentación alguna de proyectos de esta naturaleza, por lo cual surge esta alternativa de control de dispositivos por medio de una comunicación inalámbrica y así de esta manera en un futuro al desarrollo de proyectos universitarios y externos.

1.6 Delimitación

Se controlará la velocidad, la dirección, el arranque y el paro de un motor de corriente directa mediante una conexión inalámbrica. No se contempla implementar una acción de control de mayor complejidad como pudiera ser un PID. Esta decisión está justificada en el hecho de que por el momento se desea introducir la técnica de la comunicación inalámbrica y basta con el control simple de las variables citadas.

1.7 Metodología

En la metodología llevada a cabo primero se estudian los microcontroladores con sus características y aplicaciones para seleccionar el más adecuado para el proyecto. Además, el diseño de una red comunicación inalámbrica la cual cumpla con los requerimientos necesarios. Una vez que se tienen los anteriores se ponen en práctica para el control con motores DC para fines del proyecto.

- Selección del microcontrolador.
- Diseño de una arquitectura de comunicación inalámbrica.
- Control de un motor de DC por medio de la arquitectura inalámbrica.
- Diseño de un prototipo demostrativo.

Capítulo 2

2. Estado del arte

En este capítulo se describe la teoría fundamental de los elementos que conforman el proyecto de tesis. Teoría de microcontroladores, como seleccionar el microcontrolador adecuado, los principales fabricantes y sus entornos de programación. La comunicación inalámbrica y sus protocolos, el estándar de comunicaciones inalámbricas IEEE 802.15.4. Motores, sus partes y tipos.

2.1 Microcontroladores

2.1.1 Fundamentos

Un microcontrolador es un circuito integrado digital monolítico que contiene todos los elementos de un procesador digital secuencial síncrono programable de arquitectura Harvard o Princeton (Von Neumann). Se le suele denominar también como computador integrado o empotrado (*Embedde procesor*) y esta especialmente orientado a tareas de control y comunicaciones.

Por su pequeño tamaño los microprocesadores permiten empotrar un procesador programable en muchos productos industriales. Su costo reducido y consumo de energía y velocidad adaptables, resultan apropiados para numerosas aplicaciones.

Los microcontroladores se utilizan para la realización de sistemas eléctricos empotrados en otros sistemas (eléctricos, mecánicos, etcétera, Fig. 2.1) [1]

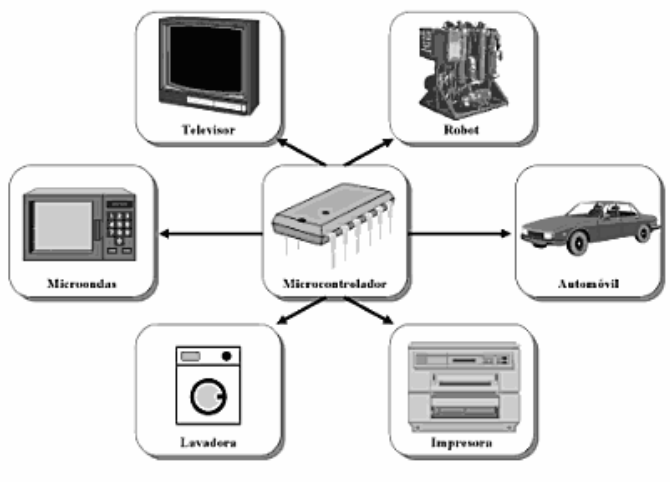

Fig. 2.1 Algunos ejemplos de campos de aplicación de los microcontroladores.

Un sistema embebido consta de diversos componentes digitales. (Fig. 2.2)

PERIFERICOS	I/O
CPU (Unidad Central de Procesamiento)	Memoria

Fig.2.2 Componentes de un sistema embebido.

Un microcontrolador contiene un *CPU* (unidad central de procesamiento), una memoria *ROM* (memoria de sólo lectura), *ó RAM* (memoria de acceso aleatorio), entradas y salidas para comunicación externa y periféricos los cuales pueden ser *PWMs* (modulación de ancho de pulso), convertidores analógicos digitales, *timers*, etcétera.

2.1.2 Arquitectura de las computadoras

Se define como la organización que tienen los diversos elementos que lo componen, así como la forma que tienen de interaccionar entre ellos al realizar sus operaciones normales. La

arquitectura con la que está diseñada una computadora define su comportamiento y sus posibilidades.

Aunque anteriormente se había realizado y probado otros modelos, en 1945 Von Neumann definió el modelo básico de arquitectura de una computadora digital que se utiliza actualmente en la mayoría de las computadoras. (Fig.2.3) [2]

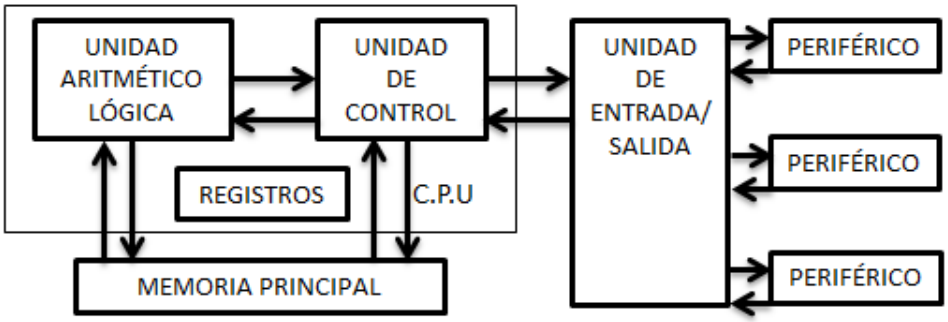

Fig.2.3 Arquitectura de una computadora.

Esta arquitectura es capaz de ejecutar una seria de instrucciones elementales llamadas instrucciones máquina; las cuales deben estar almacenadas en la memoria principal para poder ser leídas y ejecutadas. Las unidades funcionales de este modelo de computador son las siguientes: [2]

- **Memoria principal**

Es un dispositivo de almacenamiento de información dividido en celdas que se identifican mediante direcciones. Las celdas son todas iguales, del mismo tamaño o número de bits, y contienen tanto datos como instrucciones.

Los programas que se realicen serán almacenados temporalmente en la memoria, para después ser ejecutados. Los datos que utilice el programa deberán estar también en la memoria, y los datos que se produzcan se almacenarán también en ella.

La instrucción siguiente que debe ejecutar la computadora se determina en base a un registro especial de la *CPU* denominado contador de programa. Este registro contiene en cada momento la dirección en memoria principal de la siguiente instrucción que debe ser ejecutada.

- **Unidad Central de Proceso (*CPU*)**

Es el cerebro de la computadora y quien realmente gobierna y ejecuta todos los cálculos. Está compuesta a su vez de dos unidades funcionales, la unidad aritmético-analógica y la unidad de control. Además contiene varios registros para el control del estado de la computadora. Uno de estos registros es el *PC*, mencionado anteriormente. Cuando el *CPU* está integrado totalmente en un chip o pastilla, se le denomina microprocesador.

- **Unidad aritmético-lógica (*ALU*)**

En este bloque se realiza un conjunto finito de operaciones aritméticas y lógicas elementales tales como suma, resta, *AND*, *OR*, etcétera. Los datos sobre los cuales opera provienen de la memoria principal, y pueden estar almacenados temporalmente en los registros (pequeñas unidades de memoria) contenidos en el *CPU* la cantidad de registros de la *CPU* varia dependiendo del computador.

- **Unidad de control**

Se encarga de leer las instrucciones máquina almacenadas en la memoria principal, decodificarlas, y generar las señales de control de bajo nivel necesarias para que cada instrucción sea ejecutada. Como ya se ha mencionado antes, existe un registro llamado contador de programa que contiene la información de la posición de memoria (dirección) que contiene la siguiente instrucción a ejecutar. Cada instrucción ejecutada modifica el contador del programa, incrementándolo o modificándolo sustancialmente su valor para que este indique la dirección de su valor.

La unidad de control decodifica cada instrucción máquina de memoria, genera las señales de control adecuadas y según la instrucción decodificada, realizará la actualización del contador del programa.

- **Unidad de entrada-salida**

Realiza la transferencia de información con las unidades exteriores del computador, llamadas periféricos: impresoras, discos, modem, teclado, pantalla, etcétera.

- **Reloj**

Es un oscilador de frecuencia fija que sincroniza las operaciones del resto de componentes de la computadora. La frecuencia de la computadora define la velocidad de ejecución de las instrucciones y depende de la velocidad relativa de los circuitos semiconductores. [2]

Arquitectura Harvard

La arquitectura Harvard utiliza memorias separadas para instrucciones y datos. En este caso la memoria de programa tiene su bus de direcciones, y su propio bus de datos y su bus de control. Por otra parte, la memoria de datos tiene sus propios buses de direcciones, datos de control, independientes de los buses del programa. La memoria de programa es solo de lectura, mientras que en los datos se puede leer y escribir. (Fig. 2.4)

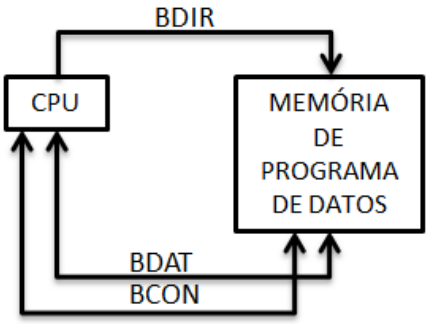

Fig. 2.4 Arquitectura Harvard.

Arquitectura Von Neumann

La arquitectura Von Neumann requiere menos líneas que la Harvard para conectar el *CPU* con la memoria, lo cual significa una conexión más simple entre ambas. Pero con esta arquitectura es imposible manipular simultáneamente datos e instrucciones, debido a la estructura de buses únicos, algo que si es posible con la arquitectura Harvard, que tiene buses separados. Esto confiere a la arquitectura Harvard la ventaja de mayor velocidad de ejecución de los programas. (Fig. 2.5)

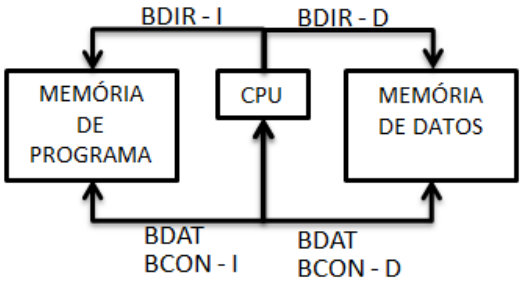

Fig. 2.5 Arquitectura Von Neumann.

En las microcomputadoras, la *CPU* es un circuito integrado; en cambio el microprocesador, es obvio que la arquitectura Von Neumman requiere menos terminales en el microprocesador que la arquitectura Harvard. Esto fue una razón decisiva para que desde sus inicio los microcomputadores basados en microprocesador se hayan diseñando utilizando casi exclusivamente la arquitectura Von Neumman. En los microcontroladores la situación es diferente. Al estar todos los componentes del sistema dentro del circuito integrado, desaparece la necesidad de minimizar el número de terminales de la *CPU*, de modo que en ellos ha predominado la arquitectura Harvard. Los microcontroladores PIC son un ejemplo de sistemas con arquitectura Harvard. [3]

2.1.3 Principales fabricantes de microcontroladores

- **ATMEL**

ATMEL fabrica los microcontroladores de la familia *AVR*, esta nueva tecnología proporciona todos los beneficios habituales de arquitectura *RISC* y memoria flash reprogramable eléctricamente. La característica que los identifica a estos microcontroladores de ATMEL es la memoria flash y *EEPROM* que incorpora. La firma también produce y vende varios subproductos de la popular familia 8051 con la diferencia de que están basados en la memoria flash. El diseño *AVR* de ATMEL difiere de los demás microcontroladores de 8 bits por tener mayor cantidad de registros (32) y un conjunto ortogonal de instrucciones (Un set de instrucciones es ortogonal cuando las instrucciones pueden usarse con cualquier registro y modo de direccionamiento, es decir, cualquier instrucción puede utilizar cualquier elemento de la arquitectura como fuente o destino). *AVR* es mucho más moderna que su competencia. Esto

hace que la arquitectura *AVR* sea más fácil de programar a nivel de lenguaje ensamblador y que sea fácil de optimizar con un compilador. El gran conjunto de registros disminuye la dependencia respecto a la memoria, lo cual mejora la velocidad y disminuye las necesidades de almacenamiento de datos. Además casi todas las instrucciones se ejecutan en 1 ó 2 ciclos de reloj contra 5-10 ciclos de reloj para los chips *8051, 6805, 68HC11* y *PIC*. Los microcontroladores *AVR* tienen *pipeline* con dos etapas (cargar y ejecutar), que les permite ejecutar la mayoría de las instrucciones en un ciclo de reloj, lo que los hace relativamente rápidos entre los microcontroladores de 8-bit.

Adicionalmente, ATMEL también proporciona en línea el entorno software (*AVR* estudio) [10] que permite editar, ensamblar y simular el código fuente. Una vez ensamblado y depurado el código fuente del programa, se transferirá el código máquina a la memoria flash del microcontrolador para esto se debe disponer de otro entorno de desarrollo para programar en forma serial o paralelo la memoria flash.

- **MICROCHIP**

Microchip ofrece soluciones para la gama completa de 8-bits, 16-bits y microcontroladores de 32-bits, con una poderosa arquitectura, las tecnologías de memoria flexibles, integrales y fáciles de usar herramientas de desarrollo, documentación técnica completa y posterior diseño de soporte. Beneficios obtenidos: [12]

- Fácil migración a través de familias de productos.
- Bajo riesgo de desarrollo de productos y mas rápida al mercado.
- Reducir el costo total del sistema.
- Soporte técnico 24/7.
- Servicios de programación de la producción.
- Calidad certificada.
- Fácil entorno de programación.

2.1.4 Entornos de programación

Un Entorno de Desarrollo Integrado (*IDE*) es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de

programación o bien, poder utilizarse para varios. Un *IDE* puede denominarse como un entorno de programación que ha sido tratado como un programa aplicación. Esto significa que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

Los componentes de cualquier entorno de desarrollo integrado son un editor de texto, un compilador, un intérprete, un depurador, que tenga posibilidad de ofrecer un sistema de control de versiones y que ayude en la construcción de interfaces gráficas de usuario.

MPLAB-IDE de MICROCHIP

MPLAB-IDE es un programa software que se ejecuta sobre una computadora para desarrollar aplicaciones para microcontroladores de MICROCHIP [12]. El MPLAB IDE constituye un entorno de desarrollo integrado distribuido gratuitamente por Microchip (fabricante de los microcontroladores *PIC*) desde su página web [12]. MPLAB-IDE incorpora todas las utilidades necesarias para la realización de cualquier proyecto y, para los que no dispongan de un emulador, el programa permite editar el archivo fuente en lenguaje ensamblador de nuestro proyecto, además de ensamblarlo y simularlo en pantalla, pudiendo ejecutarlo posteriormente en modo paso a paso y ver como evolucionarían de forma real tanto sus registros internos, la memoria *RAM* y/o *EEPROM* de usuario como la memoria de programa, según se fueran ejecutando las instrucciones. Además el entorno que se utiliza es el mismo que si se estuviera utilizando un emulador.

Herramientas accesibles desde MPLAB-IDE:

- Gestión de Proyectos.
- Edición del texto de los programas escritos en ensamblador ó *C.*
- Ensamblador, Compiladores y Montador de Enlaces.
- Depuración del código por simulación o sobre el circuito real.
- Programación o grabación final de microcontroladores.

MPLAB es un simulador de eventos discretos. No se trata de una simulación a la velocidad que desarrollará el microcontrolador. Las instrucciones se ejecutan tan rápido como puede la *CPU* de la computadora donde se esté ejecutando MPLAB. Esto significa que será

normalmente más lento que el microcontrolador real trabajando a la frecuencia que marque su oscilador. Sin embargo cuenta con utilerías que le permiten implementar aplicaciones que requieren de tiempos muy precisos como el *Stop Watch*. (La Fig. 2.6 muestra la ventana del entorno de programación de MPLAB-IDE)

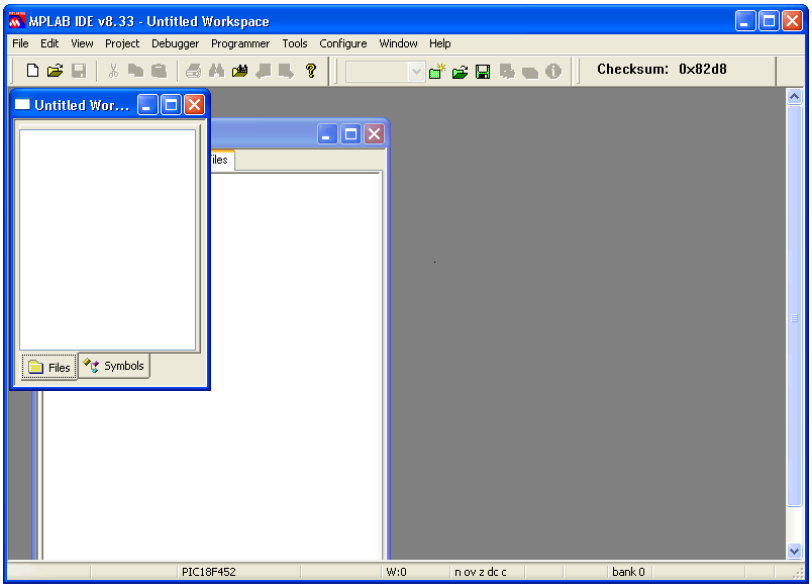

Fig. 2.6 Entorno de programación MPLAB-IDE de Microchip.

SKETCH-IDE de ARDUINO

El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino y el entorno de desarrollo del mismo nombre. El entorno de código abierto Arduino hace fácil escribir código y cargarlo a la placa E/S. Funciona en *Windows*, *Mac OS X* y *Linux*. El entorno está escrito en *Java* y basado en *Processing*, *AVR-gcc* y otros programas también de código abierto. El software de desarrollo es abierto y es posible descargarlo gratis. [15]

El entorno de desarrollo Arduino está constituido por un editor de texto para escribir el código, un área de mensajes, una consola de texto, una barra de herramientas con botones para las funciones comunes, y una serie de menús. Permite la conexión con el hardware de Arduino para cargar los programas y comunicarse con ellos.

Arduino utiliza para escribir el código lo que denomina "sketch" (programa). Estos programas son escritos en el editor de texto. Existe la posibilidad de cortar/pegar y buscar/remplazar texto. En el área de mensajes se muestra información mientras se cargan los programas y

también muestra errores. La consola muestra el texto de salida para el entorno de Arduino incluyendo los mensajes de error completos y otras informaciones. La barra de herramientas permite verificar el proceso de carga, creación, apertura y guardado de programas, y la monitorización serie.

El entorno de programación es fácil de usar para principiantes y lo suficientemente flexible para los usuarios avanzados. El software está publicado bajo una licencia libre y se encuentra preparado para ser ampliado por programadores experimentados. El lenguaje puede ampliarse a través de librerías de *C++*, y si se está interesado en profundizar en los detalles técnicos, se puede dar el salto a la programación en el lenguaje *AVR C* en el que está basado. De igual modo se puede añadir directamente código en *AVR C* en los programas si así se desea.

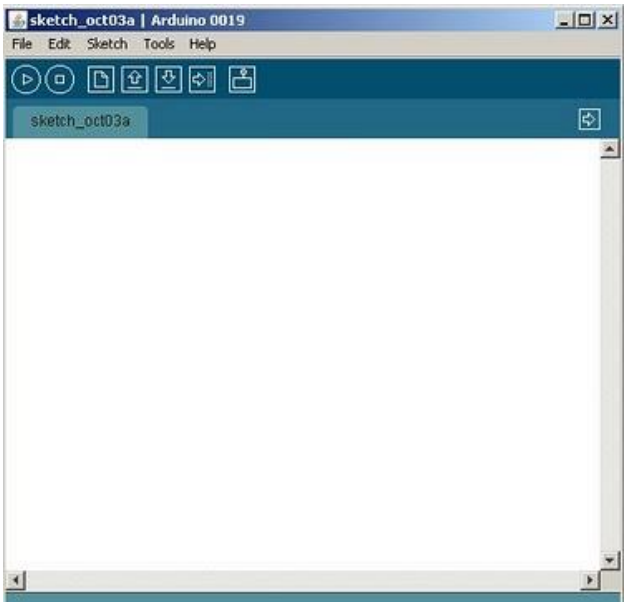

Fig. 2.7 Entorno de programación Sketch-IDE de Arduino.

2.1.5 Selección del microcontrolador adecuado

Al momento de seleccionar el microcontrolador para la realización de un proyecto se deben tener en cuenta diversos factores, como documentación, herramientas de desarrollo disponible, el precio, la cantidad de fabricantes que lo producen, las características técnicas de microcontrolador (tipo de memoria de programa, interrupciones, numero de temporizadores, entre otras).

Existen muchas compañías fabricantes de microcontroladores y constantemente compiten para vender sus productos. Es importante el costo de producción porque además de este, el consumidor también cubre los gastos de diseño del hardware, empaquetado, componentes, y el desarrollo del software. Entonces si el costo de todo el proceso de creación del microcontrolador es elevado, como resultado se tendrá un precio elevado del microcontrolador. Si el consumidor desea reducir costos debe tener en cuenta las herramientas de apoyo con que va a contar como emuladores, simuladores, ensambladores, compiladores, etcétera. Es por ello que mayormente los consumidores se inclinan con obtener productos de la misma familia.

Para poder seleccionar el microcontrolador adecuado es necesario reconocer los requisitos mínimos para la aplicación a realizar: [16]

- **Procesamiento de datos**

Puede requerirse que el microcontrolador realice cálculos críticos en un tiempo limitado. En ese caso se debe asegurar de seleccionar un dispositivo suficientemente rápido para ello. Por otro lado, habrá que tener en cuenta la precisión de los datos a manejar: si no es suficiente con un microcontrolador de 8 bits, puede ser necesario acudir a microcontroladores de 16 ó 32 bits, o incluso a hardware de coma flotante. Una alternativa más barata y quizá suficiente es usar librerías para manejar los datos de alta precisión.

- **Entrada y salida**

Para determinar las necesidades de Entrada/Salida del sistema es conveniente dibujar un diagrama de bloques del mismo, de tal forma que sea sencillo identificar la cantidad y tipo de señales a controlar. Una vez realizado este análisis puede ser necesario añadir periféricos hardware externos o cambiar a otro microcontrolador más adecuado a la aplicación.

- **Consumo**

Algunos productos que incorporan microcontroladores están alimentados con baterías y su funcionamiento puede ser tan vital como activar una alarma antirrobo. Lo más conveniente en un caso como éste puede ser que el microcontrolador esté en estado de bajo consumo pero que despierte ante la activación de una señal (una interrupción) y ejecute el programa adecuado para procesarla.

- **Memoria**

Para detectar las necesidades de memoria la aplicación debemos separarla en memoria volátil (*RAM*), memoria no volátil (*ROM, EPROM*, etcétera) y memoria no volátil modificable (*EEPROM*). Este último tipo de memoria puede ser útil para incluir información específica de la aplicación como un número de serie o parámetros de calibración.

El tipo de memoria a emplear vendrá determinado por el volumen de ventas previsto del producto: de menor a mayor volumen será conveniente emplear *EPROM, OTP* y *ROM*. En cuanto a la cantidad de memoria necesaria puede ser imprescindible realizar una versión preliminar, aunque sea en pseudocódigo, de la aplicación y a partir de ella hacer una estimación de cuánta memoria volátil y no volátil es necesaria y si es conveniente disponer de memoria no volátil modificable.

- **Ancho de la palabra**

El criterio de diseño debe ser seleccionar el microcontrolador de menor ancho de palabra que satisfaga los requerimientos de la aplicación. Usar un microcontrolador de 4 bits supondrá una reducción en los costes importante, mientras que uno de 8 bits puede ser el más adecuado si el ancho de los datos es de un byte. Los microcontroladores de 16 y 32 bits, debido a su elevado coste, deben reservarse para aplicaciones que requieran de sus altas prestaciones (Entrada/Salida potente o espacio de direccionamiento muy elevado).

- **Diseño de la placa**

La selección de un microcontrolador concreto condicionará el diseño de la placa de circuitos. Debe tenerse en cuenta que quizá usar un microcontrolador barato encarezca el resto de componentes del diseño.

2.1.6 Tipos de Microcontroladores

Los diversos tipos de microcontroladores varían dependiendo de la finalidad con la cual serán utilizados, como por ejemplo, el control de algún electrodoméstico, hasta una computadora. De eso dependerá la correcta selección del microcontrolador para alguna tarea determinada. [5]

- **Gama baja**

De 4, 8 y 16 bits. Dedicados fundamentalmente a tareas de control (electrodomésticos, cabinas telefónicas, *smart-cards*, algunos periféricos de computadoras, etc.) Generalmente son µC.

- **Gama media**

De 16 y 32 bits. Utilizados para tareas de control con cierto grado de procesamiento (control en automóvil, teléfonos móviles, *PDA*,…) Suelen ser periféricos integrados, y memoria externa.

- **Gama alta**

De 32, 64 y 128 bits. Dedicados Fundamentalmente a procesamiento (computadoras, videoconsolas, etc.) Casi en su totalidad son µP + circuitería periférica + memoria [5]

Características

Microcontroladores de 4 bits

- Pocos bytes de *RAM.*
- Sin *SO.*
- Todo el software en ensamblador.
- Cada vez menos usados.

Microcontroladores de 8 bits

- RAM de unos pocos bytes a unos cientos de KB.
- Usan ensamblador, pero también *C, C++, Java.*
- Pueden llevar *SO* específico.

Microcontroladores de 16 y 32 bits

- *RAM* de pocos KB a muchos MB.
- Generalmente llevan un *SO* de tiempo-real.
- Pueden o no tener cachés.

Microcontroladores de 32 ó 64 bits

- Básicamente un PC en un encapsulado pequeño.
- Llevan *Windows XP, Linux.*

- Relativamente caros.

En la siguiente tabla se muestran algunos campos de aplicación de microcontroladores:

Bits	Campo de aplicación
4	• Aplicaciones sensibles (juguetes, etcetera) • Número limitado de entradas y salidas • Entornos industriales especificos • Telefonia y electrodomésticos
8	• Entorno y datos orientados al "*byte*" • Aplicaciones sensibles • Periféricos inteligentes y controladores (teclados, unidades de disco)
16	• Manejo de operaciones de 16 bits • Mayor velocidad, operaciones matemáticas • Manejo de grandes volumenes de datos • Industria automotriz, grandes periféricos
32	• Manejo de grandes cantidades de datos • Gran capacidad de direccionamiento de memoria • Impresoras laser, pantallas graficas de muy alta resolución

Tabla 2.1 Campos de aplicación de microcontroladores.

2.2 Comunicación inalámbrica

2.2.1 Redes inalámbricas

Una red inalámbrica es una conexión en la que dos o más terminales se pueden comunicar sin la necesidad de una conexión por cable. Con las redes inalámbricas el dispositivo puede mantenerse conectado cuando se desplaza dentro de una determinada área geográfica.

Las redes inalámbricas se basan en un enlace que utiliza ondas electromagnéticas (radio e infrarrojo) en lugar de cableado estándar. Hay muchas tecnologías diferentes que se diferencian por la frecuencia de transmisión que utilizan, y el alcance y la velocidad de sus transmisiones.

Las redes inalámbricas permiten que los dispositivos remotos se conecten sin dificultad, ya se encuentren a unos metros de distancia como a varios kilómetros. Asimismo, la instalación de estas redes no requiere de ningún cambio significativo en la infraestructura existente como pasa con las redes cableadas. Tampoco hay necesidad de perforar las paredes para pasar cables ni de instalar portacables o conectores. Esto ha hecho que el uso de ésta tecnología se extienda con rapidez.

Una red inalámbrica hace lo mismo que cualquier otra red de computadoras, conecta equipos formando redes de computadoras pero sin la necesidad de cables. Se puede proveer acceso a otras computadoras, bases de datos, Internet, y en el caso de *WirelessLans*, el hecho de no tener cables, les permite a los usuarios contar con movilidad sin perder la conexión.

Por lo general, las redes inalámbricas se clasifican en varias categorías, de acuerdo al área geográfica desde la que el usuario se conecta a la red (denominada *área de cobertura*). (Fig. 2.8)

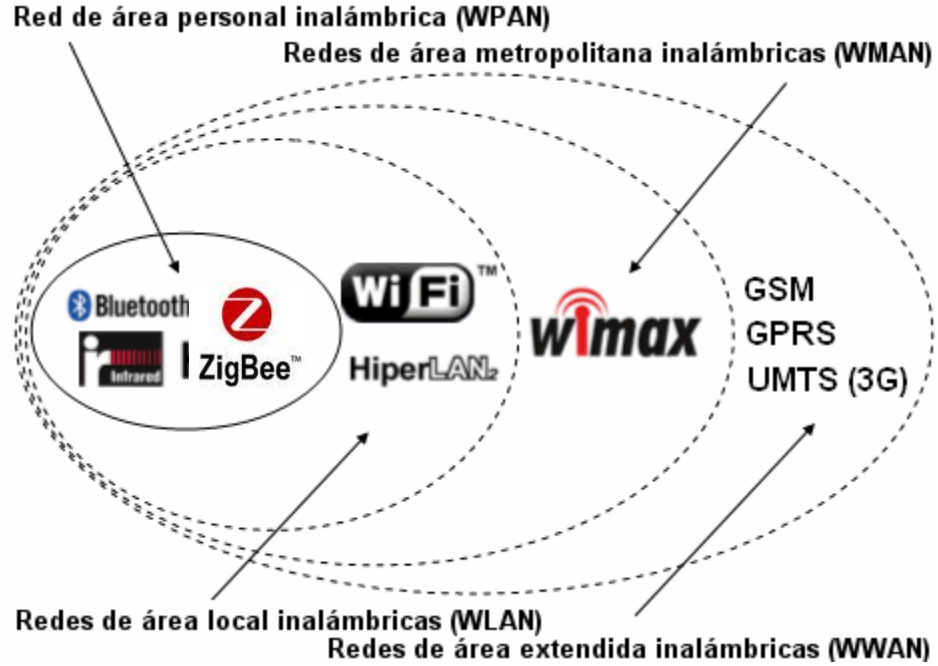

Fig. 2.8 Categorías de las redes inalámbricas.

- ***Wireless PAN*** es aquella que permite interconectar dispositivos electrónicos dentro de un rango pequeño, aproximadamente entre 9 y 30 metros (por ejemplo, concentradores *LAN*, otros dispositivos móviles, teléfonos móviles, computadoras y otros dispositivos como

impresoras y cámaras) para comunicar y sincronizar información. Las tecnologías líderes en esta área son *Bluetooth* y Zigbee.

- ***Wireless LAN*** es aquella que permite conectar una red de computadores en una localidad geográfica pequeña, de manera inalámbrica para compartir archivos, servicios, impresoras, y otros recursos. Estas redes soportan generalmente tasas de transmisión entre los 11 y 54 mega bits por segundo (mbps) y tienen un rango de entre 30 a 300 metros, con señales capaces de atravesar paredes. Las *Wireless LANs* ofrecen muchas ventajas como movilidad, flexibilidad, escalabilidad, velocidad, simplicidad y costos reducidos de instalación. Son una solución para edificios que por su arquitectura, o su valor histórico, no pueden ser perforados para instalar cableado estructurado.
- ***Wireless WAN*** es aquella en la que se pueden conectar las diferentes localidades utilizando conexiones satelitales, o por antenas de radio microondas. Estas redes son mucho más flexibles, económicas y fáciles de instalar.

Ventajas de una red inalámbrica

- **Movilidad**

Las redes inalámbricas proporcionan a los usuarios de ésta acceso a la información en tiempo real en cualquier lugar dentro de la organización o el entorno público (zona limitada) en el que están desplegadas.

- **Simplicidad y rapidez en la instalación**

La instalación de una red inalámbrica es rápida y fácil y elimina la necesidad de tirar cables a través de paredes y techos.

- **Flexibilidad en la instalación**

La tecnología inalámbrica permite a la red llegar a puntos de difícil acceso para una red cableada.

- **Costo de propiedad reducido**

Mientras que la inversión inicial requerida para una red inalámbrica puede ser más alta que el costo en hardware de una red estándar, la inversión de toda la instalación y el costo durante el ciclo de vida puede ser significativamente inferior.

- **Beneficios**

A largo plazo son superiores en ambientes dinámicos que requieren acciones y movimientos frecuentes.

- **Escalabilidad**

Los sistemas de red pueden ser configurados en una variedad de topologías para satisfacer las necesidades de las instalaciones y aplicaciones específicas. Las configuraciones son muy fáciles de cambiar y además resulta muy fácil la incorporación de nuevos usuarios a esta.

Desventajas de una red inalámbrica

- **Inseguridad**

Nunca fue más cómodo y fácil compartir Internet *WAN* o disponer de la red local *LAN*, aunque, también es verdad, que nunca fue más fácil y cómodo acceder a redes privadas por no seguir las medidas de seguridad mínimas o simplemente por no ser consciente de ellas.

- **Interferencias**

Se pueden ocasionar por teléfonos inalámbricos que operen a la misma frecuencia, también puede ser por redes inalámbricas cercanas o incluso por otros equipos conectados inalámbricamente a la misma red.

- **Velocidad**

Es menos rápida la transferencia de datos que en una red alámbrica.

Topologías de comunicaciones

Las topologías de comunicaciones son utilizadas para representar la forma de conexión de los dispositivos y el flujo físico de datos. [6]

- **Punto a punto**

Se refiere a una topología en la cual toda la comunicación se produce entre dos puntos, y solo entre estos. El caso mas simple y tal vez el mas común es el de la unión de dos equipos mediante un cable. (Fig. 2.9)

Fig. 2.9 Topología de punto a punto.

- **Punto a multipunto**

En un enlace punto a multipunto, existe un punto central que se comunica con varios puntos remotos. Generalmente esto implica que la comunicación es solamente entre un punto central y otros remotos, y estos hacia el central; no existe comunicación entre los remotos. (Fig. 2.10)

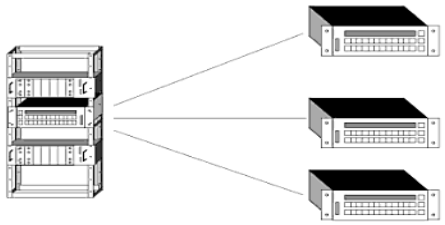

Fig. 2.10 Topología de punto a multipunto.

- **Peer-to-peer (entre pares)**

Si bien este termino involucra arquitectura de comunicaciones y conceptos mas avanzados, cuando se aplica una topología o enlace se refiere a una red cuya estructura física no se encuentra definida, pero cualquiera de sus integrantes puede dialogar directamente con otro. Es posible imaginarla como una red similar a una punto a multipunto en la que se permite la comunicación entre los remotos, como si durante cada comunicación si establecieran pequeños enlaces punto a punto virtuales. (Fig.2.10)

Fig. 2.11 topología de *peer-to-peer.*

Existen algunas otras topologías de comunicación como la de estrella, de árbol, *mesh*, entre otras, pero no se necesitará describirlas dado que no se utilizarán para el proyecto.

2.2.2 Protocolos de comunicación inalámbrica

Existen distintos protocolos para comunicaciones inalámbricas los cuales cuentan con diferentes características dependiendo de las necesidades y de las aplicaciones en la cual serán utilizados.

Zigbee

La ZigBee Alianza [14] es una sociedad abierta, sin fines de lucro de los miembros que han creado un entorno global, próspero, con el desarrollo de normas que, en definitiva ofrecen mayor libertad y flexibilidad para un mundo más inteligente, más sostenible. Los miembros son empresas, universidades y agencias gubernamentales y trabajan juntos desarrollando soluciones inalámbricas para uso en aplicaciones residenciales, comerciales e industriales. La membresía está abierta a todos.

Fig. 2.12 Logotipo de Zigbee Alianza.

ZigBee es muy adecuado para una amplia gama de control utilizado en casi cualquier mercado. La alianza ha centrado sus esfuerzos de desarrollo de normas entorno a los sectores comerciales, residenciales, energía, industriales y de consumo. Se ha desarrollado estándares globales para la automatización de la gestión y la eficiencia energética, el hogar y la construcción, la atención de salud y forma física, telecomunicaciones y electrónica de consumo. Éstos son solo algunos ejemplos de los estándares de control:

- Respuesta de la demanda.
- Infraestructura de medición avanzada.
- Controles de iluminación.
- Control de la calefacción.
- Humo inalámbrico y detectores de CO.

- Seguridad en el hogar.
- Control de persianas, cortinas y las sombras.
- La detección y monitorización clínica.
- Control remoto de sistemas de entretenimiento doméstico.
- Detección de localización en espacios cerrados.
- La publicidad en dispositivos móviles.

ZigBee es la única tecnología inalámbrica basada en estándares diseñados para abordar las necesidades únicas de bajo costo, sensores inalámbricos de baja potencia y redes de control en casi cualquier mercado. ZigBee conecta también la más amplia variedad de dispositivos en fáciles de utilizar en redes, que le da un control sin precedentes de los dispositivos de uso diario en el hogar o en el trabajo.

Características de ZigBee

- Tiene una velocidad de transmisión de 250Kbps y un rango de cobertura de 10 a 75 metros.
- A pesar de coexistir en la misma frecuencia con otro tipo de redes como Wi-Fi o *Bluetooth* su desempeño no se ve afectado, esto debido a su baja tasa de transmisión y, a características propias del estándar IEEE 802.15.4. [6]
- Capacidad de operar en redes de gran densidad, esta característica ayuda a aumentar la confiabilidad de la comunicación, ya que entre más nodos existan dentro de una red, entonces, mayor número de rutas alternas existirán para garantizar que un paquete llegue a su destino.
- Cada red ZigBee tiene un identificador de red único, lo que permita que coexistan varias redes en un mismo canal de comunicación sin ningún problema. Teóricamente pueden existir hasta 16 000 redes diferentes en un mismo canal y cada red puede estar constituida por hasta 65 000 nodos, obviamente estos límites se ven truncados por algunas restricciones físicas (memoria disponible, ancho de banda, etcétera.).
- Es un protocolo de comunicación multi-salto, es decir, que se puede establecer comunicación entre dos nodos aún cuando estos se encuentren fuera del rango de

transmisión, siempre y cuando existan otros nodos intermedios que los interconecten, de esta manera, se incrementa el área de cobertura de la red.

- Su topología de malla (*mesh*) permite a la red auto recuperarse de problemas en la comunicación aumentando su confiabilidad. [14]

Bluetooth

La tecnología inalámbrica *Bluetooth* se lleva a cabo en productos tan diversos como automóviles y aparatos médicos. El costo de la utilización de la tecnología inalámbrica *Bluetooth* se limita solo al costo del producto en el que se integra. No existe un registro de cuenta o servicio relacionado con el uso de la tecnología *Bluetooth*. Esta tecnología inalámbrica funciona con un espectro de radio sin licencia, lo que significa que no hay costo adicional para la comunicación entre dos dispositivos. [18]

Fig. 2.13 Logotipo de *Bluetooth.*

La tecnología inalámbrica *Bluetooth* en sus aplicaciones más comunes tiene un alcance de 30 pies (10m). Esta gama, dependiendo de la clase de dispositivo, se puede extender a 100 metros. La tecnología *Bluetooth* consume una cantidad baja de energía y por tanto es adecuado para los dispositivos móviles y funcionan con pilas. La tecnología ofrece servicios de voz, datos y conexiones de audio entre dispositivos. Se utiliza para aplicaciones como:

- Sensores.
- Juegos y juguetes interactivos.
- Carnets inteligentes.
- Controles remotos.
- Domótica.

Características de Bluetooth

- Soporta tantas conexiones punto a punto como conexiones punto a multipunto.
- Incorpora mecanismos de seguridad.
- No necesita apuntar para transmitir, es capaz de atravesar carteras y paredes.
- Soporta tanto servicios síncronos como asíncronos, lo que facilita la integración con *TCP/IP*.
- Regulada por organismos mundiales.

Wi-Fi

La Wi-Fi Alianza es una organización global sin fines de lucro, asociado de la industria de cientos de empresas líderes dedicadas a la proliferación de tecnología Wi-Fi a través de dispositivos y segmentos de mercado. Con los programas de desarrollo de tecnología, creación de mercado y regulatorios, la Wi-Fi Alianza ha permitido la adopción generalizada de Wi-Fi en todo el mundo. [19]

Fig. 2.14 Logotipo de Wi-Fi Alianza.

La tecnología Wi-Fi ha sido diseñada y optimizada para redes de área local (*LAN*). Los dispositivos suelen tener un alcance de hasta 200 metros y puede cubrir toda una casa con una fuerte señal Wi-Fi. Otro de los beneficios de la tecnología 802.11n es una cobertura mucho mejor.

El ahorro de energía alcanzado depende de la aplicación particular en el uso, así como la eficacia con la aplicación utiliza *WMM Power Save* (función que sirve para ahorrar energía en los dispositivos portátiles que funcionan con baterías como computadoras portátiles, teléfonos, etcétera.). Wi-Fi ofrece mejoras sustanciales en la velocidad, el alcance y la fiabilidad en comparación con otros protocolos de comunicación inalámbrica.

Aplicaciones de WI-FI

- Conexión a internet.
- Transferencia de datos.
- Seguridad.
- Controles remotos.

Características de WI-FI

- Transmiten a frecuencias de 2.4 GHz o 5 GHz. Esta frecuencia es más alta que las utilizadas en teléfonos móviles y *walkie-talkies*. Con una alta frecuencia permite a la señal transportar más datos.
- Usan los estándares de red 802.11 la cual tiene algunas variantes.
- Las radios Wi-fi pueden transmitir en tres bandas de frecuencia, o pueden saltar entre las diferentes bandas cambiando dichas frecuencias. Este cambio de frecuencia impide que se produzcan interferencias y permiten que varios dispositivos *Wireless* usen la misma conexión simultáneamente.

En la tabla 2.2 se muestran los 3 protocolos anteriormente mencionados.

	ZIGBEE	BLUETOOTH	WI-FI
Aplicaciones principales	Monitoreo y control	Remplazo de cable	Web, email, video
Recursos del sistema	4 KB – 32 KB	250KB+	1MB+
Velocidad de datos(KB/S)	20-250	720	11000+
Rango de transmisión(metros)	1-100	1-10	1000+

Tabla 2.2 Características de los protocolos de comunicación inalámbrica.

2.2.3 Estándar IEEE 802.15.4

Las características más importantes del estándar IEEE 802.15.4 [21] son la flexibilidad de la red, bajo costo y bajo consumo de energía; este estándar se puede utilizar para muchas aplicaciones domóticas e industriales, donde se requieren una baja tasa de transmisión de datos.

El estándar soporta múltiples topologías para su conexión en red, entre ellas la topología tipo estrella y la topología *peer – to peer*. La topología tipo estrella establece la comunicación entre los nodos y un controlador central único llamado el coordinador del *PAN* (*Personal Area Network*).

Fundamentos del estándar IEEE 802.15.4

El IEEE 802.15.4 es un protocolo de paquete de datos simple para redes inalámbricas ligeras. Muchos de los aspectos de este diseño han sido usados durante muchos años en redes de radio paquetes. Debido a que ZigBee se concentra en la baja transmisión de datos y es representante de las aplicaciones de baja transmisión de datos, *CSMA* (*Carrier Sense Multiple Access*) está empleado para evitar interferencias. Simplemente, los dispositivos 802.15.4 escuchan antes de transmitir. Si hay una interferencia, el dispositivo espera un período de tiempo y vuelve otra vez o se traslada a otro canal. Hay 16 canales definidos en la banda de 2.4 GHz. El reconocimiento de mensaje está también disponible para la confiabilidad de la entrega de datos mejorada, y están disponibles las estructuras "*beacon*" (guía) para mejorar la latencia. El estándar IEEE 802.15.4 define múltiples niveles de seguridad. El protocolo 802.15.4 está diseñado para la monitorización y para aplicaciones de control donde la duración de la pila es importante.

Características de 802.15.4

Bandas de frecuencia y rango de transmisión de datos	868 MHz: 20 kb/s 915MHz: 40kb/s 2.4GHz: 250kb/s

Alcance	10-20 m
Latencia	Por debajo de 15ms
Canales	868/925 MHz: 11 canales, 2.4GHz: 16 canales
Modos de direccionamiento	Todos los chips tienen 64 bits IEEE de direccionamiento
Canal de acceso	CSMA-CA
Seguridad	128 AES
Red	Hasta 2^{64} dispositivos
Rango de temperatura	-40° a +85° C

Tabla 2.3 Características de 802.15.4.

El estándar 802.15.4 emplea ambos modos de direccionamiento largos y cortos. Los direccionamientos cortos se usan en control de redes donde identifica dotes de red son asignados apropiadamente. Esto resulta en requisitos de memoria reducidos, pero todavía admite hasta 65,000 nodos de red. Como se mencionó antes, hay tres tipos de dispositivos especificados: (*RFD*) como dispositivo defunción reducida, *FFD* como dispositivo de función completa, y el Coordinador de la red. Éstos definen los dispositivos ZigBee, donde un dispositivo "*end point*" puede ser *RFD* o *FFD*, un enrutador es un *FFD*, y un coordinador de ZigBee es el coordinador de la red .802.15.4 emplea una estructura de simple trama de la que se verá con más detalle más adelante. Esta estructura combinada con el reconocimiento de comunicación, resulta una entrega de datos segura. Soporta la asociación/desasociación de la red, así como la encriptación *AES* de 128 bits, si se desea. La estructura *CSMA* permite la buena coexistencia con otros equipos. Hay también disponible una estructura de superframe opcional, para mejorar la latencia.

Tipos de dispositivos IEEE 802.15.4

- **Coordinador de red**

Es el dispositivo más sofisticado. Debe dirigir la red y por lo tanto requiere más memoria.

- **Dispositivo FFD**

Tiene funcionalidad completa. Mientras que un dispositivo *FFD* puede ser un "*end point*", generalmente será un enrutador. El *FFD* también puede trabajar como un puente a otras redes. En este caso, podría requerir más potencia de memoria y computación que el coordinador de la red. Este dispositivo no será alimentado por una pequeña batería, en general.

- **Dispositivo RFD**

Como su nombre implica, tiene un conjunto de características reducidas. Solamente tiene que escuchar/hablar con su coordinador de red y su enrutador más cercano. Esta clase de dispositivos se centra en aplicaciones de dispositivo "*end point*" trabajando con batería. [6]

2.3 Motores de corriente continúa

2.3.1 Introducción

Los motores de corriente directa son maquinas eléctricas que transforman la energía eléctrica en energía mecánica. Impulsan dispositivos tales como ventiladores, bombas, prensas punzadoras y carros. Estos dispositivos pueden tener características de par o momento de torsión-velocidad muy definida (como una bomba o un ventilador) o una extremadamente variable (como un automóvil). A diferencia de los motores paso a paso y los servomecanismos, los motores DC no pueden ser posicionados y/o enclavados en una posición específica. Estos simplemente giran a la máxima velocidad y en el sentido que la alimentación aplicada se los permite. [8]

2.3.2 Partes de un motor DC

El motor de corriente continua está compuesto de dos piezas fundamentales, el rotor y el estator.

Rotor

Constituye la parte móvil del motor, proporciona el torque para mover a la carga. Está formado por:

- **Eje**

Formado por una barra de acero fresada. Imparte la rotación al núcleo, devanado y al colector.

- **Núcleo**

Se localiza sobre el eje. Fabricado con capas laminadas de acero, su función es proporcionar un trayecto magnético entre los polos para que el flujo magnético del devanado circule.

- **Las laminaciones**

Tienen por objeto reducir las corrientes parásitas en el núcleo. El acero del núcleo debe ser capaz de mantener bajas las pérdidas por histéresis. Este núcleo laminado contiene ranuras a lo largo de su superficie para albergar al devanado de la armadura (bobinado).

- **Devanado**

Consta de bobinas aisladas entre sí y entre el núcleo de la armadura. Las bobinas están alojadas en las ranuras, y están conectadas eléctricamente con el colector, el cual debido a su movimiento rotatorio, proporciona un camino de conducción conmutado.

- **Colector**

Denominado también conmutador, está constituido de láminas de material conductor (delgas), separadas entre sí y del centro del eje por un material aislante, para evitar cortocircuito con dichos elementos. El colector se encuentra sobre uno de los extremos del eje del rotor, de modo que gira con éste y se encuentran en contacto con las escobillas. La función del colector es recoger la tensión producida por el devanado inducido, transmitiéndola al circuito por medio de las escobillas (llamadas también cepillos) (Fig.2.15)

Fig. 2.15 Partes de un motor DC.

Estator

Constituye la parte fija de la máquina. Su función es suministrar el flujo magnético que será usado por el bobinado del rotor para realizar su movimiento giratorio. Está formado por:

- **Armazón**

Denominado también yugo, tiene dos funciones primordiales: servir como soporte y proporcionar una trayectoria de retorno al flujo magnético del rotor y del imán permanente, para completar el circuito magnético.

- **Imán permanente**

Compuesto de material ferromagnético altamente remanente, se encuentra fijado al armazón o carcaza del estator. Su función es proporcionar un campo magnético uniforme al devanado del rotor o armadura, de modo que interactúe con el campo formado por el bobinado, y se origine el movimiento del rotor como resultado de la interacción de estos campos.

- **Escobillas**

Están fabricadas se carbón, y poseen una dureza menor que la del colector, para evitar que se desgaste rápidamente. Se encuentran albergadas por los portaescobillas. Ambos, escobillas y portaescobillas, se encuentran en una de las tapas del estator.

2.3.3 Tipos de motores

La característica de un par o momento de torsión-velocidad del motor deber ser adaptada al tipo de carga que tiene que impulsar, y este requerimiento ha dado lugar a tres tipos básicos de motores:

- **Motores en derivación (o Shunt)**

Los devanados de rotor y estator están conectados en paralelo y alimentados por una fuente común. También se denominan máquinas shunt, y en ellas un aumento de la tensión en el inducido hace aumentar la velocidad de la máquina.

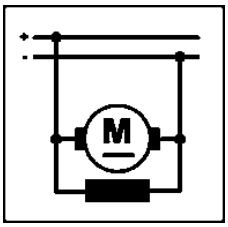

Fig.16 Diagrama de motor en derivación o Shunt.

- **Motores en serie**

Los devanados de inducido y el inductor están colocados en serie y alimentados por una misma fuente de tensión. En este tipo de motores existe dependencia entre el par y la velocidad; son motores en los que, al aumentar la corriente de excitación, se hace disminuir la velocidad, con un aumento del par.

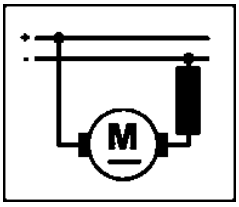

Fig.17 Diagrama de motor en serie.

- **Motores compuestos**

También llamados *compound*, en este caso el devanado de excitación tiene una parte de él en serie con el inducido y otra parte en paralelo. El embobinado en serie con el inducido está constituido por pocas espiras de gran sección, mientras que el otro está formado por un gran número de espiras de pequeña sección. Permite obtener por tanto un motor con las ventajas del motor serie, pero sin sus inconvenientes. Sus curvas características serán intermedias entre las que se obtienen con excitación serie y con excitación en derivación.

Existen dos tipos de excitación compuesta. En la llamada compuesta adicional el sentido de la corriente que recorre los arrollamientos serie y paralelo es el mismo, por lo que sus efectos se suman, a diferencia de la compuesta diferencial, donde el sentido de la corriente que recorre los embobinados tiene sentido contrario y por lo tanto los efectos de ambos devanados se restan.

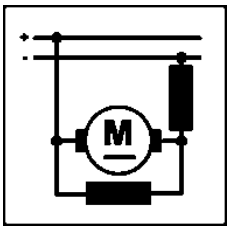

Fig.18 Diagrama de motores compuestos.

2.3.4 Aplicaciones

Aplicaciones y ventajas de los motores de corriente directa (DC)

Aunque el precio de un motor de corriente continua es considerablemente mayor que el de un motor de inducción de igual potencia, existe una tendencia creciente a emplear motores de corriente continua en aplicaciones especiales.

La gran variedad de la velocidad, junto con su fácil control y la gran flexibilidad de las características par velocidad del motor de corriente continua, han hecho que en los últimos años se emplee éste cada vez más con maquinas de velocidad variable en las que se necesite amplio margen de velocidad y control fino de las mismas.

Existe un creciente número de procesos industriales que requieren una exactitud en su control o una gama de velocidades que no se puede conseguir con motores de corriente alterna. El motor de corriente continua mantiene un rendimiento alto en un amplio margen de velocidades, lo que junto con su alta capacidad de sobrecarga lo hace más apropiado que el de corriente alterna para muchas aplicaciones. Los motores de corriente continua empleados en juguetes, suelen ser del tipo de imán permanente, proporcionan potencias desde algunos vatios a cientos de vatios. Los empleados en tocadiscos, unidades lectoras de DC, y muchos discos de almacenamiento magnético son motores en los que el rotor es de imán fijo y sin escobillas. En estos casos el inductor, esta formado por un juego de bobinas fijas, y un circuito electrónico que cambia el sentido de la corriente a cada una de las bobinas para adecuarse al giro del rotor. Este tipo de motores proporciona un buen par de arranque y un eficiente control de la velocidad.

Una última ventaja es la facilidad de inversión de marcha de los motores grandes con cargas de gran inercia, al mismo tiempo que devuelven energía a la línea actuando como generador, lo que ocasiona el frenado y la reducción de velocidad.

Principales aplicaciones del motor de corriente continua

- Trenes de laminación reversibles Los motores deben de soportar una alta carga. Normalmente se utilizan varios motores que se acoplan en grupos de dos o tres.
- Trenes Konti. Son trenes de laminación en caliente con varios bastidores. En cada uno se va reduciendo más la sección y la velocidad es cada vez mayor.
- Cizallas en trenes de laminación en caliente. Se utilizan motores en derivación.
- Industria del papel. Además de una multitud de máquinas que trabajan a velocidad constante y por lo tanto se equipan con motores de corriente continua, existen accionamientos que exigen par constante en un amplio margen de velocidades.
- Otras aplicaciones son las máquinas herramientas, máquinas extractoras, elevadores, ferrocarriles.
- Los motores desmontables para papeleras, trefiladoras, control de tensión en maquinas bobinadoras, velocidad constante de corte en tornos grandes
- El motor de corriente continua se usa en grúas que requieran precisión de movimiento con carga variable (cosa casi imposible de conseguir con motores de corriente alterna).

Capítulo 3

3. Metodología Experimental

En este capítulo se describirá la metodología llevada a cabo para realizar la prueba experimental del proyecto de tesis; se explicará la teoría acerca de las partes fundamentales que conforman el proyecto. Además se muestran experimentos realizados para la familiarización con el hardware requerido.

3.1. Arduino

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos. [15]

Arduino puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores y puede afectar aquello que le rodea controlando luces, motores y otros actuadores. El microcontrolador en la placa se programa mediante el lenguaje de programación Arduino (basado en *Wiring*) y el entorno de desarrollo (basado en *Processing*). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a una computadora, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software.

Las placas pueden ser hechas a mano o compradas montadas de fábrica; el software puede ser descargado de forma gratuita.

Es una plataforma de desarrollo de computación física de código abierto, basada en una placa con un sencillo microcontrolador y un entorno de desarrollo para crear software embebido.

Los proyectos de Arduino pueden ser autónomos o comunicarse con un software que se ejecute en la computadora.

3.1.1 Ventajas de Arduino respecto a otros sistemas

- **Accesible**

Las placas son más accesibles comparadas con otras plataformas de microcontroladores. La versión más costosa de un modulo de Arduino puede ser montada a mano, e incluso ya montada cuesta bastante menos.

- **Multi-Plataforma**

El software funciona en los sistemas operativos *Windows, Macintosh OSX y Linux*. La mayoría de los entornos para microcontroladores están limitados a *Windows*.

- **Entorno de programación simple y directo**

El entorno de programación es fácil de usar para principiantes y lo suficientemente flexible para los usuarios avanzados. Pensando en los profesores, Arduino está basado en el entorno de programación de *Procesing* con lo que el estudiante que aprenda a programar en este entorno se sentirá familiarizado con el entorno de desarrollo Arduino.

- **Software ampliable y de código abierto**

El software esta publicado bajo una licencia libre y está preparado para ser ampliado por programadores experimentados. El lenguaje puede ampliarse a través de librerías de *C++*, y si se está interesado en profundizar en los detalles técnicos, se puede dar el salto a la programación en el lenguaje *AVR C* en el que está basado. De igual modo se puede añadir directamente código en *AVR C* en los programas si así se requiere.

- **Hardware ampliable y de Código abierto**

Arduino está basado en los microcontroladores *ATMEGA168, ATMEGA328 y ATMEGA1280*. Los diseños de los módulos están publicados bajo licencia *Creative Commons*, por lo que diseñadores de circuitos con experiencia pueden hacer su propia versión del módulo, ampliándolo u optimizándolo. Incluso usuarios relativamente inexpertos pueden construir la versión para placa de desarrollo para entender cómo funciona y ahorrar algo de dinero.

3.1.2 Arduino UNO

Arduino UNO (como se muestra en la Fig. 3.1) es una placa electrónica basada en el microcontrolador *ATmega328*. Cuenta con 14 entradas digitales / pines de salida (de los cuales 6 pueden ser utilizados como salidas *PWM*), 6 entradas analógicas, un oscilador de cristal de *16 MHz*, una conexión *USB*, un conector de alimentación, una cabecera de *ICSP* (*In Circuit Serial Programming* método de programación directamente *AVR*), y un botón de reset. Contiene todos los periféricos necesarios para hacer funcionar el microcontrolador, solo tiene que conectarlo a una computadora con un cable *USB* o con un adaptador *AC-DC* o la batería para empezar.

Fig. 3.1 Placa de Arduino UNO.

Características Técnicas de Arduino UNO

- Microcontrolador *ATmega328.*
- Voltaje de Operación 5V.
- Voltaje de Entrada (recomendado) 7-12V.
- Voltaje de Entrada (límites) 6-20V.
- Canales de E / S (de los cuales 6 proporcionar una salida *PWM*).
- Pines de entrada analógica 6.
- Corriente de E / S de CC Pin 40 mA.
- De corriente continua de 3.3V Pin 50 mA.
- Memoria Flash de 32 MB (*ATmega328*) de los cuales 0,5 KB utilizado por gestor de arranque.
- SRAM 2 KB (*ATmega328*).
- EEPROM 1 KB (*ATmega328*).
- Velocidad del reloj de *16 MHz*.

Puede ser alimentado a través de la conexión *USB* o con una fuente de alimentación externa. La fuente de alimentación se selecciona automáticamente.

- Externo (no *USB*), la fuente de alimentación puede venir de un adaptador de *CA* a *CC* o una batería. Los cables de la batería se pueden insertar en los pines *GND* y *VIN* del conector de alimentación.
- La fuente puede operar en un suministro externo de 6 a 20 voltios. Si se suministran con menos de 7V, sin embargo, el adaptador puede suministrar menos de cinco voltios y la placa puede ser inestable. Si utiliza más de 12V, el regulador de voltaje se puede sobrecalentar y dañar la placa. El rango recomendado es de 7 a 12 voltios.

Los pines de alimentación son los siguientes:

- **VIN**

El voltaje de entrada a la placa Arduino cuando se utiliza una fuente de alimentación externa (a diferencia de 5 voltios de la conexión *USB* o de otra fuente de alimentación regulada). Se puede suministrar tensión a través de este pin.

- **5V**

Este pin genera una 5V regulados por el regulador en la tarjeta. La tarjeta puede ser alimentada ya sea desde la entrada de alimentación (7 - 12 V), el conector *USB* (5V), o el pin de *VIN* de la placa (7-12V). El suministro de tensión a través de los pines de 5V o 3.3V no pasa por el regulador, y puede dañar la placa. NO ACONSEJABLE.

- **3.3V**

Un suministro de 3.3 volts generada por el regulador. El consumo de corriente máxima es de 50 mA.

- **GND**

Pin de tierra.

Memoria

El ATmega328 tiene 32 MB (con 0,5 KB utilizados para el gestor de arranque). También dispone de 2 KB de *SRAM* y 1 KB de memoria *EEPROM* (que puede ser leído y escrito con la librería *EEPROM*).

Entrada y salida

Cada uno de los 14 pines digitales en el Arduino UNO se puede utilizar como entrada o salida, usando las funciones *pinMode* (), *digitalWrite* (), y las funciones de *digitalRead* (). Ellos operan a 5 voltios. Cada pin puede proporcionar o recibir un máximo de 40 mA y tiene un arreglo interno de resistencia *pull-up* (desconectado por defecto) de 20 a 50 kΩ. Además, algunos pines tienen funciones especializadas:

- Serie 0 (*RX*) y 1 (*TX*) - Se utiliza para recibir (*RX*) y transmisión (*TX*) datos serie *TTL*. Estos se encuentran conectados a los pines correspondientes de la ATmega8U2 USB a chip de serie *TTL*.
- Interrupciones externas 2 y 3 - Estas patillas se pueden configurar para desencadenar una interrupción en un valor bajo, un borde ascendente o descendente, o un cambio en el valor.
- *PWM* 3, 5, 6, 9, 10 y 11 - Proporcionar 8-bit de salida *PWM* con la función *analogWrite* ().
- LED13 - Hay un led conectado al pin digital 13, el cual funciona como un indicador de propósito general. Cuando el pin es de alto valor, el led está encendido, cuando el pasador es bajo, es apagado.

El Arduino UNO tiene seis entradas analógicas, etiquetados A0 a A5, cada una de las cuales proporcionan 10 bits de resolución (es decir 1024 valores diferentes). Por defecto miden desde el 0 a 5 voltios, aunque es posible cambiar el extremo superior de su rango con el pin y el *AREF* y la función *analogReference* ().

Comunicación

El Arduino tiene una serie de facilidades para comunicarse con una computadora, otro Arduino, u otros microcontroladores. El *ATmega328* ofrece una *UART TTL* (5V) de comunicación en serie, que está disponible en los pines digitales 0 (*RX*) y 1 (*TX*). Un *ATmega16U2* en los canales de comunicación a través de *USB* y aparece como un puerto *COM* virtual con el software en la computadora. El *firmware 16U2* utiliza el estándar de los controladores *USB*, *COM*, y no es necesario ningún controlador externo. Sin embargo, en *Windows*, un archivo. "Inf" es requerido. El software de Arduino incluye un monitor de datos

(Data Monitor) que permite el monitoreo de datos que se envía desde y hacia la placa. Los *LEDs RX* y *TX* en la tarjeta parpadean cuando se están transmitiendo datos a través del puerto *USB* a serie y la conexión *USB* a la computadora (pero no para la comunicación de serie en los pines 0 y 1).

La biblioteca *SoftwareSerial* permite la comunicación de serie en cualquiera de los pines digitales de Arduino UNO.

El *ATmega328* también soporta comunicación *I2C* (bus de comunicaciones en serie) y *SPI*. El software incluye una librería *Wire* para simplificar el uso del bus *I2C*. Para la comunicación *SPI*, utilizar la biblioteca de *SPI*.

Programación

La placa puede ser programada bajo el entorno de programación Arduino. El *ATmega328* viene pre-quemado con un gestor de arranque que le permite cargar nuevo código a la placa sin el uso de un programador de hardware externo.

También puede pasar por alto el gestor de arranque y el programa del microcontrolador a través de la *ICSP* (programación *In-Circuit Serial*). El *ATmega16U2* (o *8U2* en los *REV1* y *REV2*) el código fuente está disponible en el *firmware*.

Reset

En lugar de requerir presionar el botón de reset antes de una carga, el Arduino está diseñado de una manera que le permite ser restaurado mediante el software que se ejecuta en una computadora conectada. Una de las líneas de control de flujo de hardware (*DTR*) de la *ATmega8U2/16U2* está conectado a la línea de reposición del *ATmega328* través de un condensador 100 *nanofaradios*. Cuando ésta toma un valor bajo, el voltaje suministrado de la línea de reset cae lo suficiente como para restablecer el chip. El software de Arduino utiliza esta capacidad que le permite cargar el código con sólo pulsar el botón de carga en el entorno Arduino.

USB Protección contra sobrecorriente

Arduino UNO tiene un fusible reajustable que protege a los puertos *USB* de la computadora de pequeños cortos y de sobrecorriente. Aunque la mayoría de las computadoras ofrecen su

protección interna, el fusible proporciona una capa adicional de protección. Si hay más de 500 mA aplicados al puerto *USB*, el fusible automáticamente corta la conexión hasta que el cortocircuito o una sobrecarga se han eliminado.

Dimensiones físicas de Arduino UNO

Las dimensiones externas de la placa Arduino UNO son 70x50 milímetros (las unidades mostradas en la Fig. 3.2 se encuentran en milésimas de pulgada).

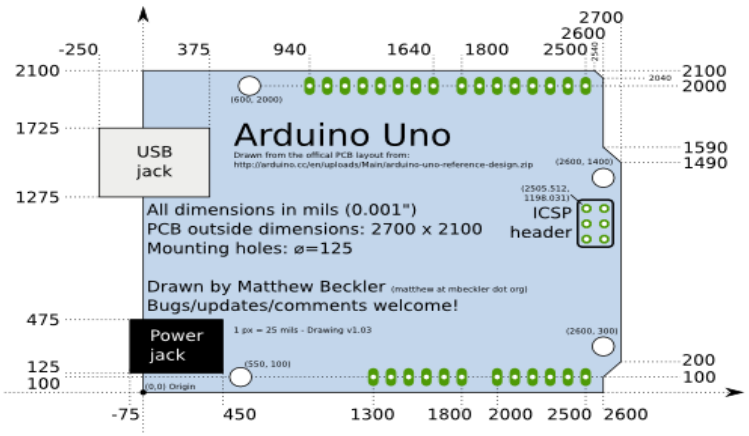

Fig. 3.2 Dimensiones placa Arduino UNO.

3.1.3 Experimentos

El propósito de esta sección es familiarizarse con el entorno de hardware y software Arduino y conocer sus herramientas de programación. Se describen solo algunos de los experimentos realizados para ilustrar la metodología empleada.

Experimento 1.- Intermitente

Encender y a pagar un led que se conecta en el pin 13 de Arduino y se configura como salida. El tiempo de encendido y apagado es de 1 segundo.

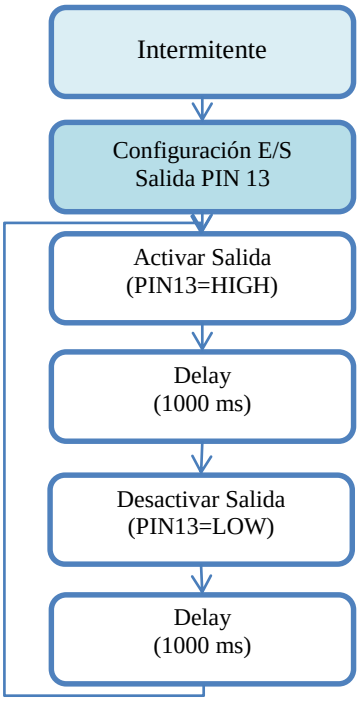

Fig. 3.3 Diagrama pseudocódigo de experimento 1.

Conexión física en Arduino

Solo se requiere de un led para la salida, para el Arduino UNO es opcional ya que cuenta con un led incorporado en el pin 13.

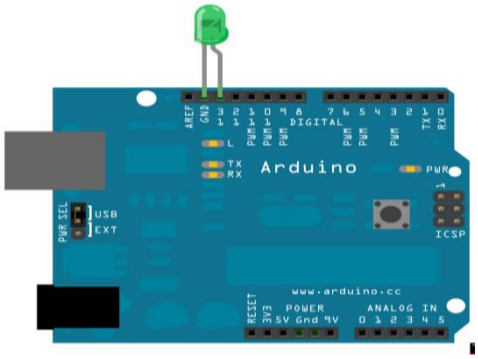

Fig. 3.4 Conexión de componentes para experimento 1.

Experimento 2.- Lectura de un pulsador

Encender y a pagar un led conectado en el pin 13 de Arduino y se configura como salida, un botón pulsador en el pin 10 y se configura como entrada. El nivel del Led será controlado mediante un botón pulsador conectado al pin 10.

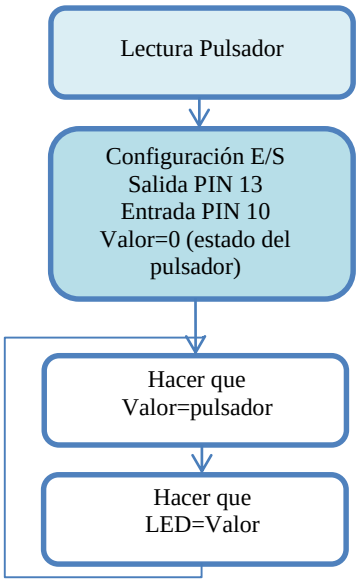

Fig. 3.3 Diagrama pseudocódigo de practica 2.

Conexión física de Arduino

Solo se requiere de un led para la salida, para el Arduino UNO es opcional ya que cuenta con un led incorporado en el pin 13. Además de un botón pulsador conectado al pin 10.

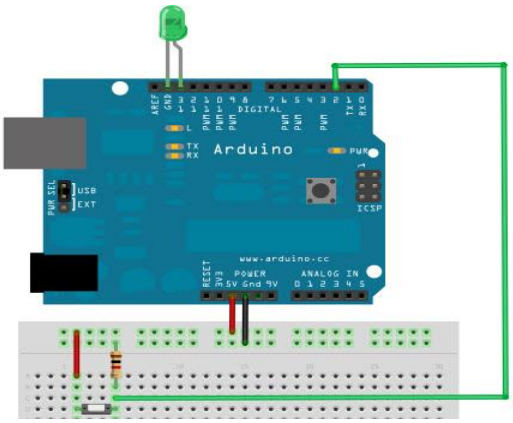

Fig. 3.4 Conexión de componentes para práctica 2.

3.1.4 Conclusiones

Utilizar la plataforma de Arduino es sencillo inclusive para personas que no conocen mucho acerca del tema; tiene un entorno de programación fácil de utilizar y bastante completo, además de un lenguaje sencillo. Se pueden realizar muchas de pruebas y practicas de todo tipo. La placa de Arduino UNO es una de las más completas y fáciles de utilizar debido a su conexión mediante *USB* a la computadora. Y las facilidades que da para salidas digitales, analógicas así como salidas *PWM*.

3.1 Zigbee

En el presente tópico se estudiará lo relacionado al protocolo de comunicaciones inalámbricas utilizado por *Zigbee.* Las principales características de los módulos *XBee* y del *Shield* que utiliza Arduino para comunicaciones inalámbricas.

3.2.1 Introducción a Zigbee

Zigbee es un protocolo de comunicaciones inalámbrico basado en el estándar de comunicaciones para redes inalámbricas IEEE 802.15.4. Creado por *Zigbee Alliance*, una organización, en principio sin fines de lucro, de más de 200 grandes empresas. Es especialmente útil para redes de sensores en entornos industriales, médicos y, sobre todo, domótica. [22]

Las comunicaciones *Zigbee* se realizan en la banda libre de 2.4GHz. A diferencia de *bluetooth*, este protocolo no utiliza *FHSS* (*Frequency hooping*), sino que realiza las comunicaciones a través de una única frecuencia, es decir, de un canal. Normalmente puede escogerse un canal de entre 16 posibles. El alcance depende de la potencia de transmisión del dispositivo así como también del tipo de antenas utilizadas. La velocidad de transmisión de datos de una red *Zigbee* es de hasta 256kbps. Una red *Zigbee* la pueden formar, teóricamente, hasta 65535 equipos, es decir, el protocolo está preparado para poder controlar en la misma red esta cantidad enorme de dispositivos.

Entre las necesidades que satisface el módulo se encuentran:

- Bajo costo.
- Ultra-bajo consumo de potencia.
- Uso de bandas de radio libres y sin necesidad de licencias.
- Instalación barata y simple.
- Redes flexibles y extensibles.

El uso del protocolo Zigbee va desde remplazar un cable por una comunicación serial inalámbrica, hasta el desarrollo de configuraciones punto a punto, multipunto, *peer-to-peer* (todos los nodos conectados entre sí) o redes complejas de sensores.

Una red Zigbee la forman básicamente tres tipos de elementos. Un único dispositivo Coordinador, dispositivos Routers y dispositivos Finales *(End Points* o *End Device*). [22]

- **El Coordinador**

Es el nodo de la red que tiene la única función de formar una red. Es el responsable de establecer el canal de comunicaciones y del *PAN ID* (identificador de red) para toda la red. Una vez establecidos estos parámetros, el Coordinador puede formar una red, permitiendo unirse a él a dispositivos routers y end points. Una vez formada la red, el Coordinador hace las funciones de router, esto es, participar en el enrutado de paquetes y ser origen y/o destinatario de información

.

- **El Router**

Es un nodo que crea y mantiene información sobre la red para determinar la mejor ruta para transmitir un paquete de información. Lógicamente un router debe unirse a una red Zigbee antes de poder actuar como Router retransmitiendo paquetes de otros routers o de end points.

- **End Device**

Son dispositivos finales no tienen capacidad de enrutar paquetes. Deben interactuar siempre a través de su nodo padre, ya sea este un Coordinador o un Router, es decir, no puede enviar información directamente a otro *end device*. Normalmente estos equipos van alimentados a baterías. El consumo es menor al no tener que realizar funciones de enrutamiento.

Cada módulo Zigbee, al igual que ocurre con las direcciones *MAC* de los dispositivos ethernet, tiene una dirección única. En el caso de los módulos Zigbee cada uno de ellos tiene una dirección única de 64 bits que viene grabada de fábrica. Por otro lado, la red Zigbee, utiliza para sus algoritmos de ruteo direcciones de 16 bits. Cada vez que un dispositivo se asocia a una red Zigbee, el Coordinador al cual se asocia le asigna una dirección única en toda la red de 16 bits. Por eso el número máximo teórico de elementos que puede haber en una red Zigbee es de 2^16 =65535, que es el número máximo de direcciones de red que se pueden asignar.

Estos módulos *XBee*, pueden ser ajustados para usarse en redes de configuración punto a-punto, punto-a-multipunto o *peer-to-peer*.

3.2.2 Modulo Xbee S2

El *XBee-XB24* del módulo de Digi serie 2 mejora la potencia de salida y el protocolo de datos respecto a versiones anteriores. Los módulos Serie 2 le permiten crear complejas redes de mallas basadas en el firmware de *ZigBee XBee ZB* malla. Estos módulos permiten una comunicación muy simple y confiable entre microcontroladores, computadoras y sistemas, realmente cualquier dispositivo con un puerto serie. Punto a punto y multipunto las redes son compatibles.

Los módulos S2 son esencialmente el mismo hardware que la serie anterior 2.5, pero con el firmware actualizado. Son compatibles con la serie 2.5 si el firmware de estos se actualiza a través del software *X-CTU*.

Serie 1 y Serie 2 módulos XBee tienen el mismo pin-out. Sin embargo, en la serie 1 los módulos no pueden comunicarse con los módulos serie (el cual es mostrado en la Fig. 3.5).

Fig. 3.5 Modulo Xbee S2 de DIGI.

Características Técnicas:

• 3.3V @ 40mA.

• 250kbps máxima velocidad de datos.

• Salida de 2mW (+3 dBm).

• 400 pies (120m) Alcance.

• Antena integrada.

• Totalmente certificado por la FCC.

• 6 de 10-bit ADC pines de entrada.

• 8 pines IO digitales.

• 128-bit de encriptación.

• Configuración local o por aire.

• AT o conjunto de API de comandos.

Circuito básico para el XBee

En la siguiente figura (Fig. 3.6) se muestran las conexiones mínimas que necesita el módulo Xbee para poder ser utilizado. Luego de esto, se debe configurar según el modo de operación que se desea para la aplicación requerida.

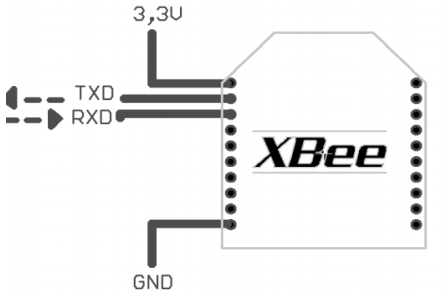

Fig. 3.6 Conexiones mínimas para XBee.

El módulo requiere una alimentación desde 2.8 a 3.4 V, la conexión a tierra y las líneas de transmisión de datos por medio del *UART* (*TXD* y *RXD*) para comunicarse con un microcontrolador, o directamente a un puerto serial utilizando algún conversor adecuado para los niveles de voltaje.

Esta configuración, no permite el uso de Control de Flujo (*RTS & CTS*), por lo que ésta opción debe estar desactivada en el terminal y en el módulo XBee. En caso de que se envíe una gran cantidad de información, el buffer del módulo se puede saturar. Para evitarlo existen dos alternativas:

- Bajar la tasa de transmisión.
- Activar el control de flujo.

Especificaciones

En la tabla 3.1 se muestran las especificaciones de modulo XBee S2:

Especificaciones	XBee S2
Rendimiento	
Interior / Urbana Rango	Hasta 300 pies (90 m), hasta 200 pies (60 m) internacional de la variante
Transmisión de potencia de salida	50mW (+17 dBm) 10mW (+10 dBm)
Velocidad de datos	250,000 bps
Rendimiento de datos	35000 bps
Velocidad de transferencia de datos	1200 bps - 1 Mbps
Sensibilidad de datos	-102 dBm
Requisitos de energía	
Voltaje	3.0 - 3.4 V
Corriente de operación(transmisión, máxima potencia de salida)	295mA (3.3 V) 170mA (3.3 V)
Corriente de operación (recibida)	45 mA (3.3 V)
Potencia baja	3.5 microA 25°C

General	
Banda de frecuencia de operación	2.4 GHz
Dimensiones	2.438cm x 3.294cm
Temperatura de operación	- 40 a 85º C (industrial)
Opciones de la antena	Whip, Chip, RP SMA, ó UFL
Redes y seguridad	
Topologías de red	Punto a punto, punto a multipunto, *mesh*
Numero de canales	14 canales de secuencia directa
Canales	11 a 24
Opciones de direccionamiento	PAN ID y direcciones, cluster IDs y end points(opcional)

Tabla 3.1 Especificaciones de Modulo XBee S2.

Dimensiones físicas

En la figura 3.7 se muestran las dimensiones físicas de los módulos XBee:

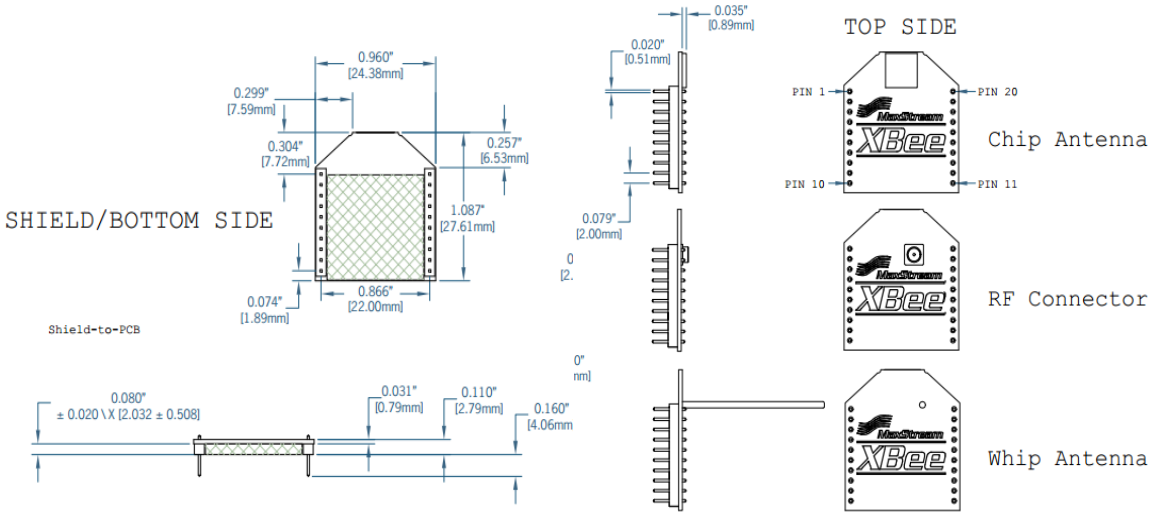

Fig. 3.7 Dimensiones de módulos OEM RF XBee.

Pines del modulo XBee

En la tabla 3.2 se muestra la función de los pines de módulo XBee S2

Numero de pin	Nombre	Dirección	Estado por default	Descripción
1	VCC	.	.	Fuente de voltaje
2	DOUT	Output	Output	UART de datos de salida
3	DIN/CONFIG	Input	Input	UART datos de entrada
4	DIO12	Both	Disabled	Digital I/O 12
5	RESET	Both	colector abierto con pull-up	Reset (el pulso del botón reset debe ser al menos 200ns)
6	RSSI PWM / DIO10	Both	Output	RX Intensidad de la señal de indicación / Digital IO
7	DIO11	Both	Input	Digital I/O 11
8	[Reservado]	Both	Disabled	Sin conectar
9	DTR / SLEEP_RQ/ DIO8	Both	Input	SLEEP pin o Digital IO 8
10	GND	-	-	GROUND
11	DIO4	Both	Input	Digital I/O 4
12	CTS / DIO7	Both	Output	Enviar el control de flujo o de E / S digital 7. CTS, si esta activado, es una salida.
13	ON / SLEEP	Output	Output	Módulo indicador de estado o Digital I / O 9
14	VREF	Input		No se utiliza para EM250. Se utiliza para programar un procesador secundario. Para mantener la compatibilidad con los módulos XBee de otro modo, Recomendamos conectar este pin de tensión de referencia si el muestreo analógico que se desea. También se puede conectar a GND.
15	DIO5	Both	Output	Indicador, Digital I/O 5
16	RTS / DIO6	Both	Input	Solicitud para Enviar el control de flujo, Digital I / O 6. Estrategia en tiempo real, si está habilitado, es una entrada.
17	AD3 / DIO3	Both	Disabled	Entrada analógica 3 o Digital I/O 3
18	AD2 / DIO2	Both	Disabled	Entrada analógica 2 o Digital I/O 2
19	AD1 / DIO1	Both	Disabled	Entrada analógica 1 o Digital I/O 1

20	AD0 / DIO0	Both	Disabled	Entrada analógica 0, Digital IO 0, o puesta en marcha botón

Tabla 3.2 Pines de modulo Xbee S2.

3.2.3 Arduino XBee Shield

El XBee shield para Arduino permite comunicar al entorno Arduino de forma inalámbrica usando ZigBee. Fue desarrollado en colaboración con *Libelium*. [15] XBee Shield está basada en el módulo XBee de *MaxStream*. [16] El módulo puede comunicarse hasta 100ft (30 metros) en interior o 300ft (90 metros) al aire libre (en visión directa). Puede ser usado como remplazo del puerto serie/*USB* o puede ser usado en modo de comandos y configurarlo para una variedad de opciones de redes broadcast o malladas. También provee conectores hembra para usar los pines digitales desde 2 hasta 7 y las entradas analógicas, las cuales están cubiertas por la shield (los pines digitales de 8 a 13 no están cubiertos por la placa, así que es posible usar los conectores de la placa directamente).

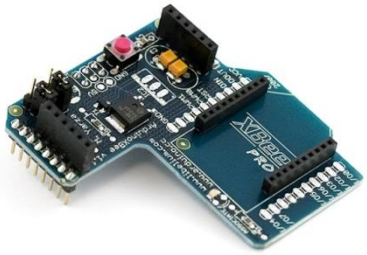

Fig. 3.8 Arduino XBee Shield.

Configuración de los *jumpers*

El XBee Shield tiene dos *jumpers* que están sobre los tres pines etiquetados como XBee/*USB* (ver Fig. 3.9). Estos determinan cómo se conecta la comunicación serie del XBee entre el microcontrolador (*Atmega8* o *ATmega168*) y el chip serie *FTDI* de la placa Arduino.

Con los *jumpers* en la posición XBee (en los dos pines más cercanos al interior de la placa), el pin *DOUT* del módulo Xbee está conectado al pin *RX* del microcontrolador; y el pin *DIN* está conectado a *TX*. Los pines *RX* y *TX* del microcontrolador están todavía conectados a los pines *TX* y *RX* (respectivamente) del chip *FTDI*; los datos enviados desde el microcontrolador serán transmitidos a la computadora vía *USB* y a la vez enviados de forma inalámbrica por el módulo XBee. El microcontrolador, sin embargo, solo será capaz de recibir datos desde el módulo XBee, no desde el *USB* a la computadora.

Con los *jumpers* en la posición *USB* (en los dos pines más cercanos al borde de la placa), el pin *DOUT* del módulo XBee está conectado al pin *RX* del pin del chip *FTDI*, y el *DIN* del módulo XBee está conectado al pin *TX* del el chip *FTDI*. Esto significa que el módulo XBee puede comunicarse directamente con la computadora; sin embargo, esto solo funciona si el microcontrolador se ha retirado de la placa Arduino. Si el microcontrolador se deja en la placa, solo será capaz de comunicarse con la computadora vía *USB*, pero ni la computadora ni el microcontrolador podrán comunicarse con el módulo XBee.

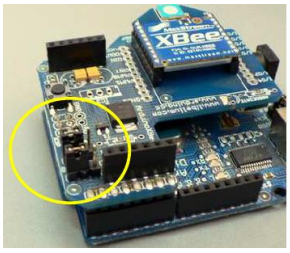

Fig. 3.9 Ubicación de los *jumpers* en XBee Shield.

Direccionamiento para módulos XBee 802.15.4

Hay múltiples parámetros que necesitan ser configurados correctamente para que dos módulos puedan comunicarse entre ellos (con configuración por defecto, todos los módulos deberían ser capaces de comunicarse unos con otros). Necesitan estar en la misma red, definida por el parámetro ID. Los módulos necesitan estar en el mismo canal, definido por el parámetro CH (canal de direccionamiento). Finalmente, la dirección de destino de un módulo (parámetros DH y DL) qué determina que módulo en esa red y canal recibirá los datos transmitidos. Esto puede suceder de las siguientes formas:

- Si el DH de un módulo es 0 y su DL es menor de 0xFFFF (16 bits), los datos transmitidos por ese módulo serán recibidos por cualquier módulo cuyos 16 bits de dirección del parámetro MY sea igual al DL.
- Si el DH es 0 y el DL es igual a 0xFFFF, las transmisiones del módulo serán recibidas por todos los módulos.
- Si el DH no es cero o el DL es mayor de 0xFFFF, la transmisión solo será recibida por el módulo cuyo número de serie sea igual a la dirección de destino del módulo transmisor (cuyos SH es igual al DH del módulo transmisor y cuyo SL sea igual a su DL).

Se pueden direccionar los módulos Xbee como se muestra anteriormente, aunque no es necesario ya que los módulos vienen con una configuración por defecto con la cual se comunican entre si sin la necesidad de configurar todos los parámetros. Solo se necesita configurarle que tipo de dispositivo será (coordinador, *router*, dispositivo final).

Parámetros de configuración

- ID.- Numero ID de la conexión a configurar.
- MY.- Dirección del módulo Xbee.
- CH.- Canal de transferencia.
- DL.- Dirección destino bajo.
- DH.- Dirección destino alta.
- SL.- Numero serial bajo.
- SH.- Numero serial alto.

La correspondencia de direcciones solo sucederá entre módulos en la misma red y canal. Si dos módulos están en diferentes redes o canales, no podrán comunicarse sea cual sea sus direcciones.

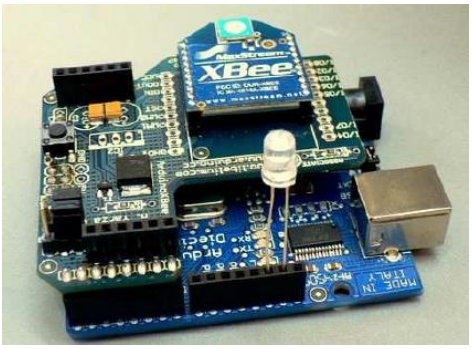

Fig. 3.10 XBee Shield montados en Arduino.

Configurando el módulo XBee

Se puede configurar el módulo XBee mediante el código que ejecuta el Arduino o desde un software en la computadora (como por ejemplo *X-CTU* que se verá mas adelante). Para configurarlo desde el Arduino, es necesario colocar los jumpers en la posición XBee. Para configurarlo desde la computadora, se requiere colocar los *jumpers* en la posición *USB* y haber quitado el microcontrolador del Arduino.

Para entrar en el modo de configuración, se envían tres signos '+': +++ y no enviar ningún otro carácter al módulo durante un segundo antes y un segundo después. Esto incluye caracteres especiales como retorno de carro o nueva línea. Por lo tanto, se está configurando el modulo desde la computadora, hay que asegurarse de que la terminal está configurado para enviar los caracteres tal como son escritos, sin esperar a pulsar "enter". De lo contrario, enviará los tres signos '+' seguidos de nueva línea (de modo que no esperará un segundo después de enviar +++). Si se consigue entrar en el modo de configuración, el módulo enviará de vuelta 'OK', seguido de retorno de carro.

Envía comando	Espera respuesta
+++	OK<CR>

Una vez en el modo de configuración, se envían comandos AT al módulo. Las cadenas de comandos AT tienen la forma ATXX (donde XX es el nombre del parámetro). Para leer el valor actual de un parámetro, se envía la cadena de comando AT seguido de retorno de carro (<CR>). Para escribir un nuevo valor del parámetro, se envía la cadena de comando AT, seguido del nuevo valor (sin espacios o nueva línea por medio), seguido de retorno de carro. Por ejemplo, para leer el identificador (ID) de red del módulo (que determina con qué otros módulos XBee se comunicará), se usa el comando ATID:

Envía comando	Espera respuesta
ATID<enter>	3332<CR>

Para cambiar el valor del ID de red del módulo:

Envía comando	Espera respuesta
ATID3331<enter>	OK<CR>

Ahora, se comprueba que el parámetro se ha configurado:

Envía comando	Espera respuesta
ATID<enter>	3331<CR>

A no ser que se le configure al módulo que escriba los cambios a la memoria no volátil, esos cambios sólo tendrán efecto hasta que el módulo pierda la alimentación. Para guardar los cambios de forma permanente (hasta que se modifiquen de nuevo) usa el comando ATWR:

Envía comando	Espera respuesta
ATWR<enter>	OK<CR>

Para resetear el módulo a los valores de fábrica, usa el comando ATRE:

Envía comando Espera respuesta

ATRE<enter> OK<CR>

El reset no será permanente a no ser que después se envíe el comando ATWR.

3.2.4 Experimentos con XBee

Los experimentos desarrollados con la Arduino XBee Shield montada sobre la placa Arduino UNO son las mismas que las mostradas anteriormente; solo que para este caso es necesario contar con dos módulos de XBee, XBee Shield y Arduino UNO, para que uno tome la función de Coordinador y el otro de *End device* (dispositivo final). El programa debe de separarse de igual manera en dos partes una emisora y otra receptora para cada Arduino. En el Emisor se conectan las entradas del sistema y en el receptor las salidas o actuadores. A continuación se muestra un ejemplo sencillo para una comunicación entre los Arduinos mediante XBee con XBee shield.

Experimento 1.- Ejemplo rápido

Para cargar un programa a la placa Arduino con XBee shield, es necesario poner los dos *jumpers* del shield en la posición *USB* (se colocan los dos *jumpers* en la parte cercana al borde de la placa) o se quitan completamente. Luego, se carga un programa desde el IDE. En este caso, se carga el programa *Communication* | *Physical Pixel* (programa de ejemplo del software) a una de las placas. Este programa manda a la placa encender el led conectado al pin 13 cuando recibe 'H' por el puerto serie y lo apaga cuando recibe 'L'. Puede ser probado conectando la placa con el monitor de puerto serie de Arduino (configurado a 9600 baudios), escribiendo H y enter (o pulsando en "*send*") el led encenderá. Envía L y el led se apagará.

Una vez que cargado el programa *Physical Pixel* y comprobado que funciona, se desconecta el primer Arduino de la computadora. Se colocan los *jumpers* en la posición XBee (en la posición más alejada del borde de la placa). Ahora, es necesario cargar otro programa a la otra placa. Los *jumpers* deben estar en la posición *USB*. Luego se carga el siguiente programa a la placa:

```
void setup() {
     Serial.begin(9600);
}
void loop(){
```

```
      Serial.print('H');
      delay(1000);
      Serial.print('L');
      delay(1000);
}
```

Cuando se haya cargado el programa, se puede comprobar que funciona con el monitor de puerto serie. Se debe de observar H's y L's llegando cada segundo. Se apaga el monitor de puerto serie y se desconecta la placa. Se necesitan cambia los *jumpers* a la posición XBee. Ahora se conectan las dos placas a la computadora, después de unos segundos, se observará el led de la primera placa encenderse y apagarse cada segundo (el led de la placa Arduino, no el led de la placa XBee Shield, que proporciona información sobre el estado del módulo XBee).

Experimento 2.- Lectura de un botón

Encender y a pagar un led conectado en el pin 13 de Arduino y se configura como salida, un botón pulsador en el pin 10 y se configura como entrada. El nivel del led será controlado mediante un botón pulsador conectado al pin 10.

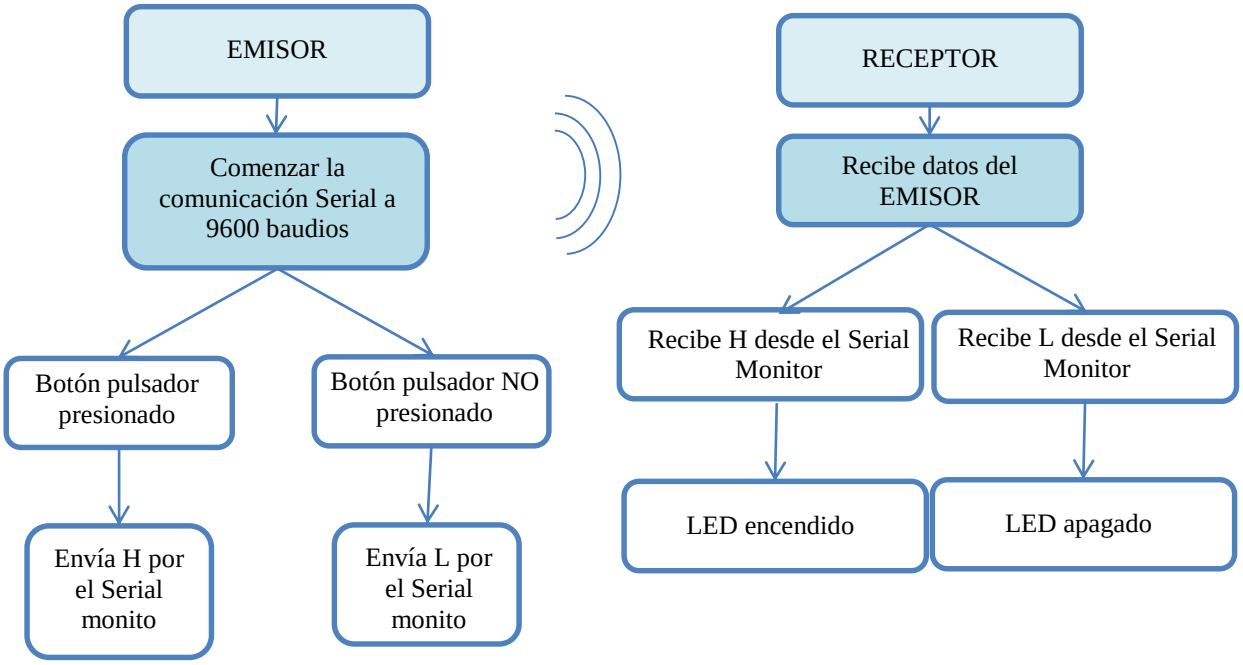

Fig. 3.11 Diagrama pseudocódigo de experimento 2.

3.2.5 Conclusiones

El módulo XBee Shield facilita una conexión entre los dispositivos XBee de comunicación inalámbrica montados en las placas de Arduino UNO, para poder controlar dispositivos, adquirir datos y otras funciones sin necesidad de tener conectado mediante un cable el controlador al dispositivo final. El XBee Shield es relativamente sencillo de utilizar; solo se tienen que configurar los módulos XBee y montarlos sobre cada una de las placas Arduino, y también separar el programa en dos partes, que son la emisora y la receptora.

3.3 Control de motores DC

En este tema se introduce a las estrategias para controlar los motores DC; además se muestran algunos experimentos de prueba para familiarizarse con la programación desde la plataforma de microcontroladores usada.

3.3.1 Fundamentos teóricos

Sentido de giro

El sentido de giro de un motor de corriente directa (o corriente continua) depende del sentido relativo de las corrientes circulantes por los devanados inductor e inducido. La inversión del sentido de giro del motor de corriente directa se consigue invirtiendo el sentido del campo magnético o de la corriente del inducido. Si se permuta la polaridad en ambos bobinados, el eje del motor gira en el mismo sentido.

Control de velocidad por ancho de pulso

La regulación por ancho de pulso, *PWM* por sus siglas en inglés, de un motor de DC está basada en el hecho de que si se recorta la corriente continua de alimentación en forma de una onda cuadrada, la energía que recibe el motor disminuirá de manera proporcional a la relación entre la parte alta (habilita corriente) y baja (cero corriente) del ciclo de la onda cuadrada. Controlando esta relación se logra variar la velocidad del motor de una manera bastante

aceptable (Fig. 3.12). Otra estrategia de controlar la velocidad de un motor DC es disminuyendo el voltaje suministrado para su funcionamiento, pero al hacer esto también se ve afectada la potencia del motor. [7]

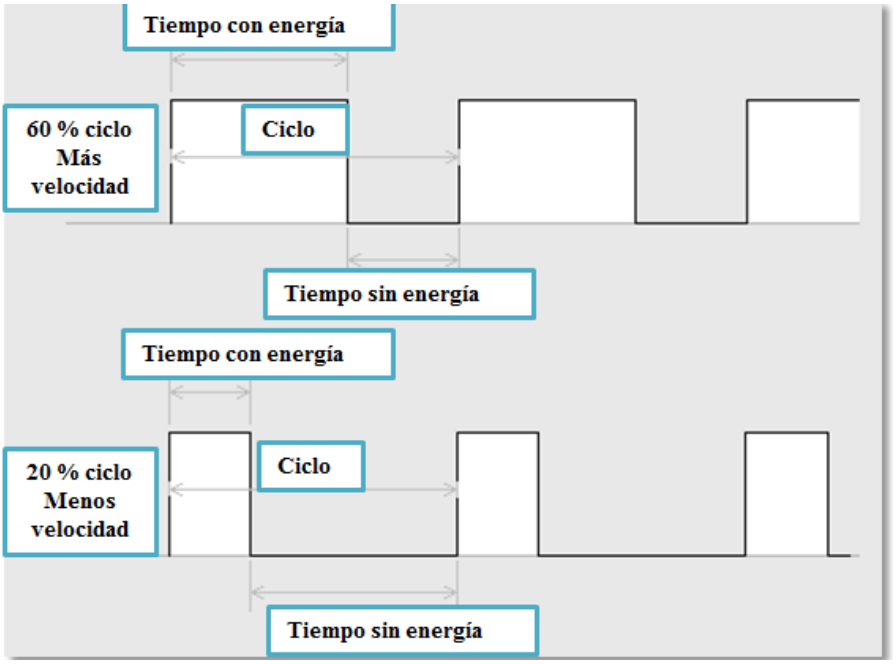

Fig.3.12 Control de velocidad por ancho de pulso (*PWM*).

3.3.2 Driver LMD18200

LMD18200 Puente H

El *LMD18200* es un Puente H diseñado para el control de movimiento. [23] El dispositivo está construido mediante un proceso multi-tecnología que combina bipolar y *CMOS* como circuitos de control. Es perfecto para la conducción de motores de corriente continua y paso a paso, se adapta a los *LMD18200* corrientes de pico de salida de hasta 6A. Facilita la detección de pequeñas pérdidas. (La figura 3.13 muestra el encapsulado del *LMD18200).*

Características Técnicas

- Entrega hasta la salida continua 3A.
- Funciona con tensiones de alimentación de hasta 55V.

- De bajo *RDS* (encendido) normalmente 0.33Ohms por interruptor en 3A.
- *TTL* y *CMOS* de insumos compatibles.
- Alerta térmica de salida a 145 ° C.
- Térmico apagado a 170 ° C.
- Protección interna de diodos.
- Protección de la carga en cortocircuito.
- Driver Interior de carga con capacidad de arranque externos.
- Frecuencia de funcionamiento de hasta 500KHz.
- Incluye diodos de recuperación rápida (70nseg los de la parte alta del puente y 100nseg los de la parte baja) en paralelo con cada transistor MOS.
- Dispone de transistores *MOSFETs* en la parte alta de la diagonal del puente para el sensado de la corriente que circula por el motor.

Aplicaciones

- Motores DC y motores de motor paso a paso.
- Posición y servomecanismos la velocidad.
- Automatización de robots.
- Máquinas de control numérico.
- Impresoras y plotters.

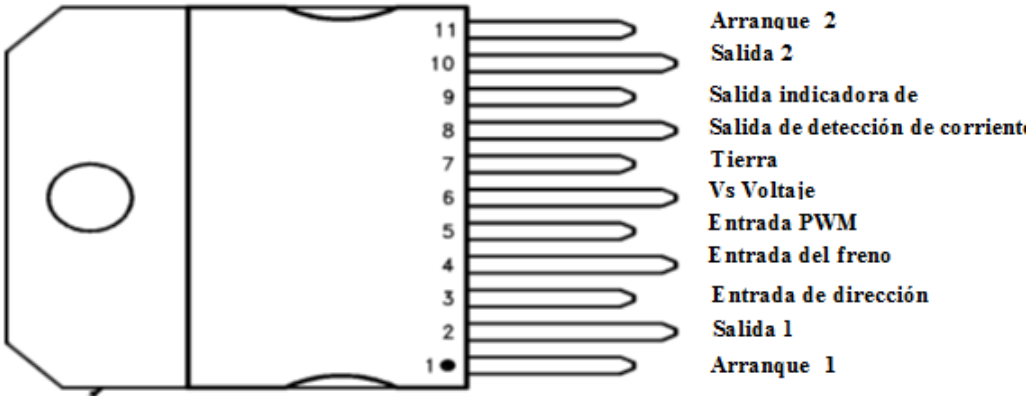

Fig. 3.13 pines del *LMD18200.*

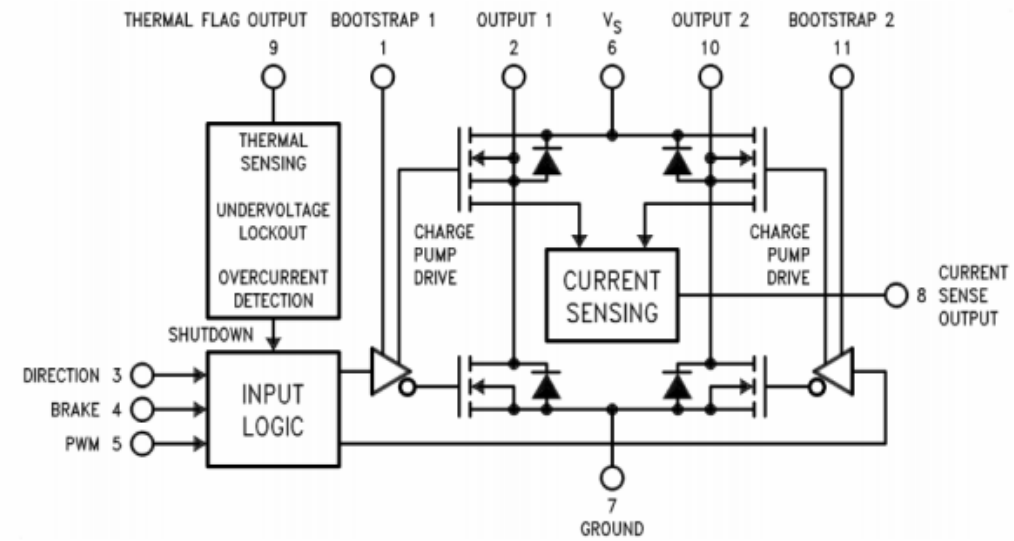

Fig.3.14 Diagrama del circuito *LMD18200.*

Transistores DMOS

Los transistores *DMOS* permiten que la corriente fluya de forma bidireccional, a la vez que tienen una baja caída de tensión cuando se encuentran en estado de saturación, debido a su baja resistencia *RDSON*. Además, dichos transistores llevan asociado un diodo intrínseco en paralelo, lo que evita tener que ponerlos externamente para proteger a los transistores en los intervalos en los que existe corriente de recirculación

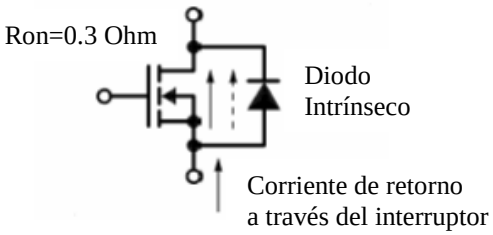

Fig.3.15 Transistores *DMOS.*

Cuando se utilizan transistores *DMOS,* la corriente de recirculación se comparte entre el diodo intrínseco y el transistor, debido a que los transistores *DMOS* son capaces de conducir corriente en ambas direcciones como se indica anteriormente (Fig. 3.15). Para corrientes inferiores a 2.5 A la caída de tensión en el transistor *DMOS* es menor que la caída de tensión en directo del diodo intrínseco que lleva asociado, y toda la corriente circula por el transistor.

Para corrientes superiores a los valores indicados anteriormente, el diodo entra en conducción y la corriente total se comparte entre ambos, transistor y diodo.

La tabla 3.3 nos muestra tabla del comportamiento dinámico del *LMD18200*

PWM	Dirección	Freno	Salida
%	H	L	Arranque 1, dirección 1
%	L	L	Arranque 1, dirección 2
%	H	H	Freno
% valor de *PWM*, H = valor alto, L = valor bajo			

Tabla 3.3 Tabla de verdad para el *LMD18200*.

Módulo 3A Puente H

El driver es fabricado por *Acroname Inc*. Este es un modulo que utiliza el circuito integrado *LMD18200*. Este cuenta codificadores de cuadratura, protección del fusible.

Este controlador Puente H incorpora un circuito de medición de fuerza electromotriz (*Back-EMF*) que le permite discernir la velocidad del motor sin un codificador. El puente se basa en el *LMD18200* Puente H por lo que tiene una resistencia muy baja (0.4 Ω), y puede manejar corrientes de 3A de forma continua y hasta un aumento de 6A.

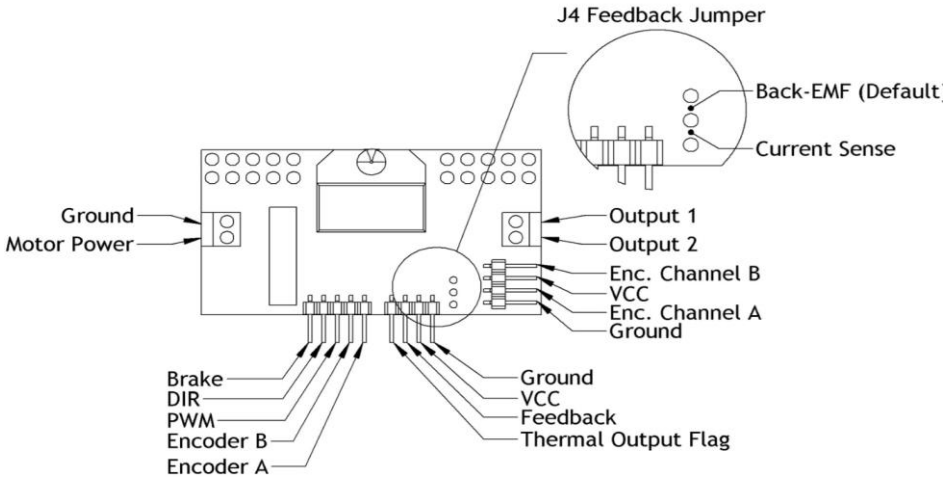

Fig. 3.16 Módulo 3A Puente H.

Características

- 3 modos de retroalimentación.
- Unidades de cargas de hasta 27,5 V.
- 3A de corriente continua.
- 6A máxima momentánea.
- *PWM* de frecuencia de 39,000-500,000 Hz.

Dimensiones físicas

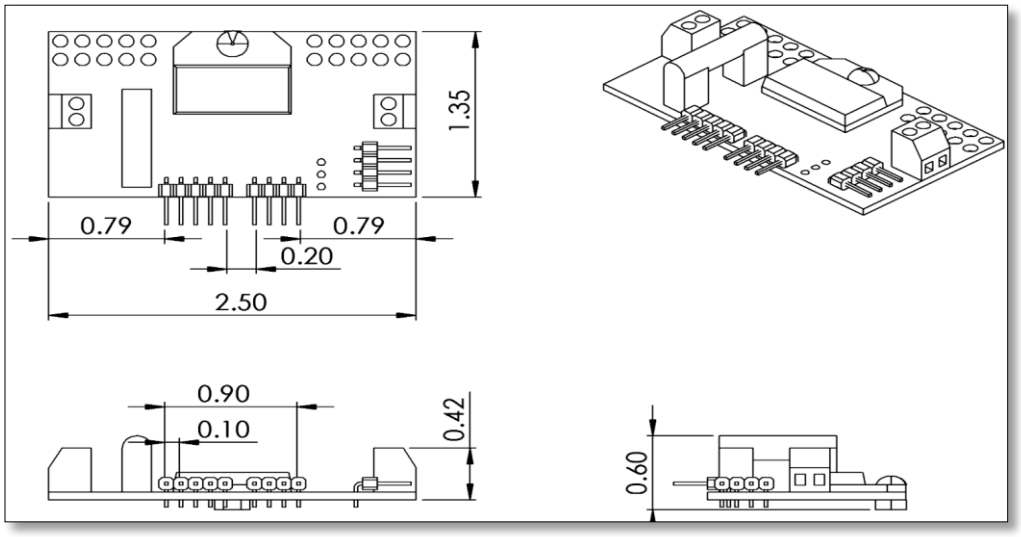

Fig. 3.17 Dimensiones de la tarjeta 3A Módulo Puente H.

3.3.3 Experimentos con motores

Experimento 1.- Control *ON/OFF*

Controlar el encendido y apagado de un motor DC conectado en el pin 3, con un botón pulsador conectado en el pin 7 se enciende el motor y al soltar el botón el motor se detiene.

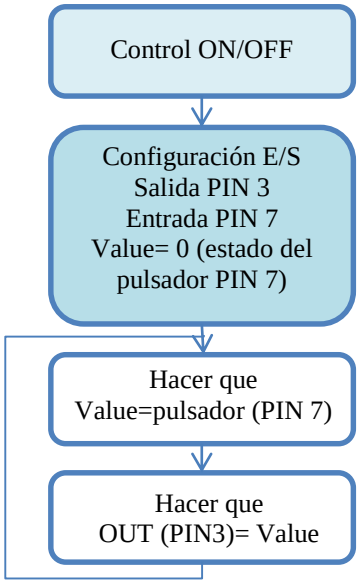

Fig. 3.18 Diagrama pseudocódigo experimento con motores 1.

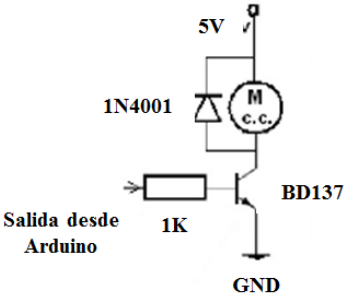

Fig. 3.19 Diagrama eléctrico de experimento 1 con motores.

Conexión física de Arduino experimento 1

Para la protección de la tarjeta de Arduino se requiere utilizar de un transistor como el *BD137* a la señal de salida el cual funcionará como un interruptor de encendido y apagado del motor. También es necesario un botón pulsador en el pin 7 el cual nos servirá para encender y apagar el motor.

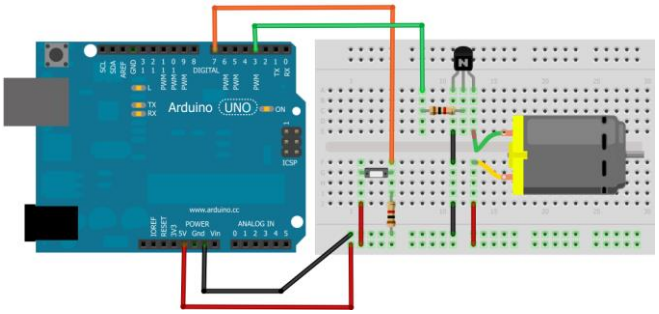

Fig. 3.19 conexión física de experimento 1 con motores.

Experimento 2.- Control de velocidad por *PWM*

Controlar la velocidad de giro de un motor DC conectado en el pin 3, con un botón pulsador conectado en el pin 7 se enciende el motor y al soltar el botón el motor se detiene.

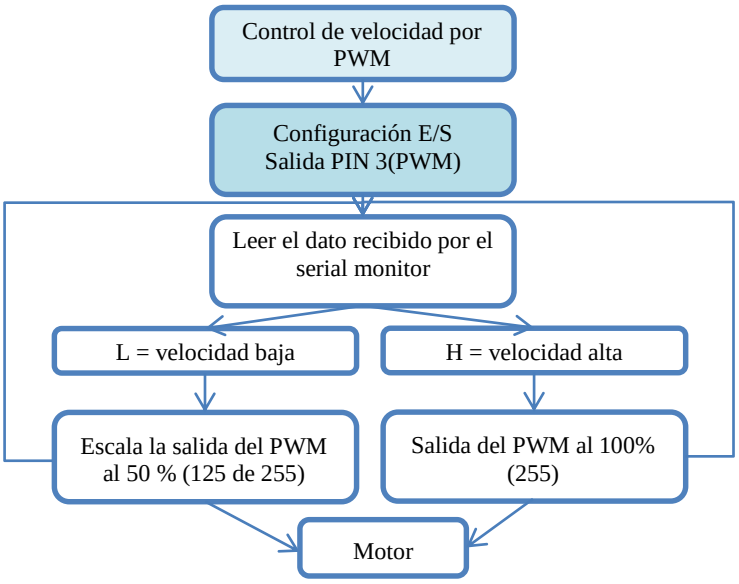

Fig. 3.18 Diagrama pseudocódigo experimento 2 con motores.

Conexión física de Arduino experimento 2

Para la protección de la tarjeta se requiere utilizar de nueva cuenta un transistor como el *BD137* a la señal de salida el cual funcionará como un switch de encendido y apagado del motor. En este experimento no se requiere de entrada física dado los datos serán introducidos por el monitor serial desde el software; para el caso ‘L’ será la velocidad baja al 50% (125) y ‘H’ será la velocidad alta al 100% (255 para 8 bits).

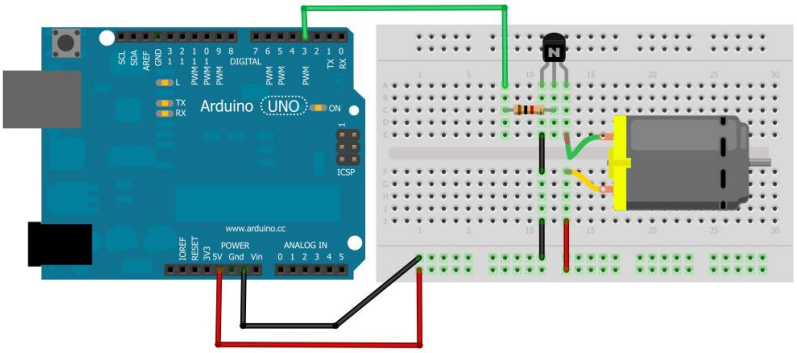

Fig. 3.21 conexión física de experimento 2 con motores.

Experimento 3.- Control un motor DC por medio de la tarjeta 3A Módulo Puente H

Controlar encendido y apagado, freno y velocidad de giro de un motor DC mediante el driver Módulo 3A Puente H y Arduino.

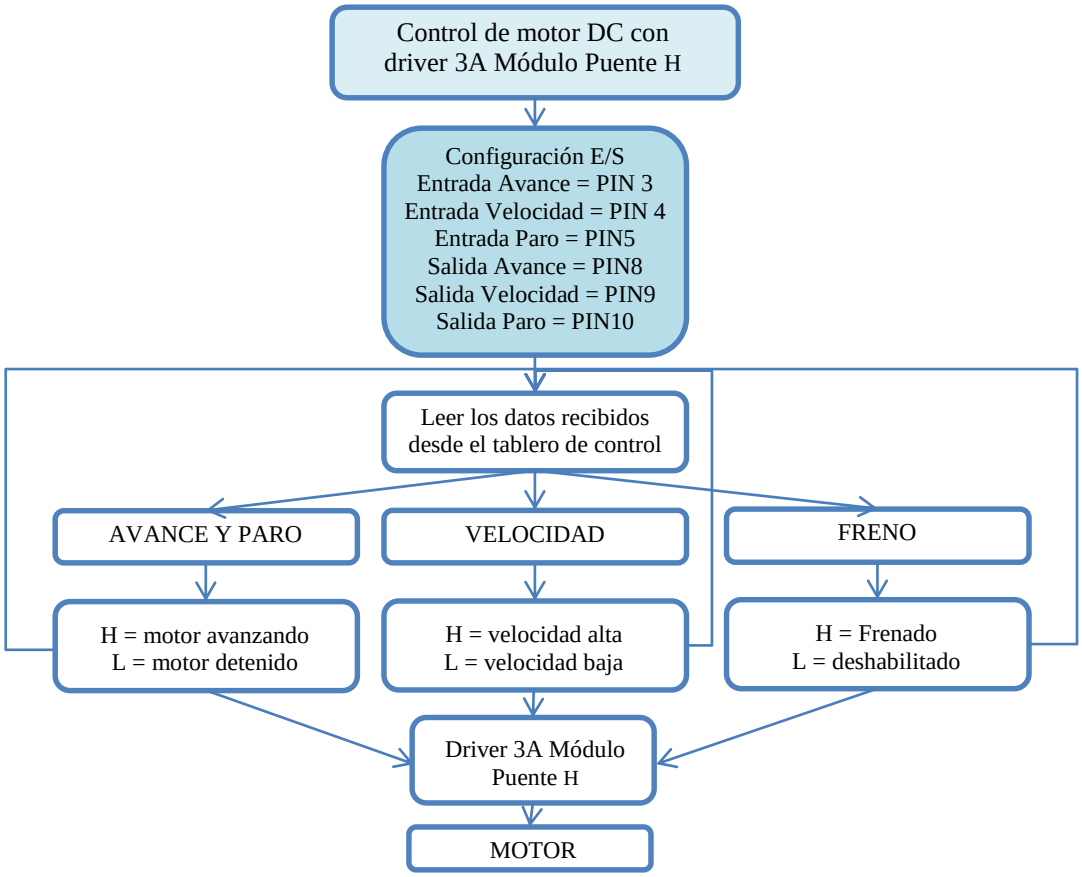

Fig. 3.22 Diagrama pseudocódigo experimento 3 con motores.

Conexión física de Arduino experimento 3

Para éste experimento se requiere conectar la placa de Arduino UNO con el modulo para control de motores 3A Módulo Puente H. Se colocarán unos botones pulsadores a las entradas de Arduino para el control de avance y paro, velocidad y freno del motor. La tarjeta 3A Módulo Puente H esta conectada un motor DC de 12v por lo cual se requiere de una fuente DC de 12v externa para alimentar el motor. La configuración descrita anteriormente se puede apreciar en la Figura 3.23.

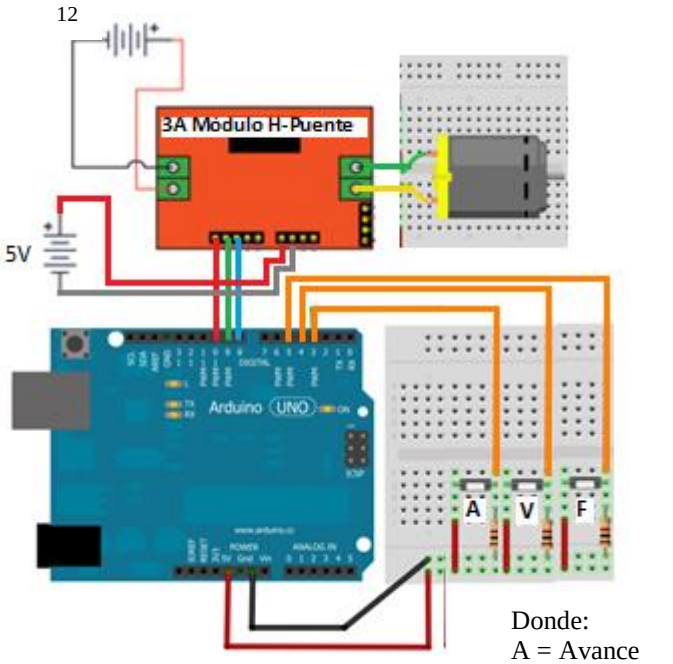

Fig. 3.23 Conexión física de Arduino con Módulo 3A Puente H de experimento 3.

3.3.4 Conclusiones

El control de motores mediante Arduino es sencillo dada su entorno de programación amigable; el control de la velocidad también es sencillo ya que Arduino cuenta con herramientas para generar señales *PWM* en los niveles deseados, por ejemplo, se puede discretizar la señal de 5v en 255 partes y tener distintas velocidades en el rango de 0 a 255.

Utilizando el Driver para control de motores Módulo 3A Puente H se facilita la etapa de potencia del sistema ya que la tarjeta cuenta con un arreglo de transistores los cuales manejan la corriente de funcionamiento a conveniencia. Además, cuenta con entradas directas para controlar al avance y paro, velocidad y freno del motor con tan solo aplicarle señales altas y bajas en dichas entradas. También cuenta con un fusible para evitar alguna falla por sobrecorriente.

3.4 Experimento de Integración

Experimento 1.- Avance y paro de un motor DC

Encender y apagar un motor que se conecta en el pin 13 de Arduino y se configura como salida. El tiempo de encendido y apagado es de 1 segundo. Se utilizará el modulo XBee y el XBee Shield para lograr la comunicación inalámbrica. También se utilizará el driver para motores modulo 3A Puente H para la etapa de potencia del circuito.

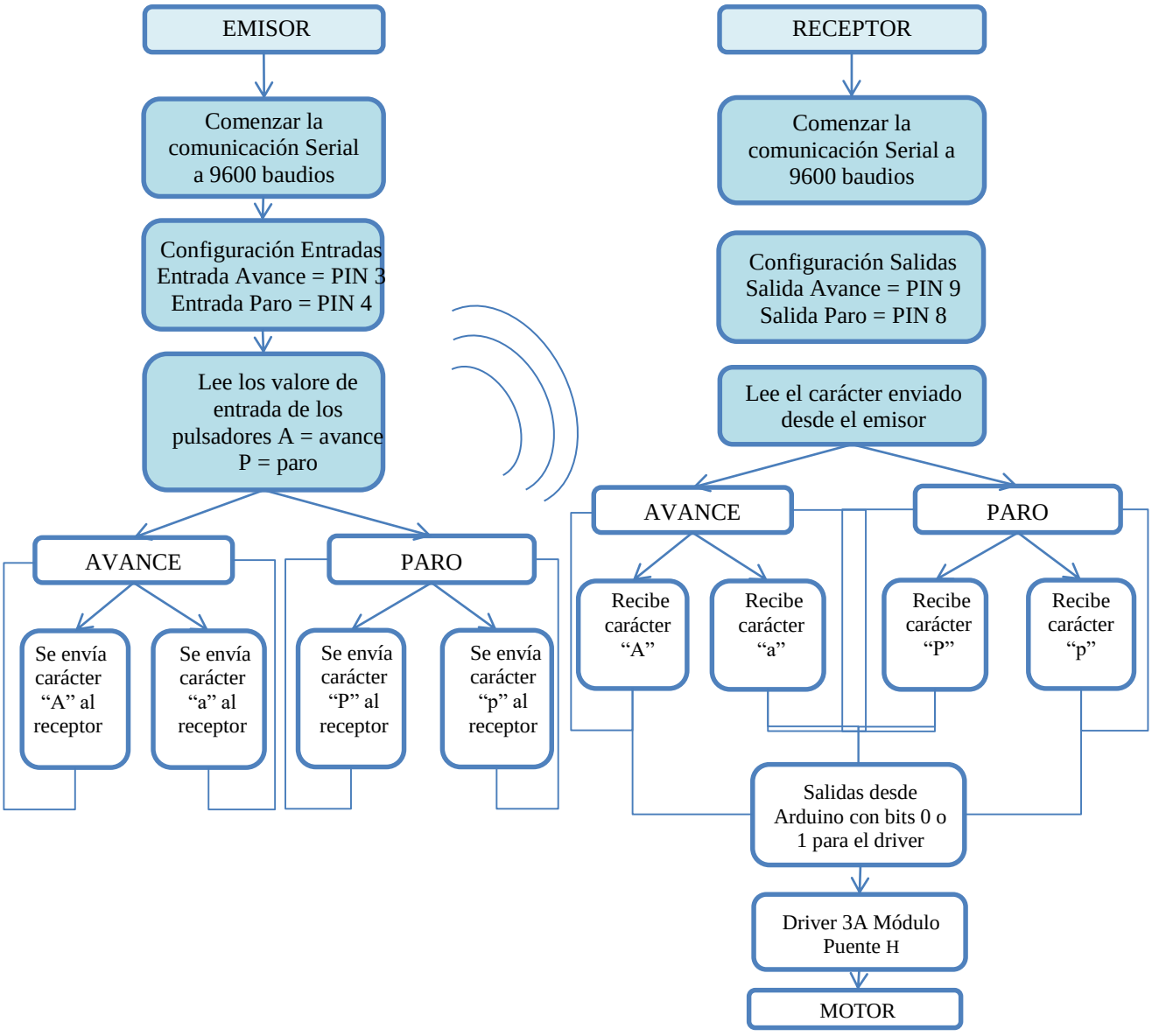

Fig. 3.24 Diagrama pseudocódigo experimento de Integración.

Capítulo 4

4. Desarrollo

En este capítulo se desarrollan los pasos llevados a cabo para alcanzar el objetivo principal del proyecto de tesis planteado anteriormente.

4.1. Arduino y la computadora

Para conectar el Arduino con una computadora y poder ser programado es necesario contar con un cable *USB* del tipo AB (cables utilizados en la conexión de impresoras, discos externos, etcétera) con el cual se realizará la transferencia de información.

Una vez que Arduino se conecta a la computadora mediante cable *USB* se enciende el led verde de la placa el cual indica que la fuente de alimentación esta conectada (el pin es nombrado como *PWR* en la placa); este permanecerá encendido mientras se encuentre conectada la placa indicando que Arduino se encuentra funcionando y está listo para ser utilizado.

4.1.1 Instalación de *drivers*

Primeramente es necesario descargar los archivos de instalación desde la página oficial de Arduino. [15]
Cuando la placa Arduino es conectada a una computadora en el sistema operativo de *Windows* la instalación de los drivers se inicializa automáticamente, pero la selección de los drivers debe realizarse de forma manual (como se aprecia en la siguiente figura 4.1), es decir, indicando la dirección de donde se encuentran guardados en la computadora; estos están guardados en la carpeta descargada previamente junto con el IDE. Estos se encuentran dentro

de la carpeta descargada; una vez seleccionada esta carpeta se instalan los drivers en la computadora correctamente.

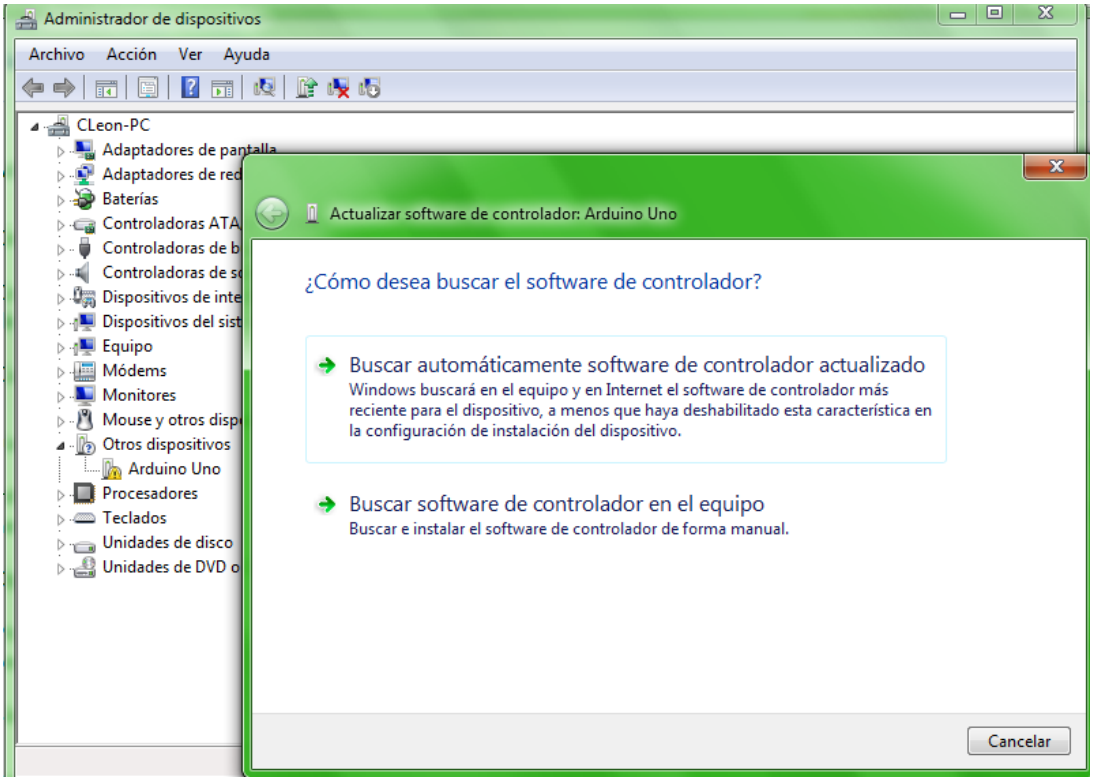

Fig. 4.1 Instalación de manual *Driver*.

Para comprobar que los drivers se han instalado correctamente se puede abrir la carpeta del Administrador del Dispositivos, en la sección Dispositivos del panel de control del sistema. Se debe encontrar "*COMUNICATIONS PORT (COM)*" y se observa el número del puerto seleccionado para Arduino (como se observa en la figura 4.2).

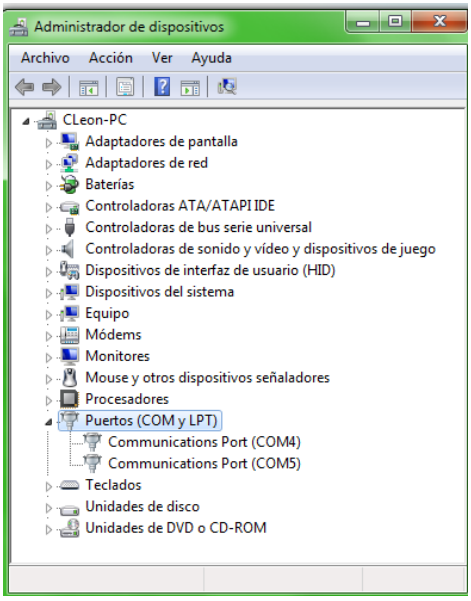

Fig. 4.2 Puertos de la computadora utilizados para la comunicación de Arduino.

4.1.2 Arduino IDE

Tras haber descargado el archivo (desde la página oficial de Arduino [15]), se debe abrir el archivo ejecutable llamado "Arduino.exe", esta es la aplicación con la cual se pueden programar y compilar las placas. Esta aplicación no requiere de instalación alguna en la computadora.

Fig. 4.3 Arduino IDE.

Para familiarizarse con el *software* se puede utilizar un *sketch* ejemplo el cual hace parpadear un led: ("*LED blink*"): *File > Examples > basics > Blink.*

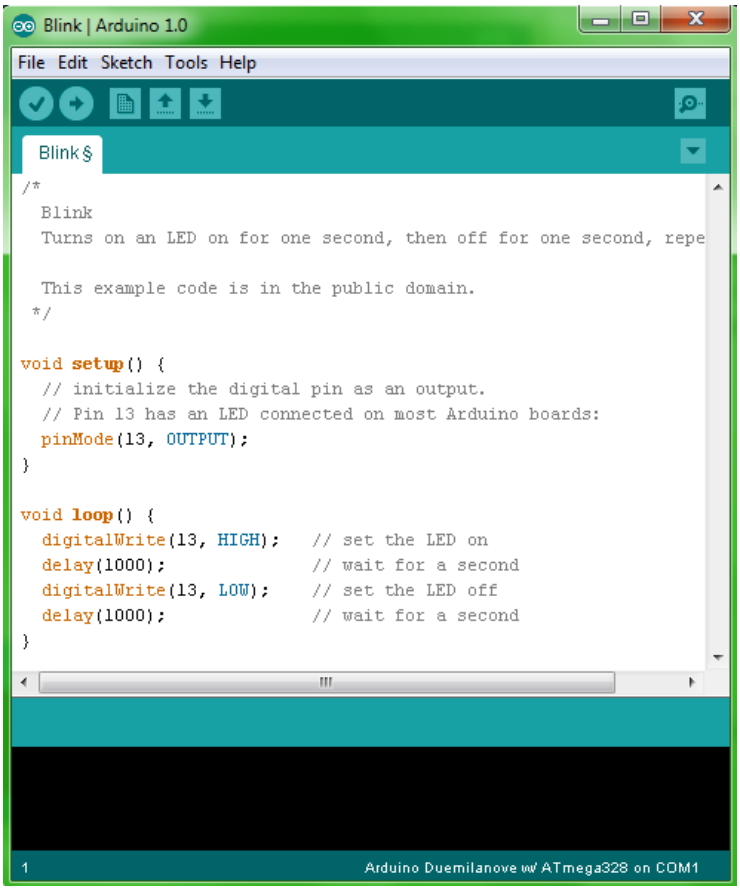

Fig. 4.4 Arduino IDE programa de prueba.

Este es un programa básico para aprender a usar el IDE y las placas de Arduino.

Selección de placa

Es necesario seleccionar el tipo de placa de Arduino a utilizar; esto se hace en el menú *Tools* > *Board* > *Arduino UNO* (para este caso, Fig. 4.5).

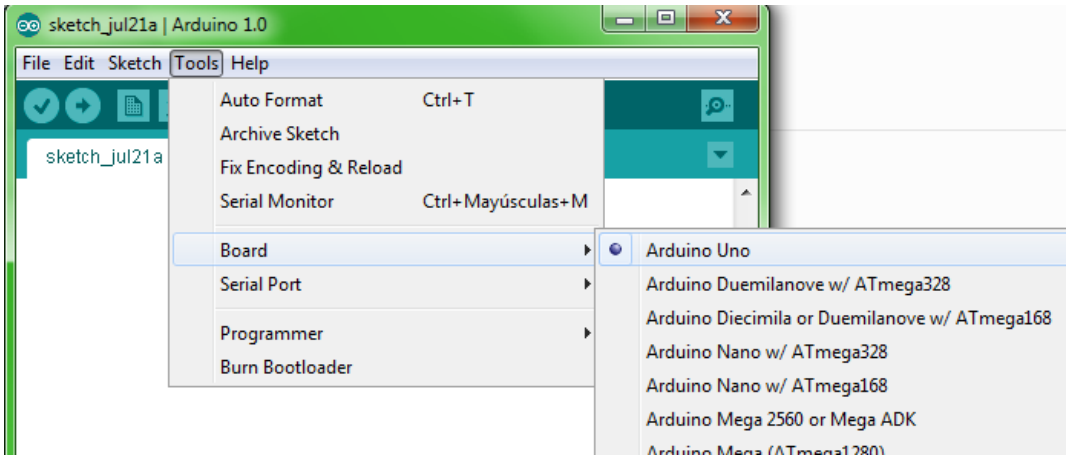

Fig. 4.5 Selección de placa.

Selección del puerto serie

Se selecciona el dispositivo serie de la placa Arduino en el menú *Tools* > *Serial Port* (Herramientas > Puertos Serie). Para asegurarse de cual es el correcto, se puede desconectar la placa y volver verificará el menú; el puerto de la placa habrá desaparecido de la lista. Ver figura 4.6

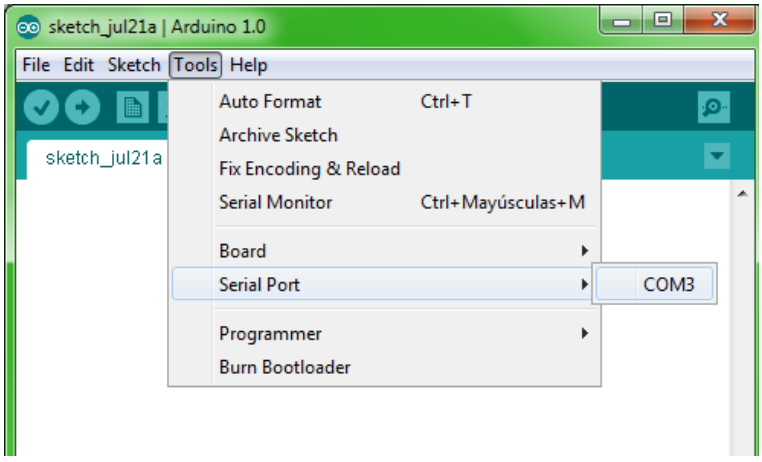

Fig. 4.6 Selección del puerto de comunicación.

Cargar el *sketch* a la placa

Se pulsa el botón "*Upload*" en el entorno Arduino. Despues de unos pocos segundos parpadean los leds RX y TX de la placa. Si la descarga del código es exitosa aparecerá el mensaje "*Done uploading*" en la barra de estado asi como se muestra en la figura 4.7.

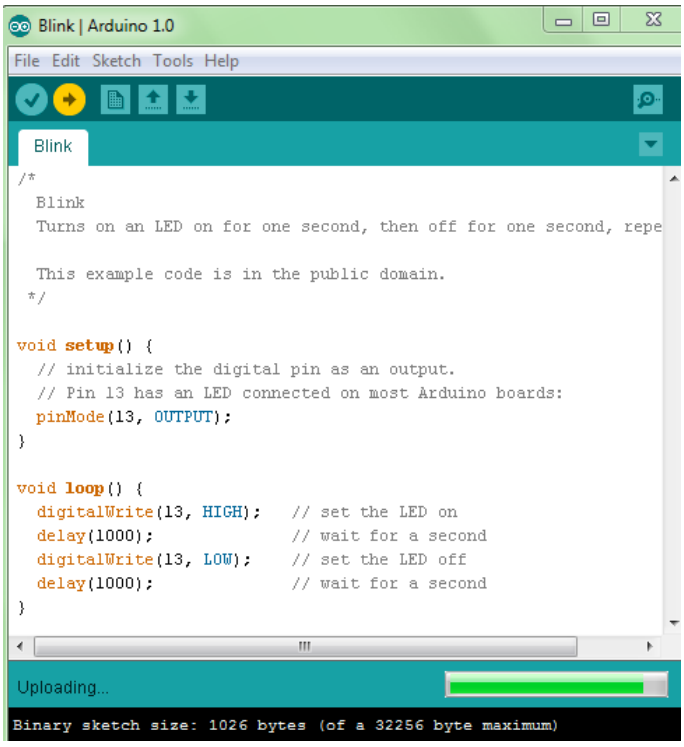

Fig. 4.7 Programa cargado a la placa.

Unos pocos segundos despues de finalizar la descarga del programa el led de la placa conectado al pin 13 comenzará a parpadear un segundo encendido y otro segundo apagado.

4.2 Configuracion de los modulos XBee

Los modulos deben ser configurados previeamente para que puedan ser utilizados. De fábrica cada módulo XBee viene configurado con un *PAN ID* (el identificador de la red personal) de 3332 y configurados con una tasa de transferencia de 9600 baudios, con datos de 8 bits, sin paridad y 1 bit de paro, utilizando estos valores de fabrica los módulos funcionan correctamente. En este caso para configurarlos se utilizará el XBee Shield de Arduino; este tiene un par de jumpers con los cuales se define si la comunicación serial se realiza hacia el puerto *USB* o hacia el módulo XBee, para ellos es necesario colocar los *jumpers* en modo *USB* y debe ser retirado con mucho cuidado el microcontrolador ATMEL de la placa.

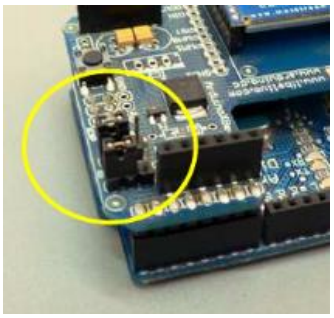

Fig. 4.8 Selección de pines para configuración de módulos XBee.

Con la ayuda del software llamado X-CTU se configuran los modulos XBee para establecer cual placa será el dispositivo coordinador y cual placa el dispositivo final para la comunicación de datos. El software puede ser descargado desde la pagina http://www.digi.com [16]

Se conecta a la computadora la placa de Arduino con el Xbee Shield y módulo Xbee montado. Una vez que se haya detectado el puerto (en este caso el puerto *USB*) se observan los puertos en los cuales se ha conectado la placa Arduino (ver Figura 4.9). A continuación se debe hacer click en el boton de *test/query* (Figura 4.10), con ello el *software* idenficará el *firmware* del módulo y si tiene alguna versión desactualizada se actualizará automaticamente.

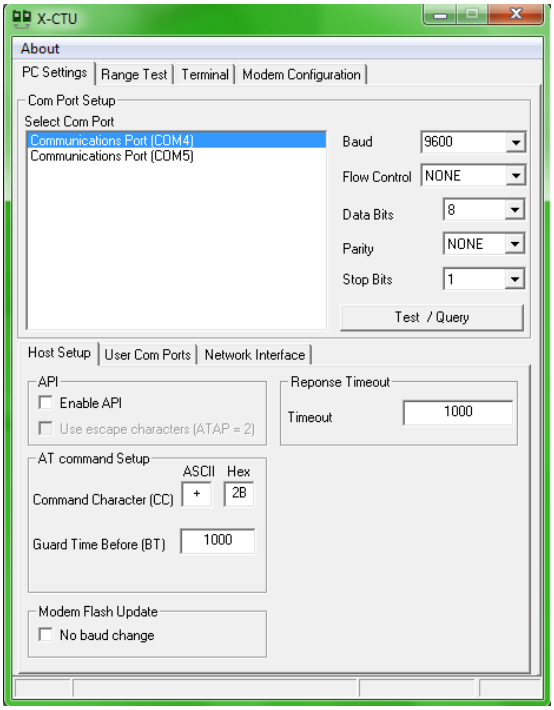

Fig. 4.9 Software para configuracion de módulos XBee.

Fig. 4.10 Firmware de módulos XBee.

En la parte superior en la ventana del *software* se tienen cuatro pestañas, se selecciona la cuarta pestaña "*modem configuration*" con esto se abrirá una lista de los parametros a configurar en los modulos Xbee. Como ya se mencionó, los módulos vienen previamente configurados de fabrica por lo cual solo es necesario seleccionar el tipo de Xbee que se esta utilizando el cual aparecerá en la pestaña "*modem XBEE"*, ademas se debe seleccionar tambien el tipo de dispositivo al que va a simular, para este caso es necesario un coordinador y un dispositivo final, estos se seleccionan en la pestaña de "*function set*".

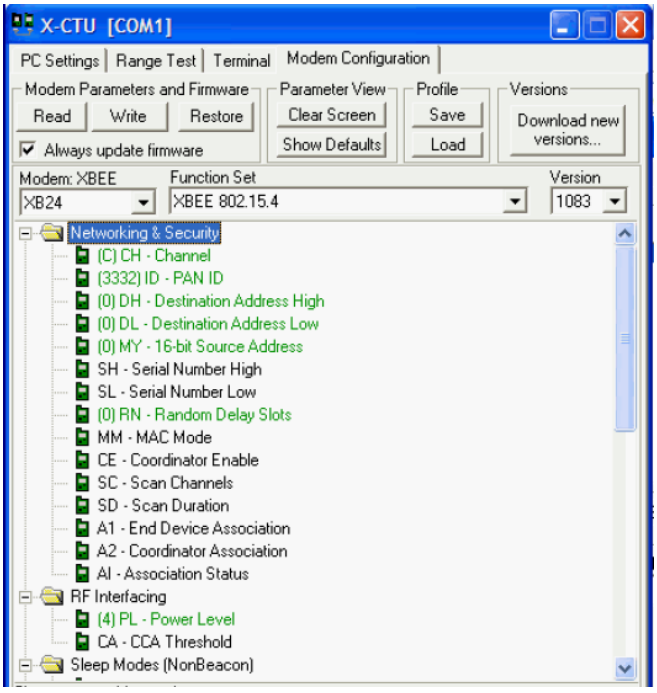

Fig. 4.11 Configuración de parámetros para módulos XBee.

Después se debe de pulsar en el botón *"write"* para cargarle los parámetros seleccionados al modulo. En ocasiones ocurren problemas durante, cuando esto suceda se debe de repetir el proceso hasta que se carguen correctamente los parámetros elegidos.
Se debe de realizar el mismo proceso de configuración para ambos módulos XBee, en caso de modificar los valores de los parámetros se debe verificar que el cambio se realice en los dos módulos para que estén en el mismo canal de comunicación y puedan transferir datos correctamente.

4.3 Arduino y XBee

Una vez configurados los módulos inalámbricos Xbee se pueden montar sobre la placa de Arduino utilizando el *Shield*, los *jumpers* de estos deben estar en la posición de XBee para que los dos dispositivos puedan comunicarse inalámbricamente.
Como ya se a mencionado se requiere de un dispositivo coordinador y un dispositivo final, por lo cual los programas deben ser diseñados de tal manera que uno de ellos envíe las señales y el otro reciba las señales y actúe conforme a las señales de entradas recibidas.
A continuación se muestra el código utilizado para el dispositivo coordinador:

```
//CONTROL PARA MOTOR DC CON XBEE, --- PROGRAMA TRANSMISOR---

int botonEncendido=2;

int botonFreno=3;

int botonDireccion=4;

int botonVelocidad=5;

int botonVelocidad2=6;

  void setup (){//-------------------------------------------

    pinMode (botonEncendido, INPUT);

    pinMode (botonFreno, INPUT);

    pinMode (botonDireccion, INPUT);

    pinMode (botonVelocidad,  INPUT);
```

```
    pinMode (botonVelocidad2,  INPUT);
    Serial.begin (9600);}
  void loop (){
//ENCENDIDO-----------------------------------------------
    int estado1=digitalRead(botonEncendido);
    if (estado1==1){
    Serial.println('E');
   // else{ Serial.println ('e');}
//FRENO---------------------------------------------------
    int estado2=digitalRead(botonFreno);
    if (estado2==1){
    Serial.println('F');}
    else{ Serial.println ('f');}
//DIRECCION-----------------------------------------------
    int estado3=digitalRead(botonDirección);
    if (estado3==1){
    Serial.println('D');}
    else{ Serial.println ('d');}
//VELOCIDADES---------------------------------------------
    int estado4=digitalRead(botonVelocidad);
    int estado5=digitalRead(botonVelocidad2);
      if (estado4 == 0 && estado5 == 1){
        Serial.println('V');}
      if (estado4 == 0 && estado5 == 0){
```

```
        Serial.println ('v');}
      if (estado4 == 1 && estado5 == 0){
        Serial.println ('b');}
delay (50); }else{
  Serial.println('e');
  Serial.println('f');
  Serial.println('d');
  Serial.println('B');
}}
```

El código utilizado para el dispositivo final es el siguiente:

```
//CONTROL PARA MOTOR DC CON XBEE -----PROGRAMA RECEPTOR--------------
; //declara las variables a usar y el pin donde se ubican
const int Encendido = 12
const int Freno = 11;
const int Direction = 10;
const int Velocidad =9;
  void setup() {//-----------------------------------------------------
    Serial.begin(9600);          // inicializa la comunicación serial
    pinMode(Encendido, OUTPUT);  //declara las entradas digitales
    pinMode(Freno, OUTPUT);
    pinMode(Direccion, OUTPUT); }
  void loop() {//------------------------------------------------------
  if (Serial.available() > 0) {
 // verifica si hay entrada de datos en el serial
```

```
   int boton = Serial.read();
// lee la entrada recibida en el serial
//--------------------------------------------------------------------
    if (boton == 'E') {                   //ENCENDIDO
        digitalWrite(Encendido, HIGH);}   //ON
    if (boton == 'e'){
        digitalWrite(Encendido,LOW);}     //OFF
//--------------------------------------------------------------------
    if (boton == 'F') {                  //FRENO
        digitalWrite(Freno, HIGH);}      //ACTIVADO
    if (boton == 'f'){
        digitalWrite(Freno, LOW);}       //DESACTIVADO
//--------------------------------------------------------------------
    if (boton == 'D') {                   //DIRECCION
        digitalWrite(Direccion, HIGH);}   //IZQUIERDA
    if (boton == 'd'){
        digitalWrite(Direccion, LOW);}    //DERECHA
//--------------------------------------------------------------------
if (boton == 'V') {                     //VELOCIDADES
        int val = 255;                    //VELOCIDAD ALTA |||
    analogWrite(Velocidad,val);}
    if (boton == 'v') {
    int val = 120;                        //VELOCIDAD MEDIA ||
    analogWrite(Velocidad,val);}
```

```
    if (boton == 'b'){
        int val = 30;                   //VELOCIDAD BAJA   |
    analogWrite(Velocidad, val); }
    if (boton == 'B'){
        int val = 0;                    //DETENIDO
    analogWrite(Velocidad, val); }
}}
```

Los códigos fueron diseñados especialmente para la practica del proyecto de tesis y pueden ser utilizados y modificados a conveniencia

Se deberán cargar los códigos mostrados anteriormente en cada módulo XBee según sea el caso.

Conexiones físicas de las placas

Para la conexión física de la placa con función de coordinador, es decir, la placa que enviará las señales requiere de un tablero de control el cual consta de varios *switchs* para cada una de las variables que se desean controlar como el encendido, la dirección, el freno y la velocidad. La fuente de alimentación del sistema puede ser una fuente de voltaje conectada a un tomacorriente o bien se puede conectar la tarjeta a la computadora (solo para alimentación no para transmisión de datos). Ver figura 4.12

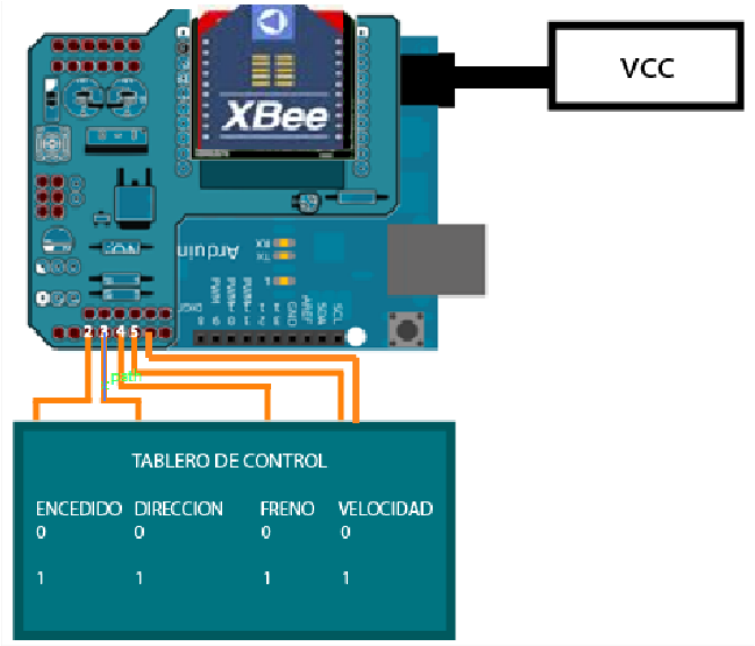

Fig. 4.12 Conexión física para la placa Coordinador.

En la siguiente figura se muestra el diagrama eléctrico del tablero de control.

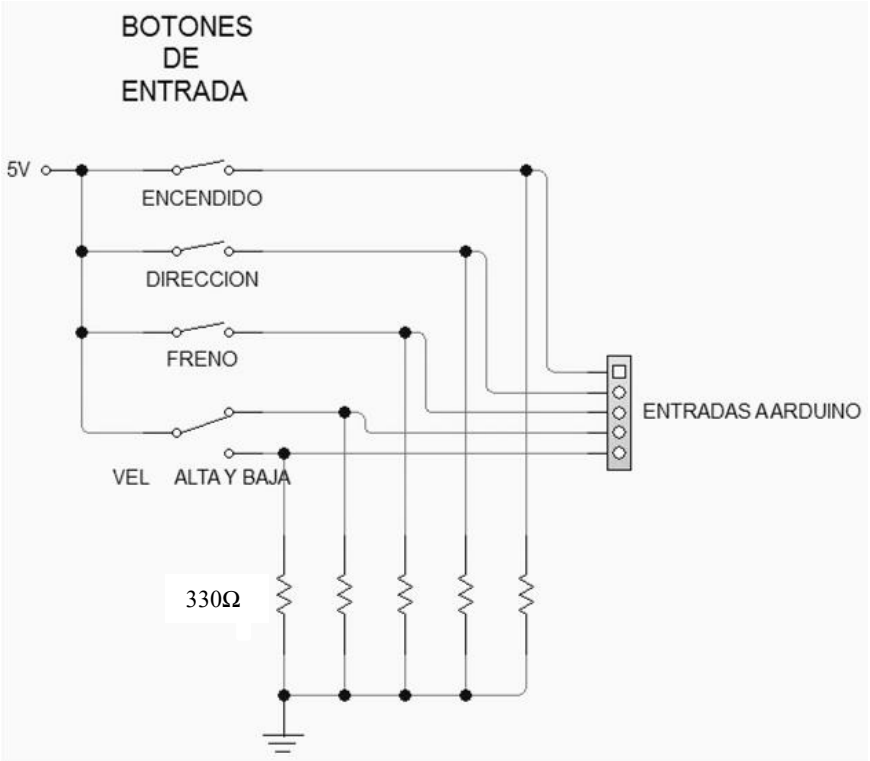

Fig. 4.13 Diagrama eléctrico para las conexiones de señales de entrada.

Para el control de la velocidad se utilizó un interruptor de dos polos tres tiros para tener diferentes valores en el mismo interruptor.

En la conexión física para la placa que cumple la función de dispositivo final deben de ser conectadas las salidas de Arduino con las entradas del *driver* para el motor de DC (las salidas de Arduino fueron previamente seleccionadas en el programa deben concordar con las del *driver*).

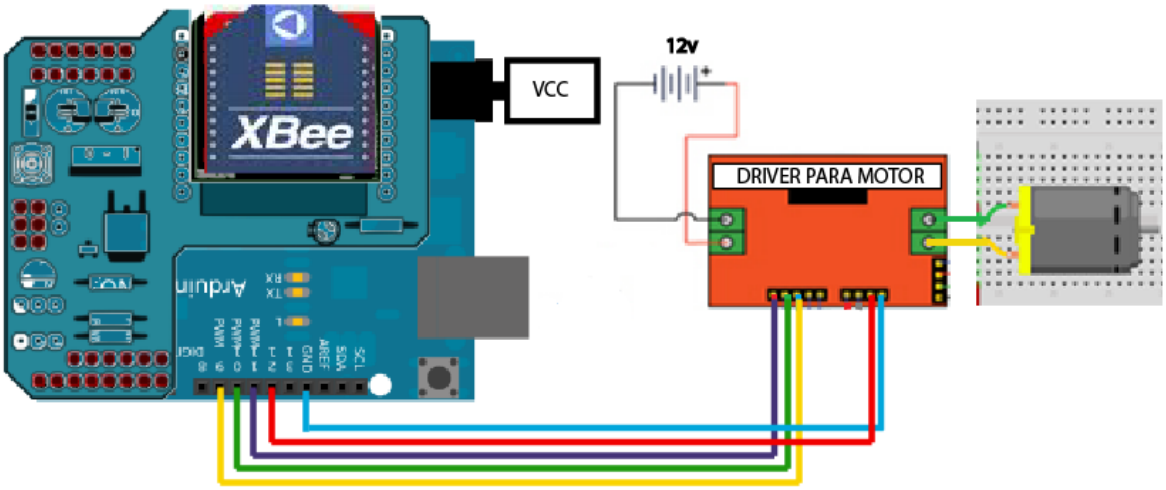

Fig. 4.14 Conexión física para la placa Dispositivo final.

Una de las salidas de Arduino va conectada al pin de alimentación del *driver* de esta manera es posible habilitar o deshabilitar el funcionamiento de la placa simulando un interruptor de encendido y apagado. (En la figura 4.14 se puede observar como la conexión de color rojo).

Capítulo 5

5. Conclusiones y trabajo futuro

5.1 Conclusiones

La controlar de diversos parámetros en los motores de corriente directa (DC) como el sentido de giro, arranque y paro, el freno, y la velocidad son habilidades de gran importancia en los proyectos ingenieriles de la actualidad ya que al poder manipular a conveniencia dichos parámetros es posible realizar un gran numero de usos y aplicaciones industriales para cumplir con diversas tareas. Conocer como manipular estos motores es de gran utilidad ya que son de los principales actuadores en la industria.

Durante el desarrollo del proyecto de tesis se demostró que la implementación de las redes de comunicaciones inalámbricas son opciones viables para el control de dispositivos a distancia, que para este caso dichos dispositivos utilizados fueron motores de corriente directa. Al utilizar estos sistemas inalámbricos se hace más sencillo el desarrollo de una infraestructura de comunicación en los proyectos, además genera un menor costo con respecto a un sistema alámbrico, un sencillo control de información y también facilita la detección errores al momento de algún problema en la comunicación.

La selección del protocolo de comunicación es paso muy importante en el proyecto, si se llegará a elegir un protocolo inadecuado podría funcionar tal vez, pero también podría traer diversos problemas y complicaciones; un claro ejemplo de esto es una mala transferencia de datos y por otra parte se podría contar con rangos de comunicación excesivos de lo necesario con lo cual solo se generaría un mayor gasto económico innecesario.

El protocolo de comunicaciones ZigBee es un buen estándar de comunicación para el proyecto llevado acabo a la maximización de la vida útil de sus baterías, a su bajo costo de adquisición y a su amplio rango de comunicación. Con este protocolo es posible controlar los motores de

corriente directa (DC) con los rangos de distancia requeridos para el proyecto y es ideal para las conexiones de punto a punto la cual es utilizada para los fines del proyecto.

En tanto al desarrollo del sistema de control de los dispositivos con la plataforma Arduino se facilita el proceso ya que este está diseñado de tal manera que cualquier persona pueda aprender a utilizarlo con facilidad y también familiarizarse rápidamente con el lenguaje y el entorno de programación con el que cuenta. Arduino es muy utilizado en proyectos de esta índole dado que es un software libre con lo cual se tiene una adquisición gratuita y sencilla para todos los usuarios.
Utilizando la tarjeta *XBee Shields* es posible comunicar de una manera sencilla y eficiente dos o mas dispositivos a distancia, esta tarjeta es la parte intermedia entre la conexión de Arduino con los módulos XBee.

Combinar las tecnologías inalámbricas con el control de actuadores abre una nueva posibilidad para el desarrollo de proyectos más complejos en la facultad de ingeniería Mecatrónica en la Universidad de Sonora.

5.2 Trabajos futuros

El proyecto de tesis cumplió con el objetivo de controlar un motor de corriente directa (DC) mediante una comunicación inalámbrica, este método de transmisión de información puede ser aplicado a diversos proyectos mecatrónicos futuros en la Universidad, un claro ejemplo es la aplicación que podría tener en el campo de pruebas de helióstatos de la Universidad de Sonora, al utilizar una tecnología de comunicación inalámbrica se ahorraría la infraestructura de cableado para la comunicación, de igual manera la detección de errores y/o problemas de comunicación entre el dispositivo coordinador y el dispositivo final seria mas rápida puesto que ya no se tendrían que buscar los problemas en los cableados, sino desde la red del dispositivo controlador.

Para lograr controlar la cantidad total de motores que se tienen en el campo de helióstatos se requeriría de una distinta topología de red con la cual se pudiesen controlar más de un dispositivo final.

Con esto se tendría una mayor optimización de los recursos tecnológicos con los que cuenta el campo de Helióstatos.

Bibliografía

[1] Microcontroladores PIC: sistema integrado para el autoaprendizaje. MARCOMBO, EDICIONES TECNICAS 2007, MARCOMBO S.A. Enrique Mandado Pérez, Luis Menéndez Fuertes.

[2] Arquitectura y programación de Microcontroladores Juan Manuel Orduña Huertas,Vicente Arnau Llombar. Universidad de Valencia 1996.

[3] Microcontroladores: fundamentos y aplicaciones con PIC. Ramón Pallás Areny. 3Q editorial.

[4] Maquinas eléctricas y sistemas de potencia .6ta ed. Wildi, Theodore Prentice hall.

[5] Principios de los microcontroladores Norberto Malpica Dpto de Tecnología Electrónica universidad Rey Juan Carlos.

[6] Equisbí: Desarrollo de aplicaciones con comunicación remota basadas en modulos zigbee y 802.15.4 Sergio R. Caprile 1 Ed. Gran Aldea Editores.

[7] Curso de máquinas de corriente contínua. Gilberto Enríquez Harpe. Limusa, Noriega Editores.

[8] Electricidad industrial CH. L. DAWES Editorial Reverté,S.A, Mc Graw Hill.

[9] Getting Started with Arduino. Massimo Banzi, 2da Ed O´REILLY.

[20] CIM, el computador en la automatización de la producción Andrés García Higuera,Fernando J. Castillo García. Ediciones de Universidad de Castillas-La Mancha.

[21] Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensors with IEEE 802.15.4 José A. Gutiérrez,Edgar H. Callaway,Raymond L. Barrett. Estandars information network.

Web sites:

[10] http://www.atmel.com/tools/ATMELAVRSTUDIO.aspx

(Consultado el 12/04/2012)

[11] http://zigbee.org

(Consultado el 15/03/2012)

[12] http://www.microchip.com

(Consultado el 12/04/2012)

[13] http://www.atmel.com/

(Consultado el 21/03/2012)

[14] http://www.zigbee.org/

(Consultado el 15/03/2012)

[15] http://www.arduino.cc/es/

(Consultado el 01/03/2012)

[16] http://www.digi.com/

(Consultado el 05/03/2012)

[17] http://es.scribd.com/doc/53085708/7/%C2%BFQue-microcontrolador-elegir

(Consultado el 24/02/2012)

[18] http://www.bluetooth.com

(Consultado el 15/04/2012)

[19] http://www.wi-fi.org/

(Consultado el 15/04/2012)

[22] http://es.scribd.com/doc/41130520/XBee-Guia-Usuario

(Consultado el 26/02/2012)

[23] http://www.datasheetcatalog.org/datasheet/nationalsemiconductor/DS010568.PDF

(Consultado el 21/04/2012)