TECNOLÓGICO DE MONTERREY

Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Ciudad de México
Proyecto de Ingeniería

Control a Distancia de un Faro Mediante HF

Alan Roberto Gómez Rosas A00995349
Jorge Marcin Cornish A00994748
Javier Rodríguez Lizardi A00996092
Mauricio Orvañanos Riquelme A00993448
Asesor: Dr. José Ramón Alvares Bada
Co-Asesor: M.C Israel Macías Hidalgo
Profesor: M.C Edgar Omar López Caudana

México D.F. a 26 de noviembre de 2007

CONTROL A DISTANCIA DE UN FARO MEDIANTE HF

ÍNDICE

1. INTRODUCCIÓN

1.1 Problemática

México es uno de los países con mayores ventajas competitivas para el comercio marítimo, contamos con 108 puertos distribuidos en 11 mil kilómetros de litorales, de los cuales 38 son internacionales.

No obstante, el principal problema es ***"la inexistencia de un sistema automático, económico y confiable para el control y monitoreo a distancia de un faro en el Sector Marítimo Mexicano".***

En el sexenio del presidente Vicente Fox Quesada se puso en marcha el Programa de Modernización del Sistema de Señalamiento Marítimo, para garantizar condiciones adecuadas de seguridad de la navegación en los litorales, puertos y vías navegables del país, y dar cumplimiento a las disposiciones de convenios internacionales como el SOLAS (Safety of Life at Sea) [12].

Es importante tener un desarrollo en la infraestructura del Sector Marítimo Mexicano que permita impulsar el crecimiento económico en un contexto de apertura al comercio internacional, para esto se requiere, necesariamente, de servicios de comunicaciones y transportes confiables y competitivos.

1.2 Objetivo

"Diseñar e implementar un sistema para el monitoreo y control a distancia en tiempo real de las funciones del faro rotatorio Max Lumina TRB-220 mediante radiocomunicación en la banda HF"

Las señales que pretendemos controlar y monitorear son:

1. Alarma de Rotación
2. Alarma de Lámpara
3. Encendido remoto
4. Apagado remoto
5. Detector de Intrusos

El faro rotatorio Max Lumina TBR-220, es un faro más moderno que incorpora varias funciones avanzadas en su diseño que lo hacen tener un sistema de control más robusto. Entre ellas se encuentra el cambio de lámpara de manera automática si alguna de las lámparas se funde. Es por ello que seleccionamos estas 5 señales.

2. DESCRIPCIÓN DEL PROYECTO

2.1 Propuesta desarrollada

Se diseñó un sistema de monitoreo y control a distancia en tiempo real de las funciones del faro rotatorio Max Lumina TRB-220 utilizando los radios Codan NGT SR transceiver de nueva generación como medio de transmisión.

Es importante señalar, que ***los radios son únicamente el medio de transmisión***, y que nuestra aportación consistió en diseñar y desarrollar todo el sistema de monitoreo y control. Esto es, leer las señales provenientes del faro a través de la tarjeta, programar la tarjeta de control para que esté monitoreando constantemente al faro, interprete las señales y automáticamente enlace los radios para dar aviso al usuario conectado en el centro de control. Una vez que llega la alarma en mensaje de texto, el usuario contará con una interfaz gráfica para tomar decisiones. El programa enviará la orden de regreso a través de HF, al otro radio donde éste se encargará de transmitirle a la tarjeta de control el mensaje que representa la acción a seguir. La tarjeta de control validará en una tabla el mensaje y actuará a través de dispositivos electrónicos adaptados al faro.

2.2 Componentes involucrados

2.2.1 Enlace de radio entre el faro y el punto de monitoreo

Se propone un sistema de transmisión automática de datos en la banda de HF. En esta banda no se requiere una infraestructura para su operación, lo cual ahorra costos considerables.

El radio marca Codan, modelo NGT SR transceiver, tiene un buen desempeño ya comprobado en la Marina Mercante. El radio opera en el modo de voz y datos utilizando genéricamente los mismos recursos. Se puede ampliar para datos de alto volumen con una transferencia hasta de ~6 kbps [13].

Figura 2.1 Radio Codan, modelo NGT SR transceiver [1]

[1] Disponible en: http://www.mobilecomms.com.au/Products/Codan/easy.htm

2.2.2 Sistema de radiación de HF

La antena dipolo de la serie Código 463 de multi-alambre de banda ancha es una antena tipo doblada de 3 alambres, diseñada para la operación con estaciones fijas. Disponible en diferentes niveles de potencia hasta 1 kW. La antena se puede instalar horizontalmente entre dos soportes o como 'V' invertida usando un solo mástil de soporte central.

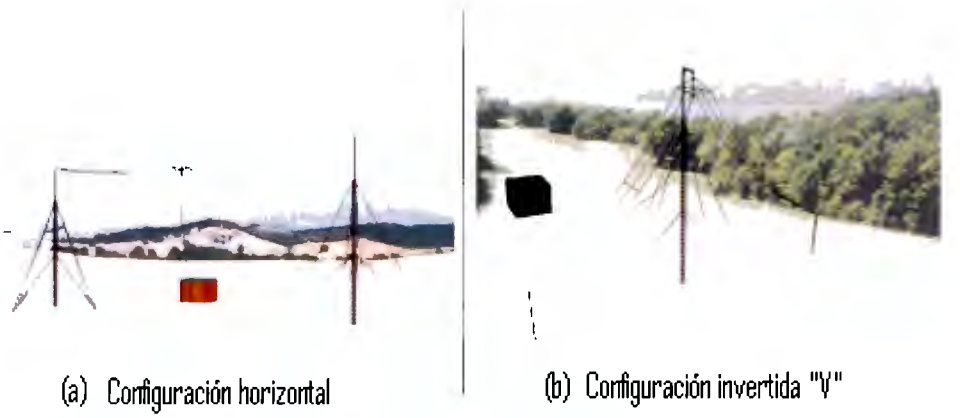


Figura 2.2 (a) Antena dipolo de la serie Código 463 de multi-alambre de banda ancha, configuración horizontal. (b) Antena dipolo de la serie Código 463 de multi-alambre de banda ancha, configuración invertida "V" [2]

2.2.3 Tarjeta de Control

Se propone un controlador inteligente para la lectura de los sensores de los datos, almacenamiento y envío, aparte de la generación del historial. El concepto del sistema de adquisición de datos está totalmente flexible en su empleo y su volumen de crecimiento.

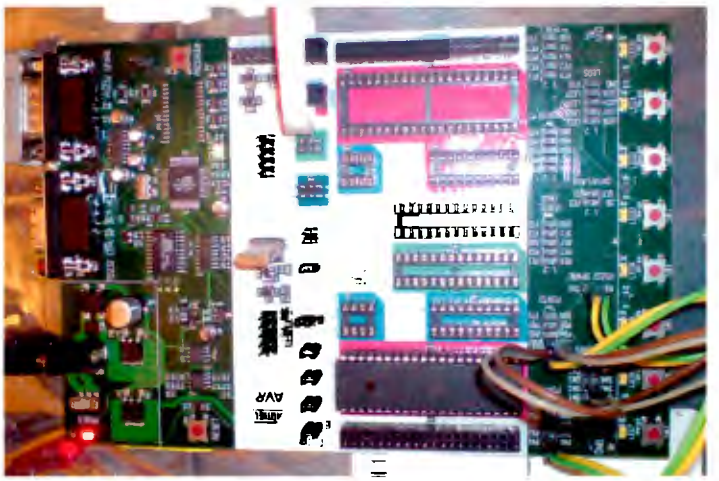


Figura 2.3 Tarjeta de control

[2] Disponible en: http://www.codan.com.au/HFRadio/Products/Antennas.htm

2.2.4 Faro

Faro rotatorio Max Lumina TRB-220, es una linterna de alta intensidad capaz de proporcionar caracteres de destello simples y complejos. Se puede configurar para proporcionar alta eficiencia con intensidades efectivas utilizando un amplio rango de linternas marinas de bajo vataje. EL TRB-220 utiliza el OMNIBUSTM II TF-3B de Tideland utilizando tres (3) lámparas marinas preenfocadas de un solo contacto y de 12 VCD.

Figura 2.4 Faro rotatorio Max Lumina TRB-220

2.2.5 Pc Personal

Dentro de la PC residirá la aplicación que controle el faro por lo que es necesario contar con ella. Los requerimientos mínimos del sistema son: procesador Intel Pentium 4, 512 MB de memoria RAM, puerto serial y tarjeta Ethernet 10/100 Mbps.

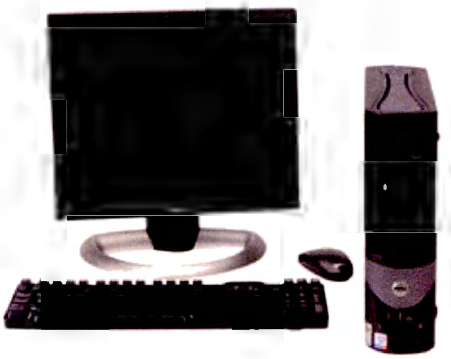


Figura 2.5 Pc personal

2.3 Esquema del proyecto

En las instalaciones del centro de control.

Figura 2.6 Instalaciones del centro de control

En las instalaciones del faro.

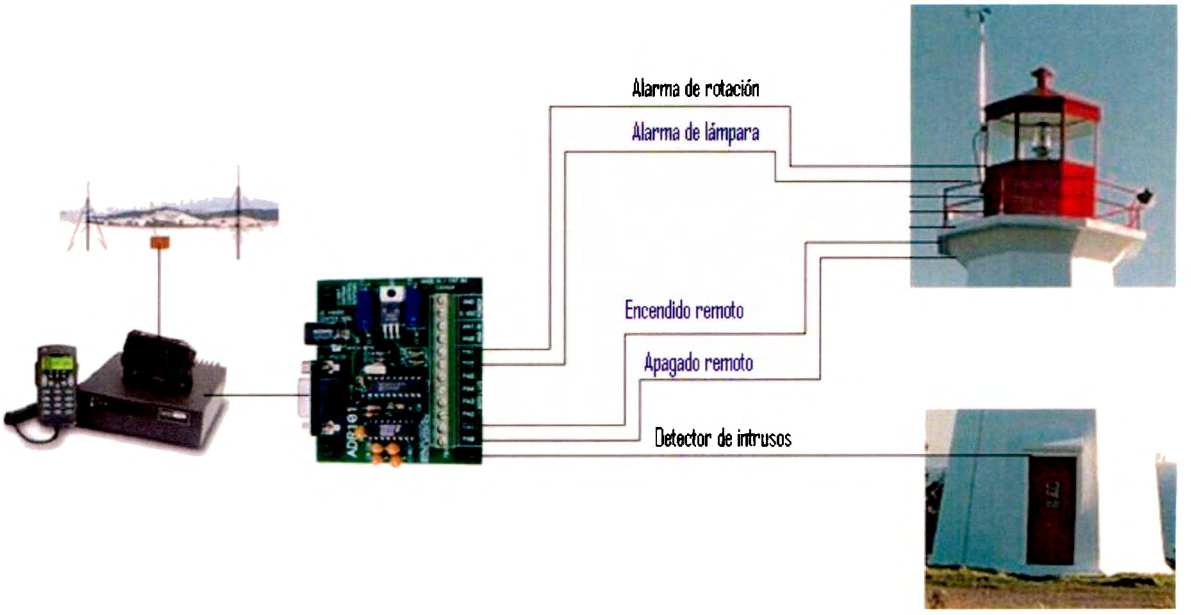
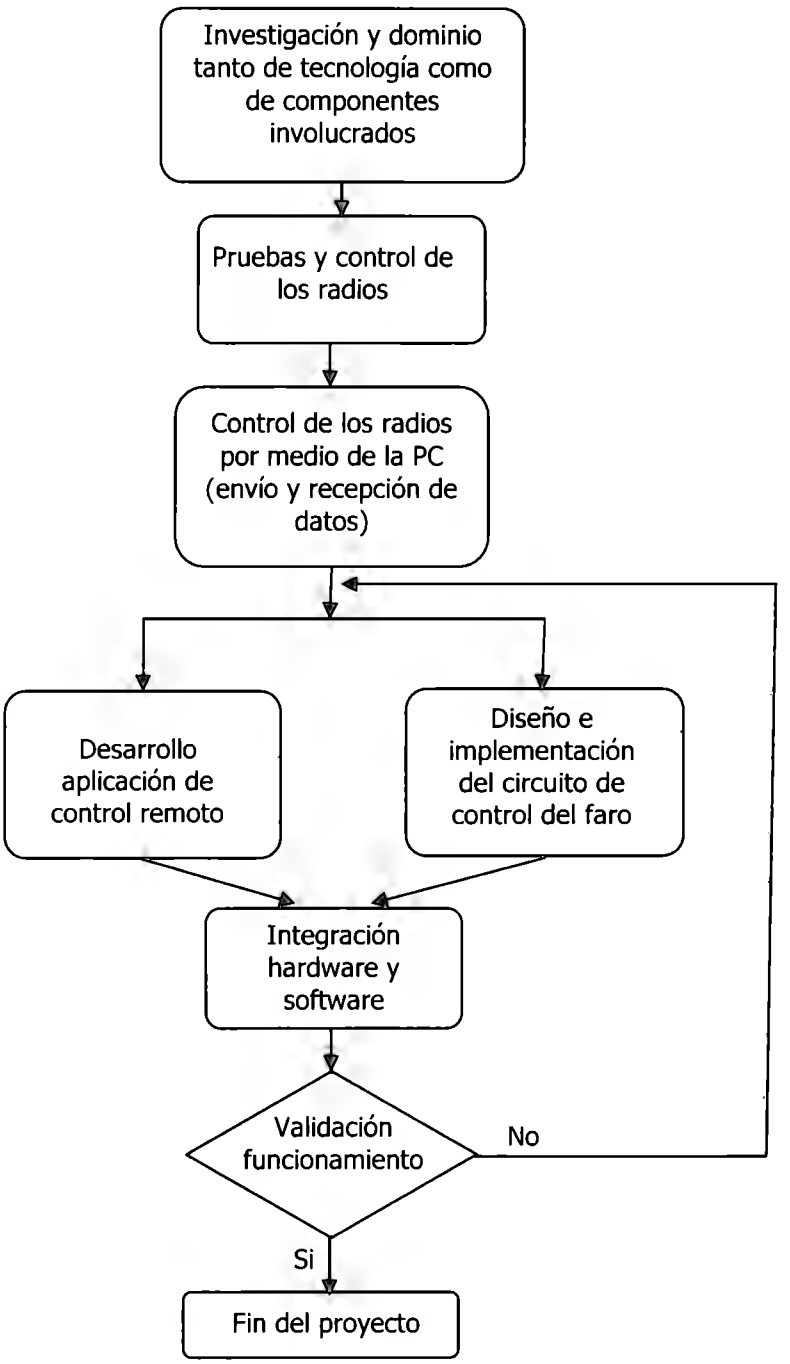


Figura 2.7 Instalaciones del faro

2.4 Partes del Proyecto

El proyecto consta de distintas fases que se ilustran a continuación con un diagrama de bloques.



3. MARCO TEÓRICO

3.1 Antecedentes del uso de HF

Por muchos años, numerosas organizaciones han estado empleando el espectro de Alta Frecuencia (HF) para comunicarse sobre grandes distancias. A finales de los años 30´s, fue reconocido que estos sistemas de comunicación eran sujetos de marcar variaciones en el rendimiento, y estas variaciones le fueron atribuidas de manera hipotética a los cambios de la ionosfera. Un esfuerzo considerable fue hecho por los Estados Unidos, así como también en otros países, para investigar los parámetros ionosféricas y poder determinar sus efectos en las ondas de radio y la confiabilidad asociada a los circuitos de HF [14].

El programa de Ionospheric Communications Enhanced Profile Analysis and Circuit (ICEPAC) es un modelo completo para los circuitos de alta frecuencia de las radiocomunicaciones (HF) en el rango de 2 a 30 MHz. ICEPAC se describe como una extensión del programa Communications Analysis and Prediction (IONCAP). El IONCAP, desarrollado por ITS (Institute for Telecommunication Sciences) y sus organizaciones predecesoras, se convirtió en uno de los modelos para predicciones de propagaciones de HF más aceptados y utilizados. Sin embargo, IONCAP demostró un rendimiento pobre en la región polar, y utilizó algunos de los viejos perfiles de estructuras de densidad de electrón. Para corregir estos problemas, IONCAP fue transformado en ICEPAC agregando el modelo de perfil Conductividad Ionosférica y Densidad de Electrón (ICED), descrito en Tascione (1987). Difiere en la estructura de la región polar de la ionosfera y en la estructura de la latitud baja y media de la ionosfera. El modelo reconoce los diferentes procesos físicos que existen en las diferentes regiones de la ionosfera. ICEPAC contiene distintos algoritmos para el canal sub-auroral, la porción ecuatorial de la zona auroral, la región polar de la zona auroral y para el casco polar [15].

Existe un programa de investigación llamado High Frequency Active Auroral Research Program (HAARP), es una investigación financiada por la Fuerza Aérea de los Estados Unidos, la Marina y la Universidad de Alaska, que se dedica a investigar la identificación y caracterización de los procesos que se pueden iniciar en la atmósfera, la ionosfera y el espacio vía interacciones con las ondas de radio de la alta potencia, que podrían cambiar el funcionamiento de las comunicaciones y sistemas de vigilancia. Se inició en 1993 para una serie de experimentos durante veinte años [16].

Figura 3.1 Vista de las instalaciones de HAARP desde el aire, en las inmediaciones del monte Sanford (Alaska) [3]

Entre los procesos se encuentran: la aceleración del electrón, incluyendo la producción de emisiones (IR) ópticas e infrarrojas; la generación, el mantenimiento y/o la supresión de las estructuras de la ionización alineadas a lo largo del campo magnético de la tierra; la modulación de corrientes en la ionosfera, produciendo los AMI (Artificial Ionospheric Mirrors) en espacio para generar las ondas de radio ULF(Ultra Low Frequency)/ELF (Extremely Low Frequency)/VLF (Very Low Frequency); y la producción de emisiones electromagnéticas estimulantes, tal y como se observa en la figura 3.2. [16].

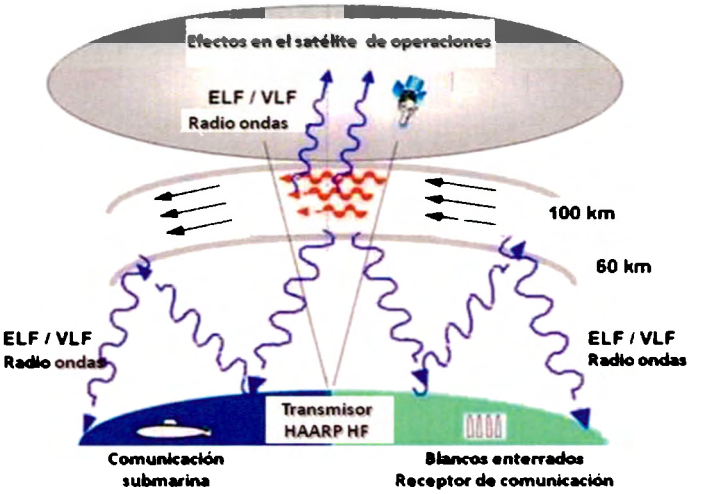


Figura 3.2 Diagrama de HAARP con su transmisor de HF [4]

[3] Disponible en: http://www.haarp.alaska.edu
[4] Disponible en: http://www.haarp.alaska.edu

Otros objetivos del programa son la investigación experimental para determinar el potencial para explotar esta tecnología ionosférica para los nuevos usos del sistema de la onda de radio. En la figura 3.3 se muestran las expectativas que se tiene sobre el proyecto con respecto a su capacidad de transmisión [16].

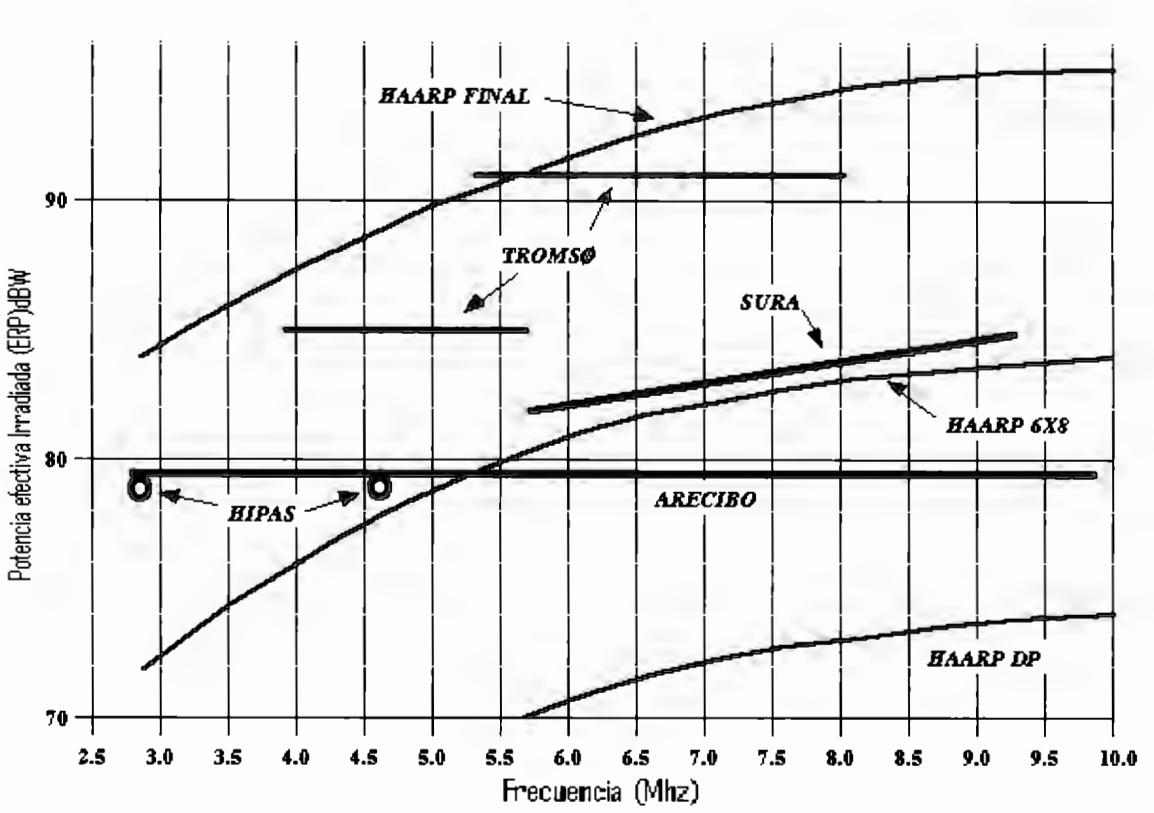


Figura 3.3 Comparación del HAARP con otras instalaciones ionosféricas [5]

3.2 Refracción de una onda de radio de HF en la ionosfera

El radar de SuperDARN cubre frecuencias entre 8 y 18 MHz que corresponden a las longitudes de onda a partir del 38 a 17 m. Estas longitudes de onda son bastante cortas como para que el medio ionosférico no cambie mucho en la distancia de algunas longitudes de onda y se puede por lo tanto aproximar por un medio estratificado del plano [17].

[5] Disponible en: http://www.haarp.alaska.edu

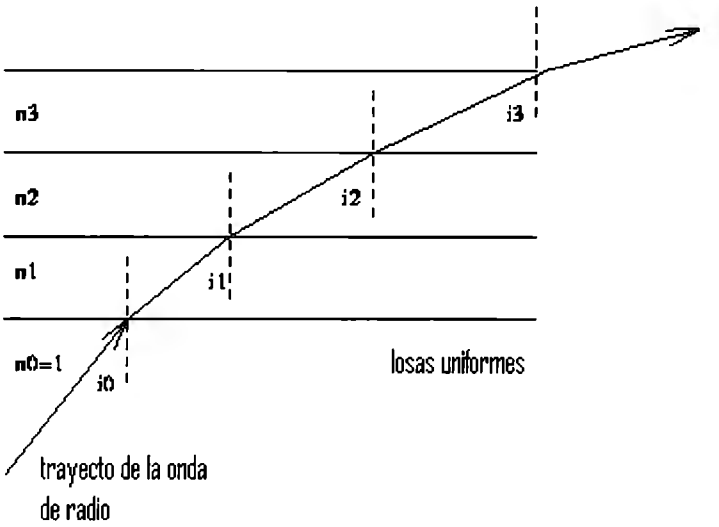


Figura 3.4 Refracción de una onda de radio en un medio estratificado del plano [6]

La ionosfera se considera como un apilado de losas finas con índices refractivos n_1, n_2, n_3. Dejar una onda plana ser incidente del espacio libre (con el índice de refracción n_c) y ángulo de incidencia i_0. La ley de Snell se puede entonces aplicar a cada límite, dando:

$$n_{r-1} \sin i_{r-1} = n_r \sin i_r \; ; \; r = 1, 2, \ldots \qquad (3.1)$$

Puesto que la frecuencia del plasma aumenta con la altura, n llega a ser más pequeña y el rayo se dobla gradualmente hacia el horizontal.

[6] Disponible en: http://www.voacap.com/itshfbc-help/icepac-tech-intro.html

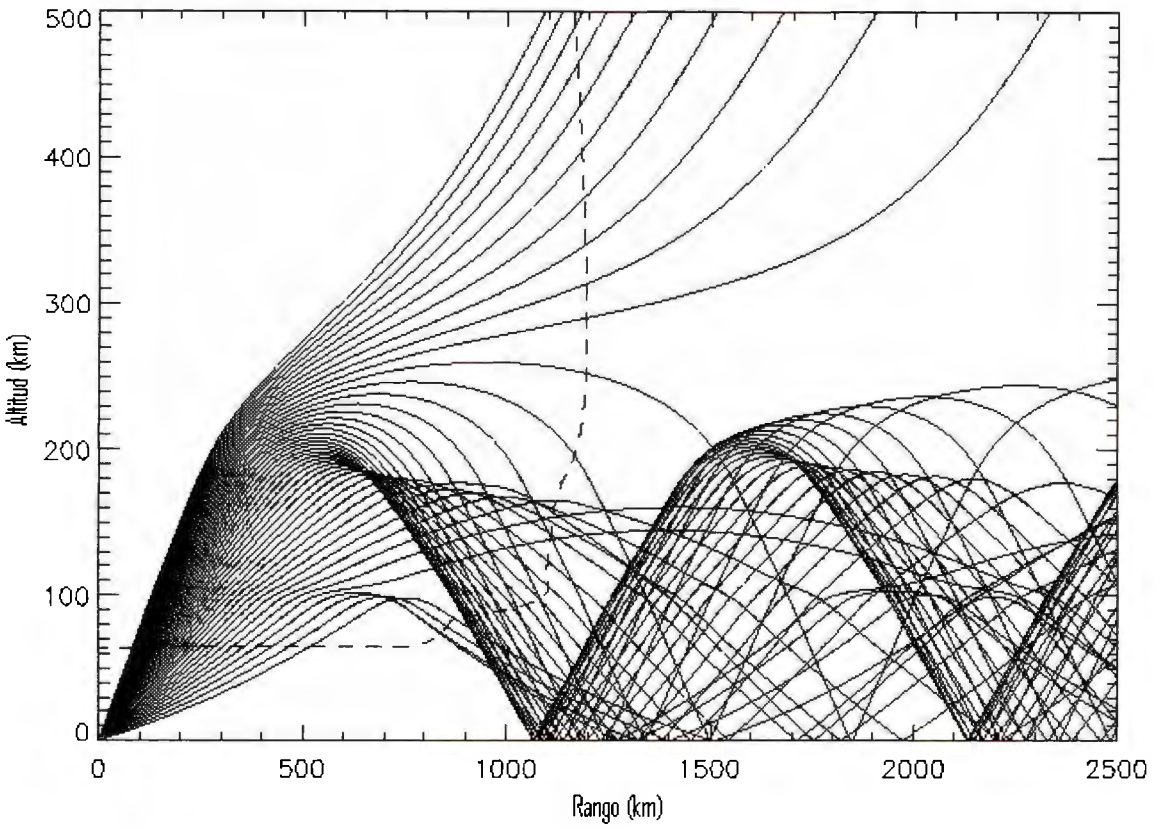


Figura 3.5 Vista de costado de las trayectorias de la propagación de los rayos SuperDARN irradia, ionosfera a una frecuencia de 12.45 MHz para los ángulos de elevación de 5 a 50 grados [7]

Algunas características importantes se pueden identificar en la figura 3.5. Los rayos con ángulos de elevación grandes se propagan a través de la ionosfera y nunca vuelven. Los rayos con ángulo de elevación pequeño pueden conseguir reflejarse en las alturas de la capa E o en las alturas de la capa F[8]; estas ondas después golpearán la tierra y se pueden dispersar a lo largo de la misma trayectoria al receptor [17].

[7] Disponible en: http://superdarn.jhuapl.edu/

[8] Para más detalle de dónde se encuentran estas capas ver la *figura 3.6. Capas de la Atmosfera.*

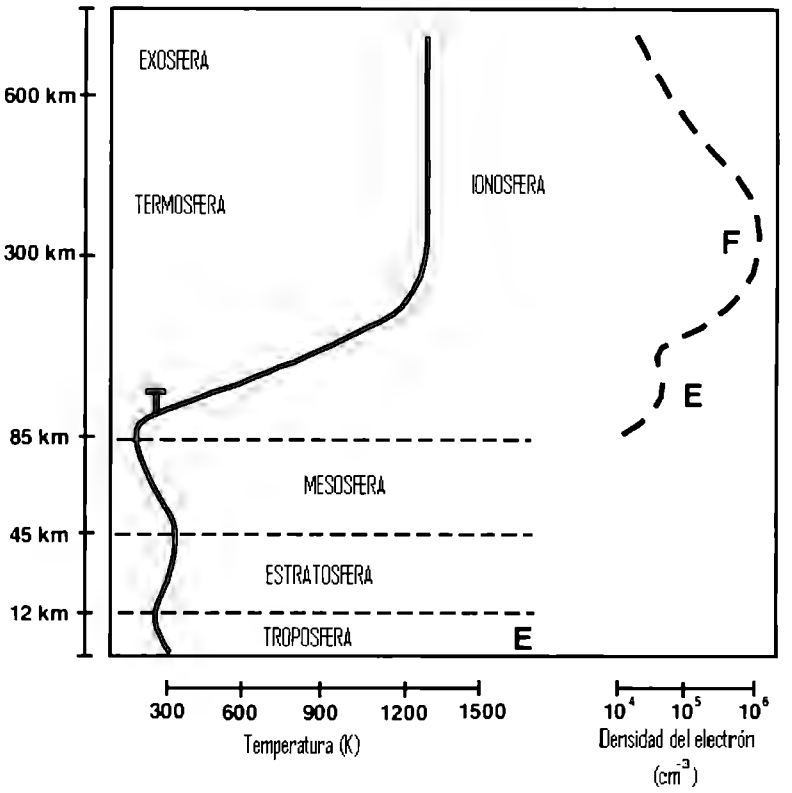


Figura 3.6 Capas de la Atmosfera [9]

Estas ondas entonces se llaman "groundscatter". Alternativamente, estos rayos se pueden reflejar del respaldo de la tierra a la ionosfera, tal y como se muestra en la figura 3.7, y la dispersión de las estructuras ionosféricas puede entonces ocurrir. Las longitudes de trayectoria para éstos modos del "salto" pueden variar de la propagación "straight-line" nominal del transmisor al "scatterer". Puesto que se asume generalmente que la gama medida (retraso) se puede convertir a la posición del blanco en base de la propagación "straight-line", la distancia al blanco se sobrestima generalmente [18].

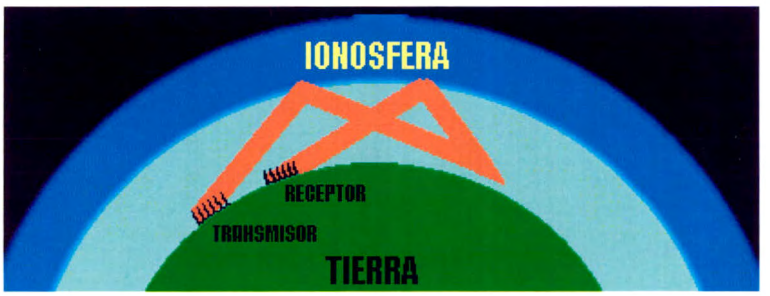


Figura 3.7 Refracción en la ionosfera [10]

[9] Disponible en: http://www.voacap.com/itshfbc-help/icepac-tech-intro.html
[10] Disponible en: http://www.voacap.com/itshfbc-help/icepac-tech-intro.html

3.3 Importancia de la transmisión de datos por medio de la ionosfera con HF

Ahora se vuelve necesario examinar el medio ionosférico más detenidamente y de este estudio obtener la información necesaria para los usuarios de la transmisión de datos por medio de HF [18].

Aparentemente el 10 por ciento de los circuitos tienen rangos de error, esto es el presente alcanzable en HF, con una tasa de transmisión del doble del máximo de aquellos empleados en las líneas "switched-wire", esto nos da un punto de comparación para ver qué tan eficiente es la transmisión HF ionosférica [19].

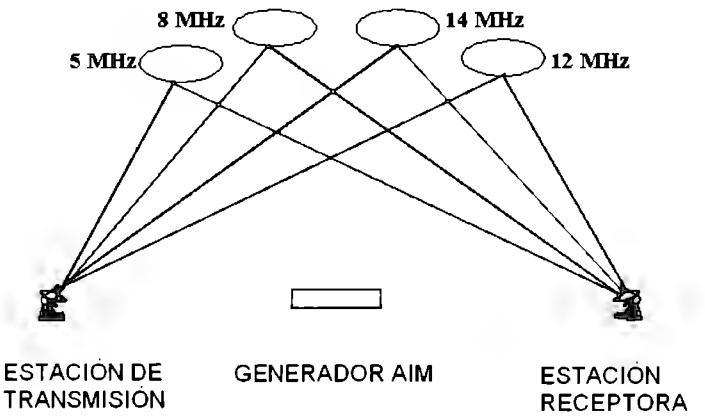


Figura 3.8 Transmisión de datos por medio de la ionosfera con HF [11]

El criterio de medición de errores para la transmisión por HF es diferente al comúnmente utilizado, ya que no se utiliza un sistema de bits sino uno basado en caracteres.

Se cree que en los próximos años se podrá mejorar de manera notable la transmisión en HF por medio de la ionosfera por medio de dos técnicas llamadas "channel matching" y "coding" [18].

"Channel matching" está poco avanzada y se ha utilizado casi desde que empezó la transmisión en HF por medio de la ionosfera, por lo cual se piensa que se puede mejorar en este rubro [19].

[11] Disponible en: http://www.voacap.com/itshfbc-help/icepac-tech-intro.html

3.4 Ventajas y Desventajas de HF

El medio de comunicación en HF presenta las siguientes ventajas:

- Independiente de una infraestructura
- Comunicación de punto a punto o multipunto
- Utilizando la ionosfera como reflector
- Alcance de varios miles de kilómetros
- Disponible sin costo
- Independiente de la topografía [20]

La principal desventaja que presenta el medio de comunicación en HF es el ruido atmosférico al cual todos los radios de HF son susceptibles, puede generar dificultades serias para comunicarse, lo cual no permite un enlace confiable. El resultado de este ruido atmosférico genera la dificultad para entender los mensajes. El ruido atmosférico en la mayoría de los casos, son picos más rápidos que la voz y se detectan con el análisis de Fourier [21].

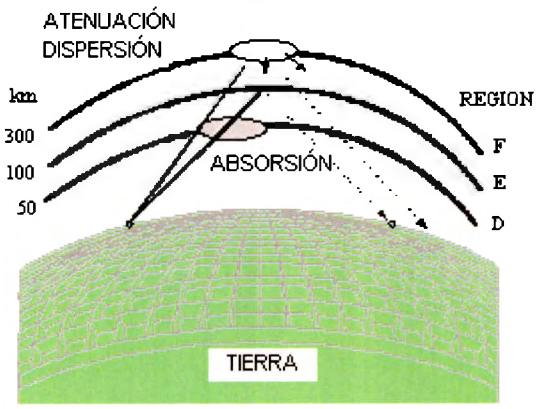


Figura 3.9 Problemas que presenta HF [12]

3.5 ¿Por qué HF?

En la tabla 3.1 se observa una comparación de HF con otras tecnologías y tomando en cuenta las variables que en esta aparecen podemos deducir que la mejor opción es la de transmisión por medio de HF.

[12] Disponible en: http://www.voacap.com/itshfbc-help/icepac-tech-intro.html

TECNOLOGÍA	HF	COMUNICACIÓN SATELITAL.	MICROONDAS
MOVILIDAD	SI	SI	NO
COSTO	Bajo, de 20,000 a 200,000 pesos, sin costo de renta mensual.	Elevado, (hasta de millones de pesos en los equipos de comunicación y cientos de miles de pesos en rentas de enlaces).	Bajo, de 20,000 a 500,000 pesos, sin costo de renta mensual.
ANCHO DE BANDA.	Reducido, rango de frecuencias de 3 a 30 MHz.	Elevado, rango de frecuencias de 200 MHz (Banda P) a 31 GHz (Banda K).	Elevado rango de frecuencias usualmente definido de 1 a 100 GHz pero con aplicaciones comunes entre 1 y 40 GHz.
COMUNICACIÓN GRUPAL	SI	SI	NO
DISTANCIA	Miles de kilómetros	Miles de kilómetros	Decenas de kilómetros.

Tabla 3.1 Comparación de HF con otras tecnologías

Como podemos observar en la tabla 3.1 el costo de HF oscila entre los 20,000 pesos a los 200,000 pesos, sin costo de renta mensual. Esta diferencia depende única y exclusivamente de los equipos de HF que se compren. Existen diversos equipos en el mercado; el costo más bajo es adquiriendo unidades de la marca SISCOM de HF, y el costo más alto es utilizando el Manpack de la marca Codan. Cabe señalar que éste último ya cuenta con antena para alcanzar distancias de miles de kilómetros, además de tener una autonomía de 48 horas sin alimentación eléctrica. Es importante señalar que estos costos son por unidad de HF.

3.6 Radios HF Codan NGT SR transceiver de nueva generación

El radio Codan NGT SR transceiver de nueva generación es un radio que incorpora además de las funciones usuales de comunicación HF, funciones avanzadas de establecimiento de llamada. También tiene la capacidad de incorporar: encripción de voz, GPS, fax y envío de datos e e-mail, mediante la incorporación de algunos accesorios [20].

3.6.1 Características principales de los radios Codan NGT SR transceiver de nueva generación

- Llamada de emergencia: el NGT SR incorpora una función de llamada de emergencia que permite enviar automáticamente una señal a las estaciones seleccionadas.
- Fácil instalación: El NGT SR fue diseñado para su fácil instalación en ambientes fijos o en movimiento.
- Monitoreo Inteligente: una variedad de canales pueden ser monitoreados mientras el radio no está en uso para recibir llamadas a través de los mismos.
- Protección: El NGT SR está protegido contra daños en la antena, sobrevoltaje, y cambios de polaridad [22].

3.6.2 Características avanzadas de los radios Codan NGT SR transceiver de nueva generación

- ALE (Automatic Link Establishment): Esta característica permite seleccionar el mejor canal para la transmisión de manera automática, esta misma característica permite interoperabilidad con otros equipos que cumplan con el Standard FED-STD-1045 ALE, además, como una característica adicional el radio cuenta con CALM (Codan Automated Link Management) esto permite:
    - Monitorear distintas redes al mismo tiempo a una velocidad de 8 canales por segundo, 3 veces más canales que los sistemas ALE.
    - Base de datos LQA (Link Quality Analysis) de 24 horas con una capacidad de almacenamiento 7 veces mayor que otros sistemas ALE, lo que permite elegir el mejor canal de transmisión a cualquier hora desde que el radio es encendido.
    - Mecanismo Listen-Before-Transmit que evita la interferencia de actividad ALE en llamadas de voz.
- Easitalk: El NGT SR utiliza técnicas de procesamiento digital de señales, en las señales de voz recibidas para minimizar la interferencia y reducir el ruido.
- BITE (Built-In Test Equipment): es un proceso que puede ser iniciado por el usuario, para hacer pruebas y enviar reportes acerca del funcionamiento del sistema.

- Facilidades de llamada: El NGT SR permite realizar llamadas selectivas, llamadas telefónicas y envío de mensajes [23].

3.6.3 Ventajas del sistema CALM

El sistema CALM proporciona las siguientes ventajas:

- Selecciona automáticamente la frecuencia óptima.
- Puede ser utilizado para comunicación de voz, datos y correo electrónico.
- Cuenta con un sistema de seguridad basado en la encriptación profunda.
- Genera múltiples registros sellados con hora y fecha en una Link Quality Analysis (LQA), base de datos avanzada.
- Provee una red de auto administración.
- Conoce y predice la evolución de la propagación durante el día.
- Adquisición de datos múltiples para una predicción de cobertura muy confiable.
- Para cualquier momento y tipo de llamada, la base de datos (LQA) dispone de una variedad de componentes para seleccionar automáticamente el canal correcto y más eficiente para el usuario.
- Cualquier llamada renueva la base de datos en todos los radios, es decir, con la primera llamada de la nueva estación, los radios existentes agregan el nuevo usuario a su base de datos [22].

3.6.4 Comunicación utilizando CALM

La comunicación utilizando CALM se realiza de la siguiente manera:

1. Todos los radios están escaneando sus juegos de frecuencias.

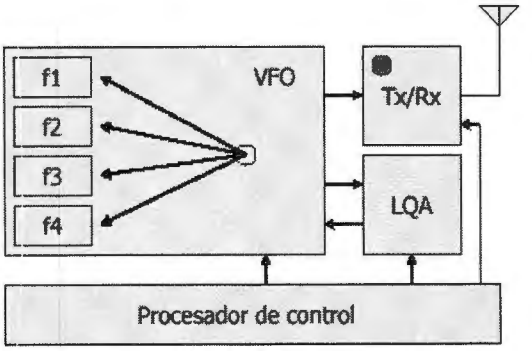


Figura 3.10 Escaneo de las frecuencias

2. Uno de los usuarios genera una llamada y el procesador manda el comando a la LQA.

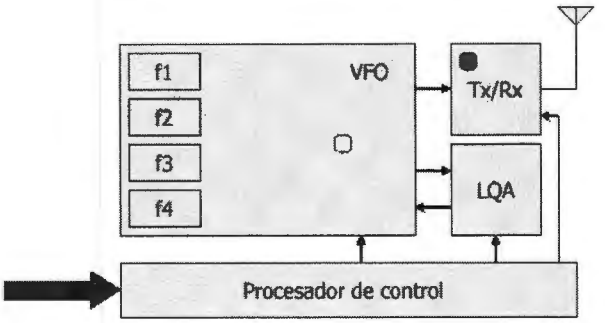


Figura 3.11 Generación de llamada

3. La LQA determina la frecuencia más probable para enlazar los radios. El dato se pasa al VFO (Variable Frequency Oscillator) para sintonizar la frecuencia determinada.

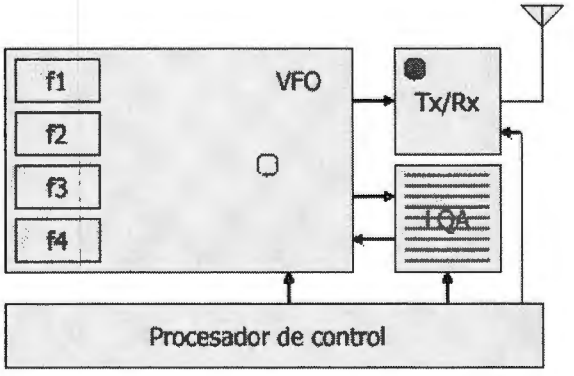


Figura 3.12 Determinación de la frecuencia

4. VFO sintoniza la frecuencia determinada por la LQA (f3). El transmisor emite la llamada en FSK (Frequency Shift Keying) de 8 tonos.

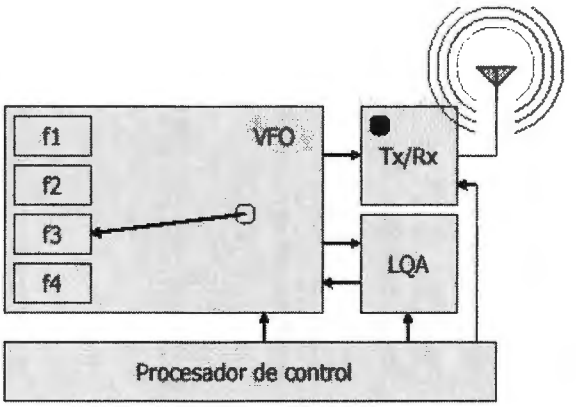


Figura 3.13 Emisión de la llamada en FSK de 8 tonos

5. Todos los radios están escaneando sus juegos de frecuencias. En f3 se recibe FSK de 8 tonos y el VFO amarra en esta frecuencia.

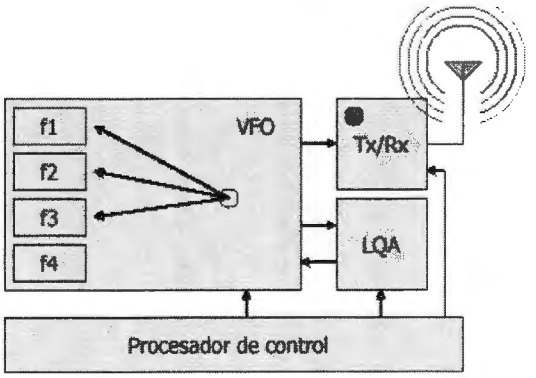


Figura 3.14 El VOF obtiene la frecuencia óptima

6. Se verifica el direccionamiento de la red y la identificación del usuario. Sigue una serie de acuerdos mutuos y se avisa el usuario audible.

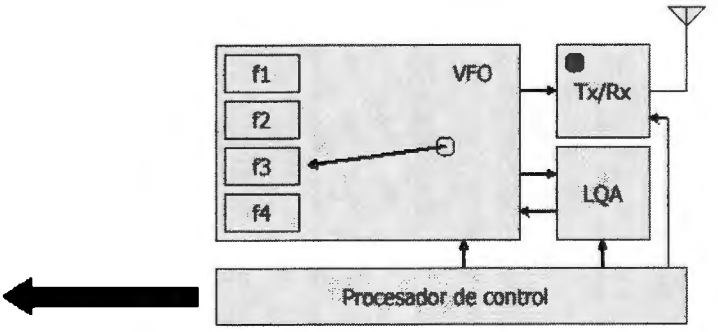


Figura 3.15 Verificación y confirmación de la conexión

7. Se comunican los radios de la manera conocida. Posteriormente se cuelga la llamada.

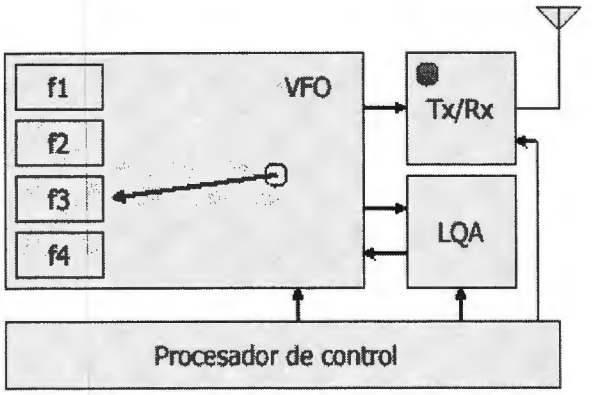


Figura 3.16 Comunicación de los radios

8. Todos los radios están escaneando otra vez sus juegos de frecuencias.

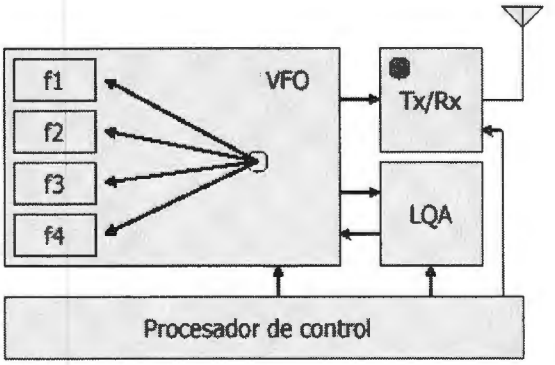


Figura 3.17 Escaneo de las frecuencias

3.7 ¿Por qué los radios Codan NGT SR transceiver de nueva generación?

En la tabla 3.2 se muestra una comparación entre distintas marcas de radios HF, basada en cinco principales aspectos que son: rango de frecuencias, número de canales, potencia de transmisión, modo de operación y costo del equipo, y tomando en cuenta todas estas variables podemos decir que en la relación calidad/costo la mejor opción es la de la marca Codan.

MODELO	NGT SR TRANSCEIVER	VX-1700	IC-M700PRO	TK-80
MARCA	CODAN	Vertex	ICOM	KENWOOD
RANGO DE FRECUENCIAS	Rx 250 KHz a 30 MHz Tx 1.6 MHz a 30MHz	Rx 30 KHz a 30 MHz Tx 1.6 MHz a 30 MHz	Rx 500 KHz a 30 MHz Tx 1.6 MHz a 30 MHz	Rx 500 KHz a 30 MHz Tx 1.8 MHz a 30 MHz
NÚMERO DE CANALES	400	200		80
POTENCIA DE TRANSMISIÓN	125 Watts	125 Watts	150 Watts	100 Watts
MODO DE OPERACIÓN	SSB (JBE, USB, LSB)	SSB (USB, LSB)	SSB (USB, LSB, CW)	SSB (USB, LSB, CW)
COSTO	1,500 dlls	1,500 dlls	2,100 dlls	1,600 dlls
IMAGEN				

Tabla 3.2 Comparación entre distintas marcas de radios HF

Como se puede apreciar en la tabla 3.2, los radios Codan modelo NGT SR transceiver de nueva generación son los que tienen mejor desempeño en la mayoría de los rubros antes mencionados como lo son en costo y numero de canales, lo cual para el desarrollo de nuestro proyecto es sumamente importante dadas las características del mismo.

4. DESARROLLO Y RESULTADOS

4.1 Primera etapa

En esta primera etapa nos enfocamos a realizar pruebas de interconexión entre los radios Codan NGT SR transceiver de nueva generación, sin emplear la computadora. Es decir, tuvimos que familiarizarnos con el funcionamiento de los mismos.

La primera prueba fue establecer una llamada de voz entre dos estaciones. Para ello, fue necesario identificar las estaciones con un nombre para que se reconocieran dentro de la red.

Los nombres establecidos fueron "STATION1" y "STATION2". No fue necesario asignar el canal de comunicación, debido a que los radios automáticamente seleccionan el canal más adecuado dependiendo de la hora en la que se realiza el enlace.

a) b)

Figura 4.1 a) STATION1 b) STATION2

La segunda prueba consistió en el envío de un mensaje de texto entre estas dos estaciones. Prueba que resultó sin mayores problemas, una vez configuradas las estaciones.

Por último realizamos pruebas de comunicación grupal con la función ALLCALL, llamadas de emergencia entre las estaciones y solicitud de estatus de una estación a otra.

Como sabemos el alcance de los radios depende de las condiciones de la ionosfera que se den a lo largo del día, a su vez éste depende también del tipo de antena que se utilice en los radios. Usualmente los radios utilizan 3 tipos de antenas que son: antena dipolo de cable, antena de cable largo y antena dipolo de banda ancha. Para las pruebas de comunicación que realizamos, utilizamos una antena de cable largo, esta antena opera en un rango de frecuencias de 1.6 a 30 MHz y nos permite transmitir hasta un máximo de 500 Km.

Comenzamos por programar los radios para que operaran en el rango de frecuencias deseado (3-30 MHz) y les asignamos el nombre de "STATION1" y "STATION2" respectivamente, para así proceder a realizar llamadas de voz, para ello se puede seleccionar el modo de operación automático, en el que el radio selecciona la frecuencia a la que va a transmitir de acuerdo con una base de datos que le dice cuál es la frecuencia de transmisión

más adecuada dependiendo de la hora a la que se desea establecer la conexión, o el modo de operación manual en el que el usuario establece la frecuencia, por otra parte se puede activar un mecanismo de disminución de ruido conocido como "Easytalk" que utiliza algoritmos de procesamiento digital de señales para ese fin. Cabe señalar que para poder establecer cualquier tipo de enlace entre los radios, se necesita que uno de los radios reconozca al otro y envíe paquetes de prueba, para ello utilizamos la función de "AllCall" que busca radios dentro de el alcance y los da de alta para poder comunicarse con ellos.

Después de las llamadas de voz utilizamos el envió de mensajes de texto, estos mensajes se envían mediante el teclado del radio asemejando a un teléfono celular, de igual forma el destino se debe definir de acuerdo al nombre que tiene asignado el radio con el que se desea establecer la comunicación en este caso "STATION1" o "STATION2".

Además de éstas, aprendimos a utilizar otras funciones como la de llamada de emergencia, el GPS y la función de verificación del radio, es decir una característica de los radios que permite obtener los parámetros de transmisión de otra estación mediante el envío de una solicitud.

Una vez realizadas las pruebas de comunicación con los radios, logramos controlar y llevar a cabo la comunicación de los mismos a través de una computadora. Para esto desarrollamos dos programas en lenguaje C#, que nos permitieran establecer la comunicación serial con los radios, lograr enviar los comandos que éstos reconocen y permitir la recepción de mensajes. Es importante mencionar que estos primeros dos programas son aplicaciones de consola y que el objetivo de desarrollarlos fue aprender el funcionamiento de la comunicación serial de los radios Codan NGT SR transceiver de nueva generación, para así dar paso al desarrollo de una interfaz gráfica.

En esta primera etapa del proyecto como ya se mencionó, se desarrollaron dos programas en lenguaje C#. Uno de ellos, llamado programa de escritura, nos permite únicamente enviar comandos vía puerto serial a uno de los radios (por ejemplo STATION1), para que éste se enlace directamente con el otro radio (por ejemplo STATION2) y así establecer una comunicación entre ambos radios.

Este programa a su vez es capaz de enviar mensajes de texto entre los radios utilizando el comando "pagecall" seguido del nombre del radio al que se quiere enviar y el mensaje a enviarse entre comillas. Por ejemplo: *pagecall STATION2 "mensaje de prueba"*. Este comando es

el que su utilizó para el desarrollo de todo el sistema de comunicaciones entre ambas partes del proyecto.

Las pruebas que se realizaron con este programa fueron el envío de tres comandos que identifican los radios. Estos comandos son:

- alecall
- hangup
- pagecalle

El comando "alecall" seguido del nombre del radio al que te quieres comunicar, te proporcionará un enlace de voz entre los radios, por ejemplo si escribes: **alecall STATION1** y lo envías por el puerto serial de la computadora al radio que está conectado a esta, en este caso STATION2, se establece una llamada de voz entre STATION1 y STATION2. En la Figura 4.2 se muestra la corrida de esta instrucción.

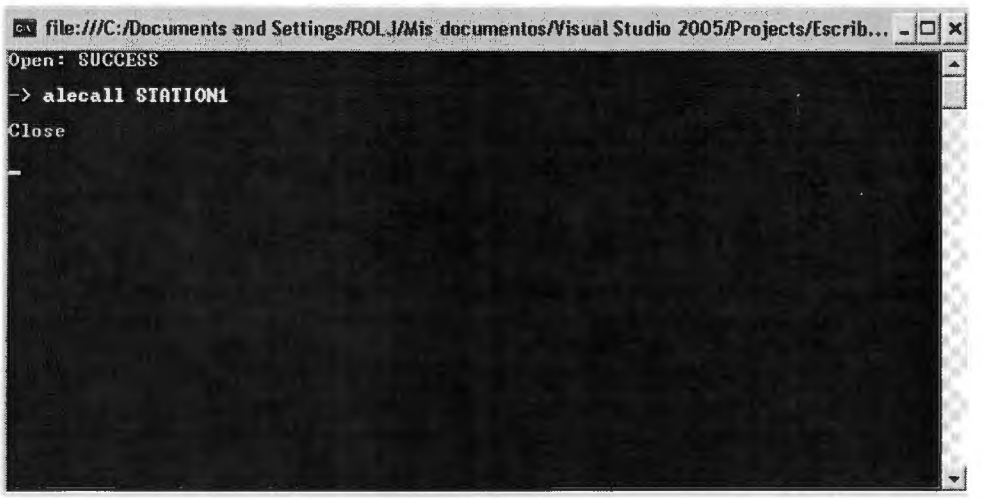


Figura 4.2 Corrida de la instrucción alecall

En la corrida de esta instrucción lo que hace el programa de escritura es abrir el puerto serial para transmitir a 9600 baudios de la computadora al radio un "string" concatenado que contiene el comando "alecall" seguido del nombre del radio al que se pretende llamar, en este caso "STATION1". Una vez que es transmitido vuelve a cerrar el puerto.

El comando "hangup" te permite terminar con el enlace de voz, por ejemplo si escribes: **hangup** y lo envías por el puerto serial de la computadora al radio que está conectado a esta, el radio inmediatamente va a terminar la llama de voz con el radio que este enlazado. En la Figura 4.3 se muestra la corrida de esta instrucción.

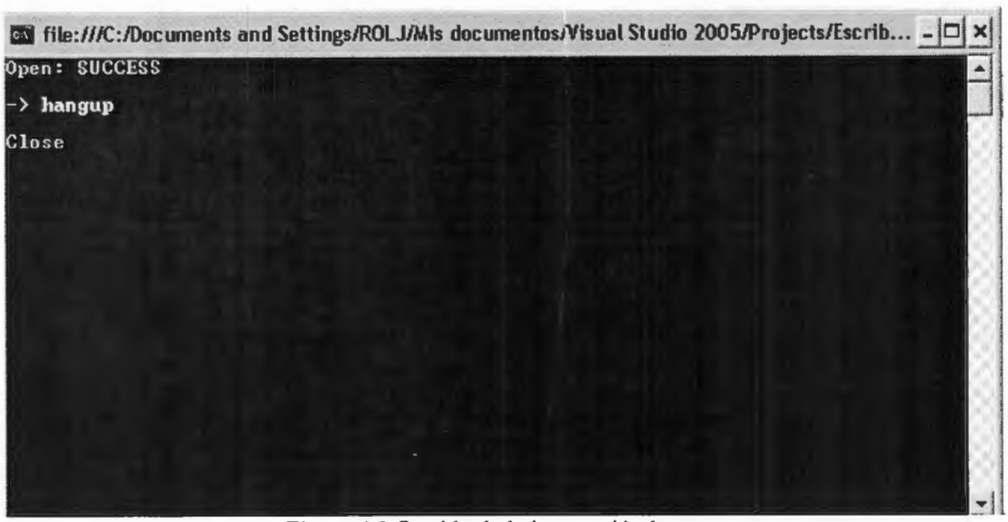


Figura 4.3 Corrida de la instrucción hangup

En la corrida de esta instrucción lo que hace el programa de escritura es abrir el puerto serial para transmitir a 9600 baudios de la computadora al radio un "string" que contiene el comando "hangup". Una vez que es transmitido vuelve a cerrar el puerto.

El comando "pagecall" seguido del nombre del radio al que te quieres comunicar y entre comillas un mensaje, te permite enviar un mensaje de un radio a otro, por ejemplo si escribes: **pagecall STATION1 "Cambiar Lampara"** y lo envías por el puerto serial de la computadora al radio que está conectado a esta, en este caso STATION2, se manda el mensaje *Cambiar Lampara* del radio STATION2 al radio STATION1. Este comando es el que se utiliza para llevar a cabo la comunicación entre las dos partes del proyecto, que son: el centro de control y las instalaciones del faro. En la figura 4.4 se muestra la corrida de esta instrucción.

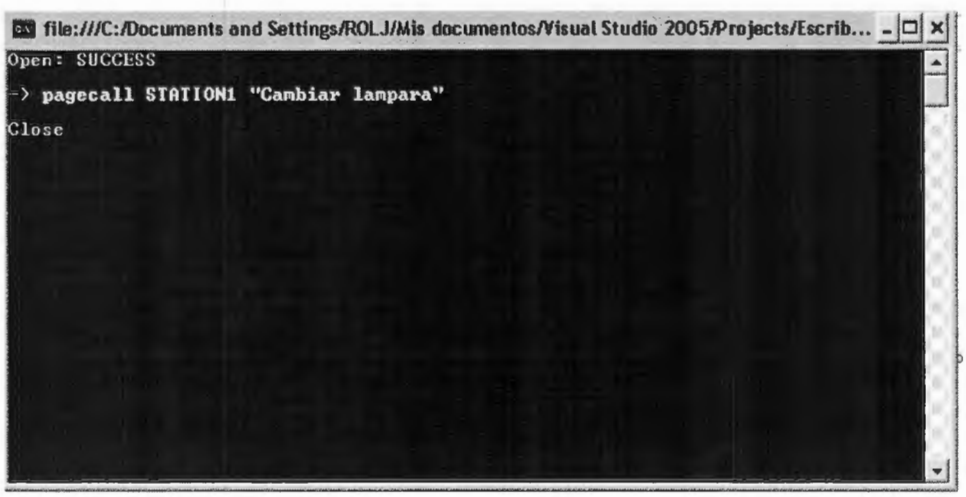


Figura 4.4 Corrida de la instrucción pagecalle

En la corrida de esta instrucción lo que hace el programa de escritura es abrir el puerto serial para transmitir a 9600 baudios de la computadora al radio un "string" concatenado que contiene el comando "pagecall" seguido del nombre del radio al que se pretende comunicarse, en este caso "STATION1" y un mensaje entre comillas. Una vez que es transmitido vuelve a cerrar el puerto.

El segundo programa, llamado programa de lectura, realizado en esta primera etapa del proyecto nos permite únicamente recibir información vía puerto serial del radio que esté conectado a la computadora, por ejemplo STATION1, esta información es enviada desde el otro radio, en este caso STATION2, en forma de mensaje de texto ya sea de forma manual (directamente del radio) o a través del programa de escritura antes descrito.

Las pruebas que se realizaron con este programa fueron la recepción de mensajes de texto enviados desde uno de los radios utilizando el programa de escritura. En la Figura 4.5 se observa la ejecución de una de las pruebas realizadas.

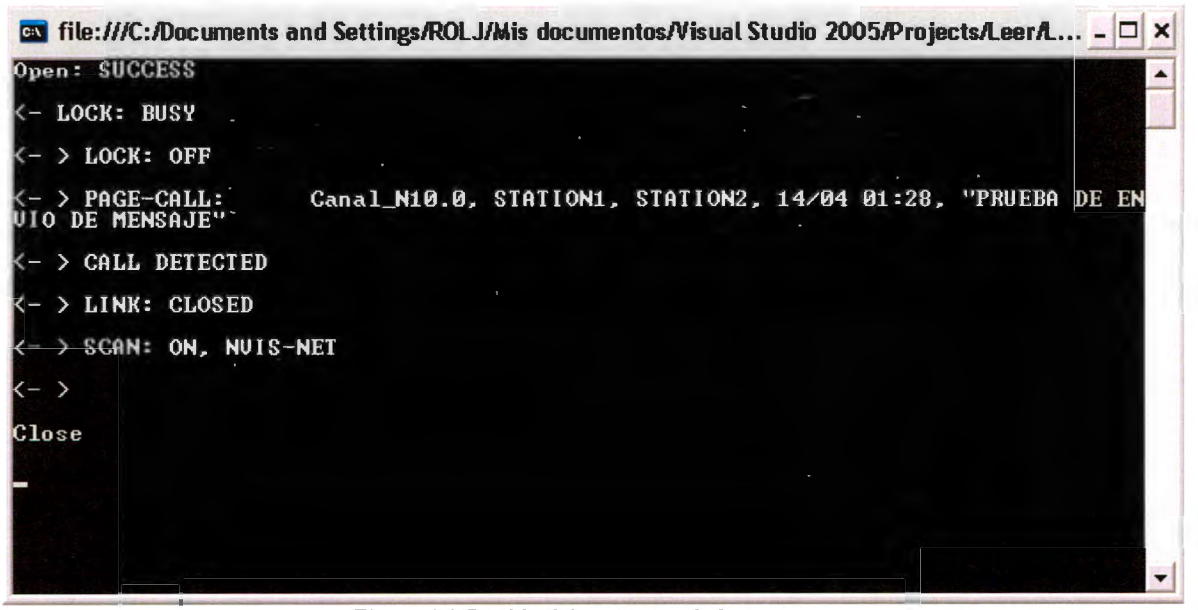


Figura 4.5 Corrida del programa de lectura

Al ejecutar el programa de lectura éste abre el puerto serial, una vez abierto el puerto serial el programa es capaz de recibir cualquier "string" proveniente del radio. Lo que se observa en la Figura 4.5 son los diferentes "strings" que se reciben en un mensaje de texto, dentro de estos "strings" se puede ver entre comillas el mensaje que se envió desde el otro radio, que en este caso es ***PRUEBA DE ENVIO DE MENSAJE***. También dentro de estos "strings" se puede

observar el canal que se utilizó para realizar la transmisión, el nombre del radio que está mandando el mensaje, el nombre del radio que está recibiendo el mensaje y la hora y la fecha de recepción del mensaje.

De forma gráfica los resultados obtenidos en esta primera etapa los podemos ver en la Figura 4.6, la cual muestra un diagrama a bloques del funcionamiento del sistema de comunicación.

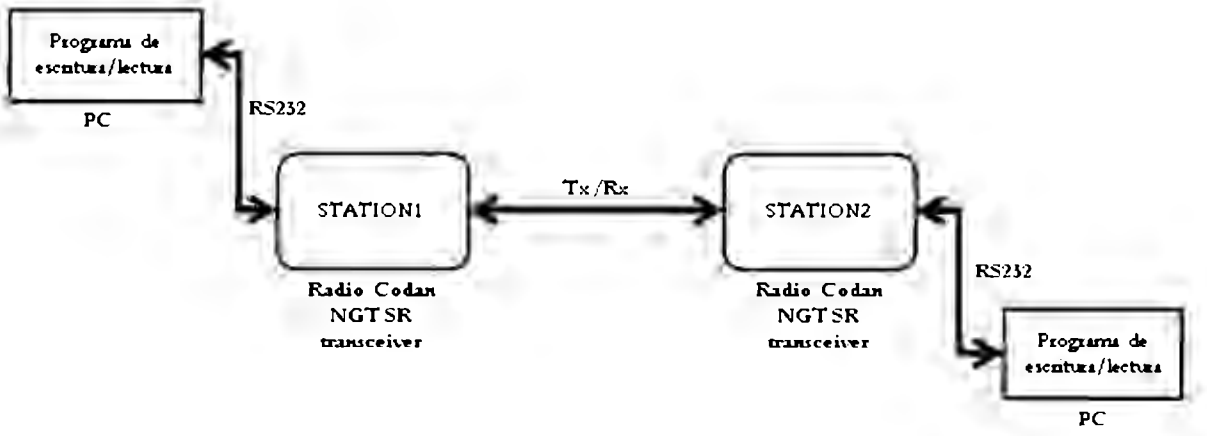


Figura 4.6 Diagrama a bloques de sistema

En la figura 4.6 se observa el proceso de transmisión y recepción de un mensaje de una computadora a otra utilizando como medio de transmisión los radios Codan NGT SR transceiver. El proceso que se describe es el siguiente: desde una computadora se corre el programa de escritura, este programa envía un mensaje al radio de nombre STATION1 a través del RS232, este radio se comunica con el radio de nombre STATION2 y este a su vez transmite al programa de lectura el mensaje que recibió proveniente del radio llamado STATION1, a través del RS232 y por último el programa de lectura despliega el mensaje recibido. Este proceso se da en ambos sentidos.

En esta primera etapa del proyecto los resultados de las pruebas fueron satisfactorios, se logró enviar y recibir datos de una computadora a otra utilizando como medio de transmisión los radios Codan NGT SR transceiver. Esto nos permite trabajar en una interfaz gráfica que nos permita manipular este sistema de comunicaciones de una manera más fácil y que sea entendible para cualquier usuario.

Es importante mencionar que los resultados obtenidos son muy importantes en el desarrollo del proyecto, ya que en el objetivo se plantea hacer un control y monitoreo a

distancia, y para ello, es de suma importancia lograr el envío y recepción de datos, utilizando como medio de transmisión los radios Codan NGT SR transceiver, para posteriormente poder trabajar en la parte de control del faro.

4.2 Segunda etapa

En esta segunda etapa del proyecto se trabajo en dos etapas principalmente que fueron: el desarrollo de la aplicación gráfica de control remoto y en el diseño e implementación del circuito de control del faro.

Desarrollo de la Aplicación Gráfica de Control Remoto.-

Esta parte del proyecto consta principalmente de tres bloques que son: identificación de usuario, tablero virtual y control del sistema de comunicaciones. En la Figura 4.7 podemos observar un diagrama a bloques de la aplicación gráfica a control remoto.

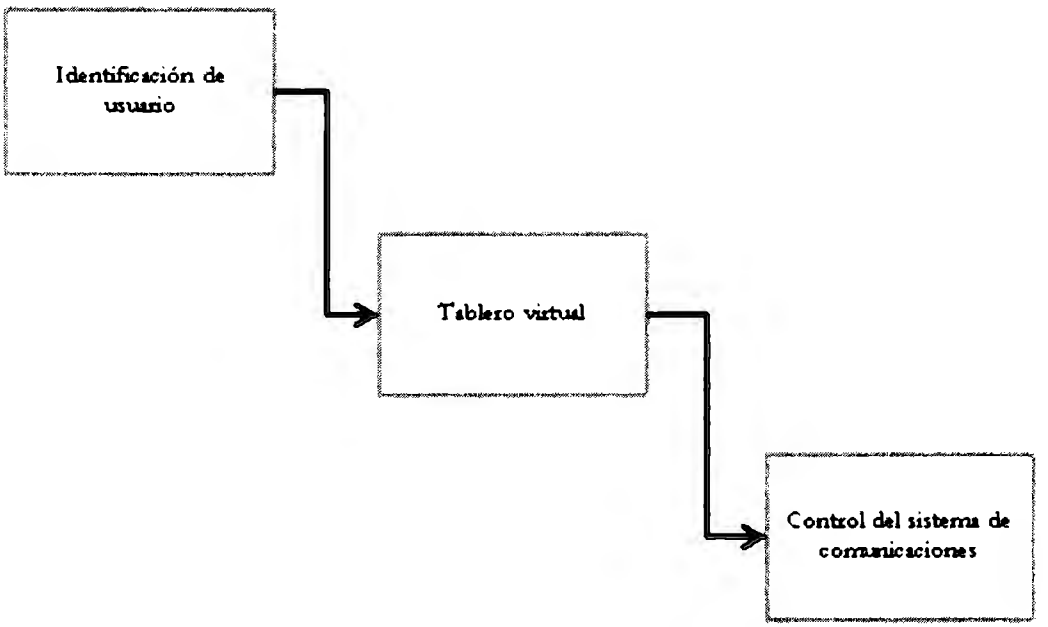


Figura 4.7 Diagrama a bloques de aplicación gráfica a control remoto

1.- Identificación de Usuario.-

La identificación de usuario se implementó como una estrategia de seguridad para que no todas las personas tengan accesos a la manipulación del sistema. Para desarrollar la identificación de usuario fue necesario utilizar una base de datos de *Microsoft Access*, que no es

más que una colección de datos clasificados y estructurados que son guardados en uno o varios ficheros pero referenciados como si de un único fichero se tratara [8].

Los datos de una base de datos relacional se almacenan en tablas lógicas relacionadas entre sí utilizando campos claves comunes. Por ejemplo para este caso se utilizaron tres campos que son: nombre, usuario y "password", estos datos son *columnas* que se agrupan en una *fila*. El conjunto de todas las *filas* forman una tabla de la base de datos, como podemos observar en la Figura 4.8 [8].

Nombre	Usuario	Pasword	Agregar nuevo campo
Alan Gómez	alan	america	
Javier Rodriguez	javier	aguilas	
Jorge Marcin	jorge	chivas	
Mauricio Orvañanos	mauricio	potros	
*			

Figura 4.8 Tabla de la base de datos

Como se puede observar en la Figura 4.8, una tabla es una colección de datos presentada en forma de una matriz bidimensional, donde las filas reciben también el nombre de *tuplas* o *registros* y las columnas de *campos*. *Microsoft Access* te permite realizar sobre una determinada base datos operaciones como insertar, recuperar, modificar y eliminar datos, así como añadir nuevas tablas o eliminar [8].

Para crear una base de datos en *Microsoft Access* se tienen que seguir los siguientes pasos:

- Abrir *Microsoft Access*.
- Ejecutar l orden *Nuevo*... y seleccionar *Base de datos en blanco* en el diálogo que se visualiza.
- Introducir el nombre de la base de datos. Pulsar el botón *Crear*.
- Añadir una nueva tabla. Seleccionar la página *Tablas* y hacer doble clic en "Crear una tabla en vista de diseño".
- Introducir el nombre, el tipo y las propiedades para cada uno de los campos de un registro. En este caso es lo que se muestra en la Figura 4.8.
- Requerir los tres primeros casos campos (propiedad *Requerido* = *si*).
- Para el campo *Password* establecer la propiedad *Indexado* al valor *sí* (*sin duplicados*).
- Definir *Password* como clave principal.

- Abrir la tabla e introducir los datos [8].

Una vez creada la base de datos es necesario vincularla con el programa de C# para tener el control de la identificación de usuarios completa. Para esto se necesita enlazar los controles que van a visualizar los datos de cada registro de la tabla con los campos respectivos del conjunto de datos, para ello hay que hacer lo siguiente en C#:

1. Situarse en el diseñador de formularios.
2. Seleccionar la caja de texto *ctUsuario* y su propiedad **DataBindings**; expandir este nodo y vincularlo con el campo *Usuario* de la tabla *Validación* del conjunto de datos *dsValidación* con la propiedad **Text** de *ctUsuario* [8].
   Esta operación añadirá a *Form1* un conjunto de datos *dsValidación* de tipo *dsValidación*, una adaptador *validaciónTableAdapter* para acceder a la tabla de *Validación* de la basa de datos y un componente *valodaciónBindingSource* de la clase **BindingSource** conectado al origen de datos *dsValidación* [8].
3. Hay que repetir el paso 2 para la caja de texto de *ctUsuario.*

La propiedad **DataBindings** da un control de acceso a la colección **Control-BindingsCollection** que permite almacenar los vínculos que mantiene ese control con los orígenes de datos desde los cuales quieren proveerse, lo que, a su vez, le permitirá interactuar de forma directa con ellos [8].

Por ejemplo, cuando en el punto 2 anterior establecimos un vínculo entre la propiedad **Text** de *ctUsuario* y la columna *Usuario* de la tabla *Validación* del **DataSet**, el asistente de C# añade la siguiente línea de código:

```
ctUsuario.DataBindings.Add("Text", dsValidacion, "validacion.usuario")
```

Esta sentencia añade a la colección **ControlBindingsCollection** de *ctUsuario* un vínculo entre su propiedad **Text** y la columna *Usuario* de la tabla *Validación* del **DataSet**. Con esto logramos tener una vinculación completa de la base de datos de *Microsoft Access* con el programa de C#.

2.- Tablero Virtual.-

El tablero virtual se implementó como una solución al monitoreo de los faros, el cual te permite seleccionar un faro entre 6 diferentes y trabaja con él. Para desarrollar el tablero virtual

también se creó una base de datos en *Microsoft Access*, llamada *Faros*. Esta base de datos se creó de la misma manera que la base de datos de la parte de identificación de usuario, con la variante que se le cambiaron los campos utilizados. En la figura 4.9 s observa la base de datos de los faros.

Faro	Estado	Ciudad	Código	Coordenada	Observaciones	Agregar nuevo campo
Isla Mujeres	Quintana Roo	Cancun	1	23.45.125	OK	
Pie de la Cuesta	Guerrero	Acapulco	2	143.54.102	Revision cercana	
Tulum	Quintana Roo	Playa del Carmen	3	25.48.98	OK	
Nuevo Vallarta	Jalisco	Puerto Vallarta	4	123.45.3	OK	
Calica	Quintana Roo	Puerto Morelos	5	23.49.97	En Revision	
Laguna	Zihuatanejo	Ixtapa	6	159.34.98	OK	

Figura 4.9 Base de datos de los faros

Una vez creada esta nueva base de datos, hay que realizar el mismo procedimiento para vincular esta nueva base de datos con el programa en C#, la diferencia que tiene esta nueva Forma de Windows con la anterior es que añade dos opciones más para el usuario que son: un control de desplazamiento y la opción avanzada de buscar datos.

El control de desplazamiento consiste en cuatro botones que te permiten navegar por la base de datos *Primero, Anterior, Siguiente* y *Último* y una etiqueta para mostrar el registro (1, 2, 3, 4, ...) que se está mostrando y el número total de registros. Cuando se llene el conjunto de datos el formulario mostrará en las cajas de texto los datos relativos al primer registro y la etiqueta *etPosicion* debe de indicar "1 de *n_regs*", siendo *n_regs* el total de registros, que en este caso son 6.

El control de desplazamiento funciona de la siguiente manera: el objeto **BindingSource** hace de puente entre el control y el conjunto de datos, proporcionando acceso a los datos actualmente mostrados por el control de una forma indirecta, incluyendo navegación, ordenación, filtrado y actualización. Este objeto se encargará de sincronizar los cuatro controles para que juntos muestren el faro, el estado, la ciudad, la estación, coordenadas y observaciones del registro que está en esa posición. La propiedad **Position** de **BindingSource** mantiene la posición del registro (fila de la tabla) actual; por lo tanto, para moverse por los registros de una tabla hay que utilizar esta propiedad. Así, para desplazarse al primer elemento hay que asignar a **Position** el valor de la propiedad **Count** menos uno, para ir al elemento siguiente al actual hay que sumar a **Position** uno, y para ir al elemento anterior al actual hay que restar a **Position** uno [8].

La opción avanzada para buscar datos, específicamente el nombre del faro, se añadió con el objetivo de encontrar más rápido un registro en una base de datos mucho más grande. El botón *Buscar* permitirá al usuario buscar un registro determinado a partir del actual, utilizando el método **Find** de **DataRowCollection** o el método **Select** de **DataTable**. El método **Select**, localiza y almacena en una matriz *filaBuscada* todas las filas que pasen el filtro de que el nombre del faro contenga una subcadena igual a la especificada por la variable *criterio*. Por ejemplo, se podría buscar el faro "Pie de la cuesta" por "Pie de", por "pie", etc [8].

3.- Control del Sistema de Comunicaciones.-

El control del sistema de comunicaciones es un ambiente gráfico que nos permite enviar y recibir mensajes utilizando como medio de trasmisión los radios Codan NGT SR transceiver de nueva generación. Además en ella puedes visualizar datos como: nombre del faro, estación, ubicación, coordenadas y estatus del faro.

El envío de mensajes utiliza tres botones que son: encendido remoto, apagado remoto y llamada de voz. El funcionamiento del botón de *encendido remoto* es el siguiente:

1. Primero se detecta que fue oprimido el botón de *encendido remoto.*
2. Entra al evento que tiene asignado este botón.
3. Dentro del evento, se declara una variable *Texto* de tipo "string" al cual se le concatena el comando "pagecall", el nombre del radio que es obtenido directamente de la variable *estación* que hereda de la forma *Faros* del tablero virtual, la letra *a* entre comillas para indicar que ese es el mensaje y por último una diagonal invertida seguida de la letra *r* (código ASCII de la tecla enter). En forma de código queda de la siguiente manera:

```
string Texto;
Texto = "pagecall "+ Faros.estacion + " \"a\"" + "\r";
```

4. Dentro del mismo evento, una vez declarado el "string" se utiliza el evento **Write** que recibe como parámetros la variable *Texto* y que es un evento del objeto **Serial**, el cual nos permite escribir este "string" en el puerto serial para que sea transmitido. En forma de código queda de la siguiente manera:

```
Serial.Write(Texto);
```

El puerto serial está configurado para que transmita a 9600 baudios por el puerto COM1. Tiene un buffer de escritura de 2048 y un buffer de lectura de 4096, no tiene paridad y no maneja ningún control de flujo.

El botón de *apagado remoto* funciona de la misma manera que el de encendido remoto, la única diferencia está en el paso 3 y es la letra que se envía entre comillas, para el apagado remoto es la letra *e*. El por qué se envían estas letras se detallan más adelante en este mismo documento.

El botón de *llamada de voz* funciona de manera similar a los botones de encendido y apagado remoto, la diferencia se encuentra en el paso 3. Veamos más a detalle el funcionamiento de este botón:

1. Primero se detecta que fue oprimido el botón de *llamada de voz*.
2. Entra al evento que tiene asignado este botón.
3. Dentro del evento, se declara una variable *Texto* de tipo "string" al cual se le concatena el comando "alecall", el nombre del radio que es obtenido directamente de la variable *estación* que hereda de la forma *Faros* del tablero virtual y por último una diagonal invertida seguida de la letra *r* (código ASCII de la tecla enter). En forma de código queda de la siguiente manera:

```
string Texto;
Texto = "alecall "+ Faros.estacion + "\r";
```

4. Dentro del mismo evento, una vez declarado el "string" se utiliza el evento **Write** que recibe como parámetros la variable *Texto* y que es un evento del objeto **Serial**, el cual nos permite escribir este "string" en el puerto serial para que sea transmitido. En forma de código queda de la siguiente manera:

```
Serial.Write(Texto);
```

La recepción de mensajes funciona de la siguiente manera:

1. El programa principal está leyendo constantemente el puerto serial, para ver si se presenta algún evento.
2. Cuando se presenta un evento de lectura, se declara una variable *cadena* de tipo "string" en la cual se va a guardar lo que se reciba del buffer de lectura del puerto serial a través del evento **ReadLine** que no recibe ningún parámetro y que es un evento del objeto **Serial**. En forma de código queda de la siguiente manera:

```
int cadena = Serial.ReadLine();
```

3. Dentro del mismo evento se hace un barrido con un ciclo "for" para buscar las letras *i*, *m* y *p* dentro de la variable *cadena* la cual recibe toda la trama en donde se encuentran estas letras que son el mensaje a identificar. Una vez encontrada la letra es guardada en una variable de tipo "string" llamada *letra*. En forma de código queda de la siguiente manera:

```
for (int i = 0; i < tamaño;i++ )
{
    if (cadena[i] == '"' && cadena[i + 1] != '\r')
    {
        letra = cadena[i+1];
    }
}
```

4. En el mismo evento se declara una estructura de tipo "switch" que recibe como parámetro la variable *letra*, esta variable es comparada en los "case" para ver a que letra corresponde y así cambiar el estatus del faro, por ejemplo si la letra que se recibió es una *i* el estatus del faro va a cambiar a *intruso detectado*. Esto se desarrollo con el objetivo de hacer más fácil el entendimiento del software para cualquier usuario. En forma de código queda de la siguiente manera:

```
switch (letra)
{

    case 'i':
        LbRecibidos.Text = " Estatus del Faro: Intruso
                           detectado";
        break;
    case 'm':
        LbRecibidos.Text = " Estatus del Faro: Foco fundido";
        break;
    case 'p':
        LbRecibidos.Text = " Estatus del Faro: Falla de
                           rotación";
        break;

}
```

A la aplicación gráfica del control del sistema de comunicaciones se le agregó un botón de *opciones avanzadas*, el cual le permite al usuario hacer pruebas directamente de envío de datos vía puerto serial y verificar la recepción de los datos, con el objetivo de revisar que efectivamente se están enviando y recibiendo mensajes entre las instalaciones del centro de control y las instalaciones del faro utilizando como medio de transmisión los Codan NGT SR transceiver de nueva generación.

De manera gráfica y en conjunto la aplicación gráfica de control remoto funciona de la siguiente manera:

1. Se despliega la ventana de identificación de usuario:

Figura 4.10 Identificación de usuario

2. El usuario ingresa un nombre de usuario válido y oprime el botón *Buscar Usuario.*
3. Una vez validado el usuario ingresa la "password" y presiona el botón *Validar.*
4. Si la "password" corresponde con el usuario, se despliega el tablero virtual para que lleves a cabo la selección del faro a operar.

Figura 4.11Tablero virtual

5. Si el usuario desea hacer una búsqueda especifica de algún faro oprime el botón de *Opciones avanzadas* para realizarla la búsqueda, aquí el usuario proporcionara el nombre completo o parcial del faro que está buscando.

Figura 4.12 Búsqueda de faro

6. Después de validar los datos del faro a operar, el usuario ya puede realizar las funciones programadas de este software.

Figura 4.13 Control del sistema de comunicaciones

7. Si el usuario desea verificar la recepción de datos o realizar algún envío manual de datos, el software le permite hacerlo al oprimir el botón de *Opciones avanzadas.*

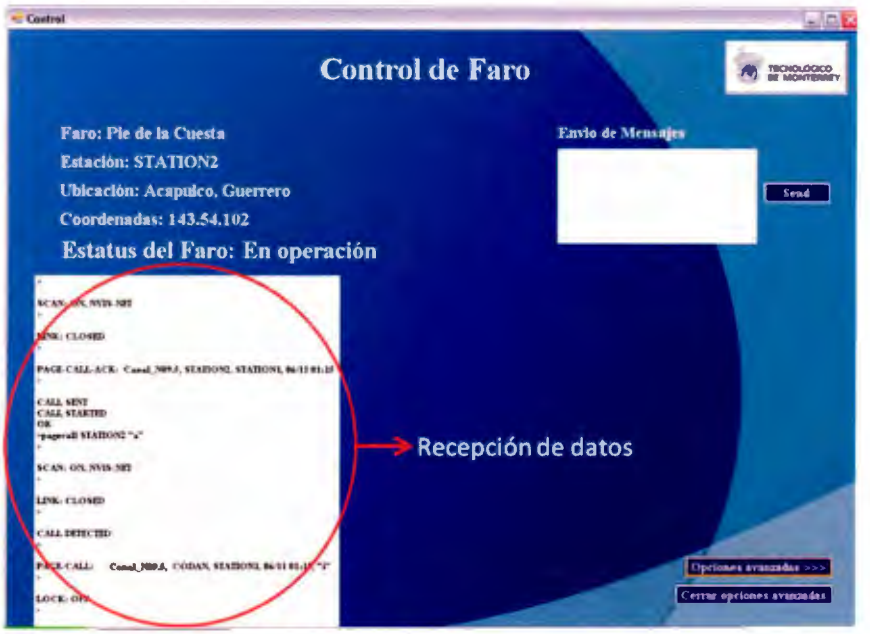


Figura 4.14 Verificación de recepción y envío de datos

Desarrollo del Circuito de Control del Faro.-

Esta parte del proyecto consta de 2 bloques: la tarjeta de control y los sensores y dispositivos electrónicos que controlan al faro.

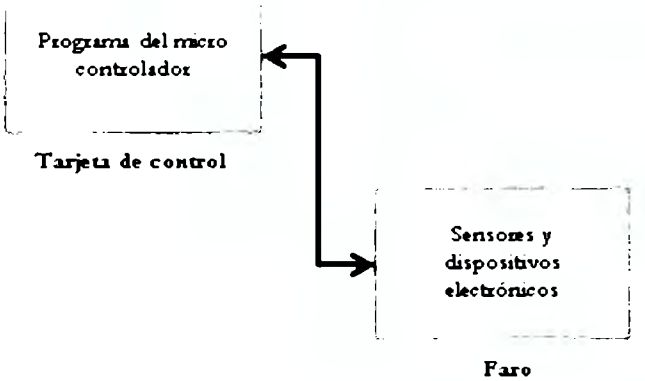


Figura 4.15 Diagrama a bloques del circuito de control del faro

1.- Programa del microcontrolador.-

Otro de los resultados obtenidos en la segunda etapa del proyecto fue un programa realizado en lenguaje C que funciona tanto para el microcontrolador ATTINY 2313, que fue utilizado en un principio en nuestro proyecto, como para el ATMEGA 16. Este programa realiza 2 funciones principales:

La primera consiste en recibir un "string" de datos como el siguiente:

```
CALL DETECTED
>
LOCK: BUSY
>
LOCK: OFF
>
PAGE-CALL:    Canal_N10.0, STATION2, STATION1, 26/08 15:08, "i"
>
CALL DETECTED
>
LINK: CLOSED
>
SCAN: ON, NVIS-NET
>
```

Este "string" de datos es generado por el radio CODAN NGT que se encuentra en las instalaciones del centro de control, que a su vez se enlaza con el radio que se encuentra en las instalaciones del faro y mediante el puerto serial envía la información al microcontrolador; éste desecha alguna información como el nombre de la estación y el día y la hora del mensaje, e identifica qué carácter fue enviado desde el centro de control, es decir la letra que se encuentra entre comillas, y dependiendo de ello realiza una acción. Si el carácter recibido es una letra "a", el puerto A0 cambia de 0v a 5v, y si es una letra "e" el mismo puerto cambia de 5v a 0v.

La segunda de las tareas consiste en monitorear las alarmas generadas por el faro mediante las interrupciones. Estas funcionan de manera distinta, por ejemplo cuando se produce la alarma de foco fundido o detección de intrusos, se produce un cambio de 5v a 0v en el puerto D2 o D3 según sea el caso, lo que activa una de las interrupciones externas, dentro de la interrupción se envía a través del puerto serial al radio CODAN de las instalaciones del faro, el siguiente string: ***PAGECALL STATION1 "i"*** ***\r*** el comando ***PAGECALL*** indica que se

desea enviar un mensaje de texto, ***STATION1*** se refiere a la estación a la que se desea enviar el mensaje, lo que se encuentra dentro de las comillas ***"i"*** es el mensaje que se está enviando y ***\r*** indica el final del string. La letra utilizada para indicar que se fundió el foco es la "m" y para indicar que se detectó un intruso se envía la "i".

Por otra parte la alarma de rotación funciona de manera distinta, cada que el faro da una vuelta completa sobre su eje, se produce una interrupción externa y esta se guarda en memoria como un pulso. De igual manera, cada 60 segundos se produce una interrupción por timer y se checa si en ese tiempo se produjo algún pulso, de ser así se restauran los valores iniciales de las variables y se vuelve a realizar el mismo proceso, pero, si no se detectó ningún pulso se envía la letra "p" que corresponde a la alarma de rotación.

En resumen, este programa es el encargado de proveer comunicación entre los radios y el faro de acuerdo a la lógica explicada en los párrafos anteriores, para así poder transformar mensajes de texto en señales eléctricas que realicen una acción y en sentido contrario interpreta señales para producir mensajes de texto.

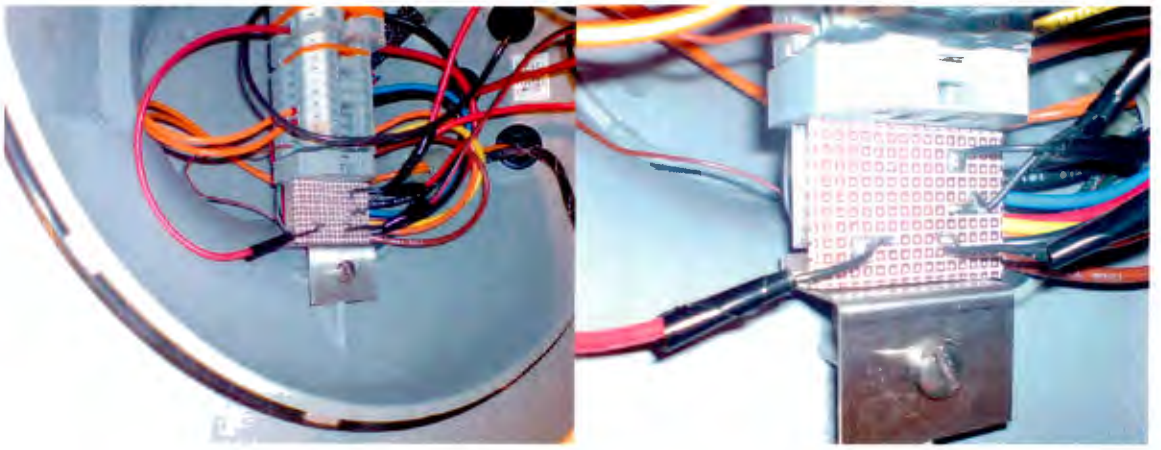
2.- Sensores y dispositivos electrónicos.-

Para la elaboración del circuito de control del faro fue necesario instalar un circuito que nos permitiera manipular las funciones básicas, así como también diversos sensores para llevar acabo el monitoreo constante de las señales provenientes de los mismos.

2.1.- Encendido/Apagado Remoto.-

El principal problema para llevar acabo el Encendido/Apagado remoto, fue que a partir de una señal TTL de 0 ó 5 volts proveniente del microcontrolador, teníamos que cortar o habilitar la alimentación de corriente al circuito de encendido del faro.

La solución fue instalar un "relay" o relevador de corriente directa que aguantara hasta 10 Amperes dado que el encendido consume una potencia efectiva de unos 110 Watts, y el faro tiene que ser alimentado con un voltaje alrededor de los 15 Volts. Otra ventaja que nos provee el "relay" es que acopla y al mismo tiempo aísla el circuito de la tarjeta de control con el circuito del faro, de manera que el microprocesador está protegido contra cualquier descarga.



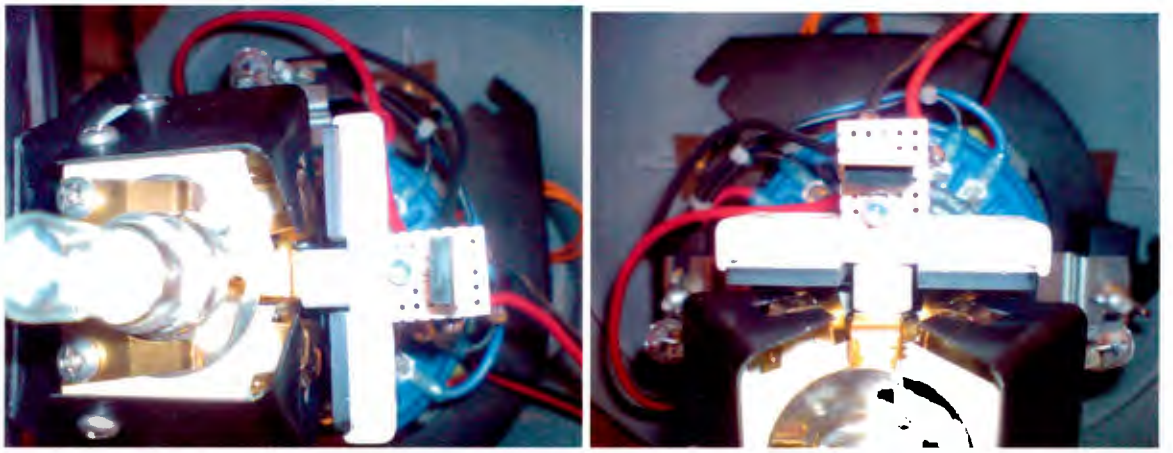
a) b)

Figura 4.16 "Relay" de corriente directa que controla el Encendido/Apagado Remoto a) Vista superior b) Acercamiento Vista Superior

Cómo se puede observar en la figura anterior, el "Relay" se instaló dentro del faro en la parte inferior, y se cuidó a detalle que todos los cables quedaran perfectamente aislados.

2.2.- Alarma de Foco Fundido.-

Para llevar acabo esta función decidimos optar por un fototransistor, ya que de esta forma podemos detectar a través de la intensidad luminosa, si se encuentra fundido un foco, o si falló la alimentación al foco, o cualquier cosa que haga que éste no se encuentre en estado de encendido. Este sensor quedó conformado por un fototransistor, un arreglo de resistencias que nos proporciona un voltaje alrededor de 10 volts cuando detecta fotones (foco encendido) y un voltaje de 0 volts cuando el foco se encuentra apagado. Para que el microcontrolador pudiese disparar la interrupción externa, es necesario que el sensor nos arroje señales TTL, de manera que se instaló un regulador de voltaje LM7805 para que tomásemos la salida de éste que está regulada a 5 volts y no directamente del sensor.



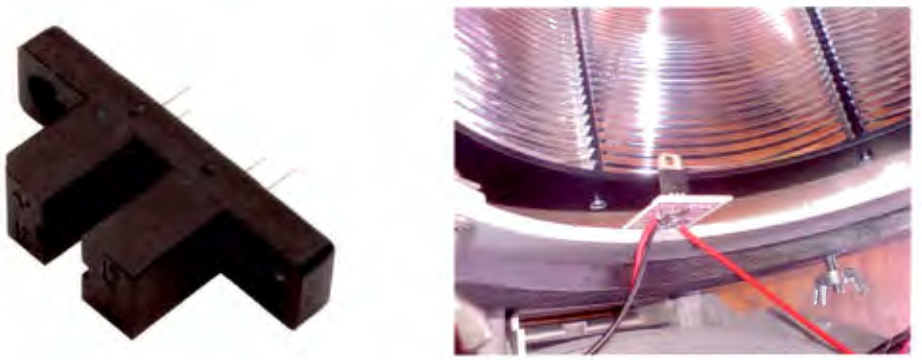
5) b)

Figura 4.17 Sensor de Foco Fundido (Fototransistor) colocado en el interior del faro junto al foco a) Vista superior perspectiva 1 b) Vista superior perspectiva 2

Como se puede observar en la figura anterior, el sensor se montó a un costado del foco para que pudiese recibir de manera directa los fotones emitidos por éste.

2.3.- Alarma de Rotación.-

Esta fue la alarma más difícil de elaborar, ya que optamos por utilizar un opto-interruptor y la instalación del mismo tuvo un alto grado de dificultad. El opto-interruptor se fijó en la parte inferior del faro justo debajo de la parte giratoria y fabricamos una estructura de aluminio para lograr obstruir el haz de luz infrarroja en cada ciclo de rotación. Esta estructura se instaló con una alta precisión en la parte rotatoria del faro. El opto-interruptor requiere un voltaje de alimentación de 5 volts, de manera que también le instalamos un regulador de voltaje para alimentarlo. La salida del opto-interruptor con el arreglo de resistencias adecuado nos proporciona un voltaje de +5 Volts cuando se obstruye el haz de luz, y 0 volts aproximadamente cuando no hay nada que lo obstruya.



a) b)

Figura 4.18 Opto-interruptor para la alarma de rotación a) opto-interruptor b) sensor de rotación ya instalado.

Este sensor genera un pulso por cada período de rotación de manera que el microcontrolador al detectar que a determinado tiempo no se genera este pulso, envía la señal de alarma de rotación al mismo para ser interpretada.

2.4.- Detector de Intrusos.-

Esta es una señal ajena al faro, que decidimos instalar por si alguien penetra a la torre donde se encuentra el faro. Hay diversos tipos de sensores que podemos utilizar para esta alarma como lo puede ser un fototransistor con un diodo emisor de luz de halógeno, un sensor magnético o un opto-interruptor. En un principio optamos por utilizar el fototransistor, pero debido a su corto tiempo de vida la mejor opción es utilizar un sensor magnético o un opto-interruptor.

Figura 4.19 Sensor de Intrusos: Diodo emisor de halógeno.

Figura 4.20 Sensor de Intrusos: Opto-interruptor

2.5.- Tarjeta de Control.-

Para la parte de control, se utilizó una tarjeta Atmel AVR con puerto serial, con espacio para un microcontrolador modelo AtMega 16 que es el cerebro de todo.

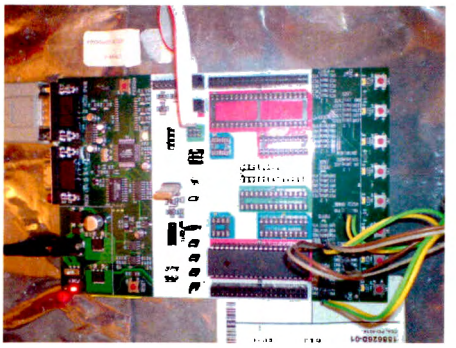


Figura 4.21 Tarjeta de Control Atmel AVR

Como se sabe, el uso de radiofrecuencia puede producir ruido en el microprocesador y así afectar el funcionamiento del mismo, sin embargo para ello se utilizan capacitores de "by-pass" que ayudan a filtrar señales no deseadas, en la tarjeta que se utilizo (STK 500), estos capacitores ya están incluidos.

Por otra parte se debe señalar que el ruido producido depende de la potencia de transmisión utilizada, por lo que a baja potencia el ruido no es significativo y no afecta al micro incluso sin el uso de los capacitores de "by-pass".

Pruebas del sistema

Para validar el funcionamiento de nuestro sistema realizamos diversas pruebas. Estas pruebas se realizaron a diferentes horas del día con el propósito de ver cómo afectaba el cambio en las condiciones ionosféricas la comunicación de los radios. Para estas pruebas se utilizó una antena de cable largo que opera en un rango de frecuencias de 1.6 a 30 MHz y nos permite transmitir hasta un máximo de 500 Km. Las pruebas fueron realizadas dentro de distintos puntos del Tecnológico de Monterrey Campus Ciudad de México, los resultados se muestran a continuación.

La primera de las pruebas se realizó con la "STATION1" ubicada en el laboratorio de electrónica 05 y la "STATION2" ubicada detrás de la biblioteca.

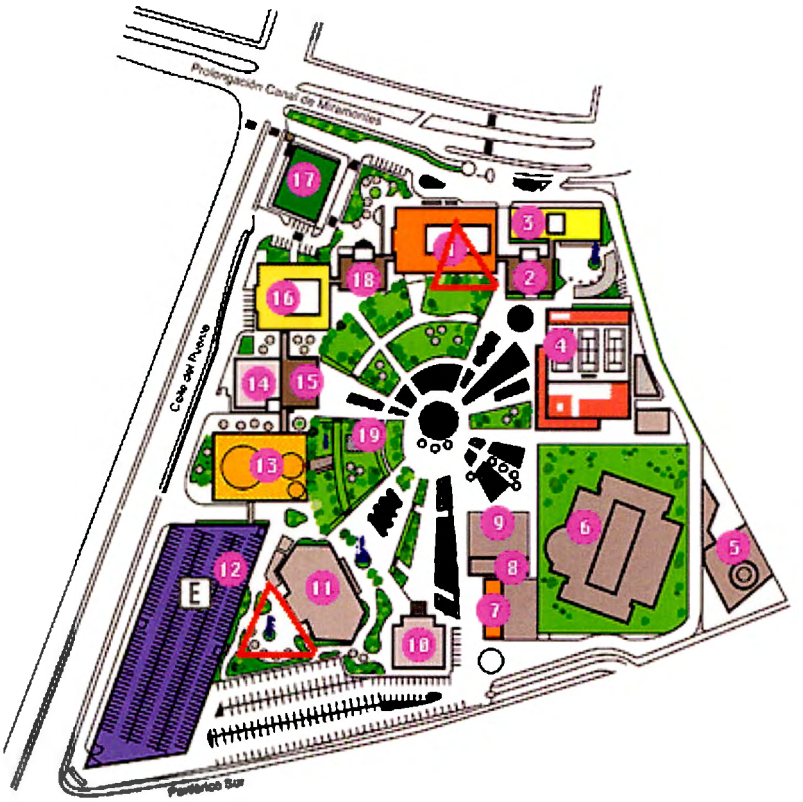


Figura 4.22 Ubicación geográfica de las estaciones

En la tabla 4.1 se observan los resultados que se obtuvieron de esta primera prueba.

Numero de prueba	Hora de la prueba	Encendido /Apagado	Alarma de rotación	Alarma de Intrusos	Alarma de foco fundido	Frecuencia de transmisión (MHz)
1	10:02	Sí	Sí	Sí	No	9
2	10:15	Sí	Sí	Sí	Sí	9
3	10:29	Sí	Sí	Sí	Sí	9
4	10:46	Sí	Sí	Sí	Sí	9
5	11:02	No	No	No	No	--
6	11:14	Sí	Sí	Sí	Sí	9.5
7	11:30	Sí	Sí	Sí	Sí	9
8	11:44	No	No	No	No	--
9	12:01	Sí	Sí	Sí	Sí	9
10	12:15	Si	Si	Sí	Sí	9
11	12:32	Sí	Si	Sí	Sí	9
12	12:44	Sí	Sí	Sí	Sí	9
13	13:01	Sí	Sí	Sí	Sí	9
14	13:15	Sí	Sí	No	Sí	8.5
15	13:31	Sí	Sí	Si	Sí	8.5
16	13:45	Sí	Sí	Sí	Sí	8
17	13:59	Sí	Sí	Sí	Sí	8.5
18	14:16	Si	Si	Si	Si	8
19	14:30	Sí	Sí	Sí	Sí	8
20	14:44	Sí	Sí	Si	Sí	8

Tabla 4.1 Resultados de pruebas

La segunda de las pruebas se realizó con la "STATION1" ubicada en el Cenote y la "STATION2" ubicada detrás de la biblioteca.

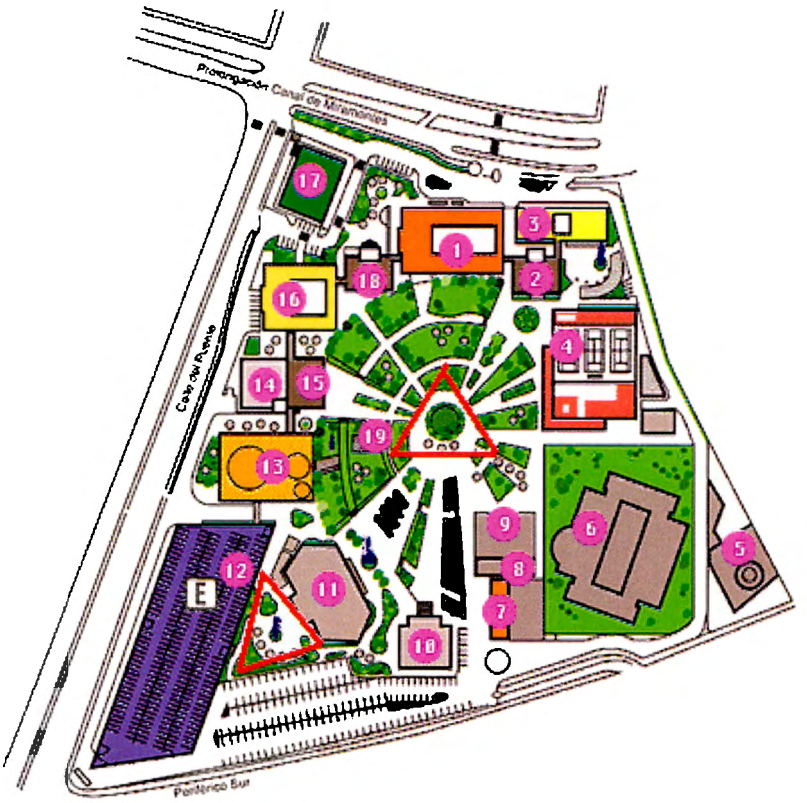


Figura 4.2 Ubicación geográfica de las estaciones

En la tabla 4.2 se observan los resultados que se obtuvieron en la segunda prueba.

Numero de prueba	Hora de la prueba	Encendido /Apagado	Alarma de rotación	Alarma de Intrusos	Alarma de foco fundido	Frecuencia de transmisión (MHz)
1	10:28	Sí	Sí	Sí	Sí	9
2	10:44	Sí	Sí	Sí	Sí	9
3	11:01	Sí	Sí	Sí	Sí	9
4	10:15	Sí	Sí	Sí	Sí	9
5	11:32	Sí	Sí	Sí	Sí	9
6	11:40	No	No	No	No	--
7	11:56	Sí	Sí	Sí	Sí	9.5
8	12:12	Sí	Sí	Sí	Sí	9
9	12:31	No	No	Sí	No	9
10	12:45	Si	Si	Sí	Sí	9
11	13:02	Sí	Sí	Sí	Sí	9
12	13:16	Sí	Sí	Sí	Sí	9
13	13:30	Sí	Si	Sí	Sí	9
14	13:42	Si	Sí	Si	Sí	8.5
15	13:56	Sí	Sí	Si	Sí	9
16	14:12	Sí	Sí	Sí	Sí	8.5
17	14:31	Sí	Sí	Sí	Sí	8
18	14:44	Sí	Sí	Sí	Sí	8
19	14:59	Sí	Sí	Sí	Sí	8
20	15:14	Si	Si	Si	Si	8

Tabla 4.2 Resultados Segunda Prueba

Con base en las pruebas realizadas, podemos verificar que el sistema funciona mejor cuando es mas tarde, esto debido a que el enlace entre los radios es una parte vital del proceso, y este puede presentar fallas alrededor de las 10 y las 12 debido a las condiciones de la ionosfera. En cuanto a los otros dispositivos involucrados, podemos decir que se presentaron muy pocas fallas y estas están asociadas a la conexión de los cables de las alarmas a la tarjeta de control, es decir de la correcta conexión de los cables de los dispositivos del faro a la tarjeta de control depende el que se presente una falla o no.

5. CONCLUSIONES

Primero que nada podemos decir que el proyecto plantea una solución innovadora, confiable y económica para el monitoreo de faros a distancia; innovadora porque no existe actualmente un sistema que monitoree a larga distancia las funciones que nosotros estamos controlando; confiable ya que de acuerdo a las pruebas que realizamos consideramos que el sistema es bueno ya que las fallas que ocurrieron fueron mínimas sin depender éstas de la distancia entre los radios; y económico ya que tomando en cuenta lo mencionado en la tabla 3.1 observamos que la tecnología HF comparada con la transmisión satelital y de microondas es lo más económico y que además de tener un pago inicial más bajo que las otras dos tecnologías, en el caso de los radios no se necesita pagar una renta mensual, lo cual hace que los costos a largo plazo sean bajos.

En la integración de los elementos que conforman el sistema, se presentaron algunas dificultades, que nos obligaron a modificar aspectos del mismo ya que en primera instancia no se nos proporcionó el faro que pretendíamos utilizar, lo cual como ya se mencionó anteriormente ocasionó un replanteamiento de objetivos. Esto ocasionó que tuviéramos que aplicar conocimientos de electrónica e instalar los componentes necesarios en el faro para poder controlar las funciones más importantes de este.

El uso de tecnología HF de nueva generación, en conjunto con el software y hardware desarrollado representan la base para el monitoreo y control a distancia de diversos dispositivos, lo cual significa que con el trabajo realizado hasta ahora se sentó una base para poder utilizar este mismo sistema, no sólo en faros sino en diversas aplicaciones que requieran ser controladas y monitoreadas a larga distancia, siendo esto también aplicable para reducir los costos al aplicar este sistema. Teniendo en consideración lo anterior, también sería necesario tomar en cuenta que para poder aplicar este proyecto en otros sistemas, son necesarios cambios en el mismo, ya que dependiendo de la aplicación que se le quiera dar habría diferentes formas de aplicar el hardware y software correspondientes, pero teniendo muy en cuenta que las bases para esto ya están muy bien cimentadas, con esto creemos que la aportación de nuestro proyecto es buena ya que es un proyecto versátil y multifuncional.

El desarrollo de este proyecto, involucró aspectos de software, hardware, electrónica digital y electrónica analógica, lo que representó un gran reto y a su vez un gran aprendizaje ya que se ocuparon muchos de los conocimientos obtenidos a lo largo de la carrera, además de ser un proyecto muy variado en cuanto a la investigación aplicación y aprendizaje. Primero en

la parte de la aplicación de C Sharp se utilizaron conocimientos de programación, y aunque a pesar de que en ninguna de las materias se nos enseñó C Sharp, hubo otras como computación en las que se programaba en C ++ o Java que sentaron muy buenas bases para aprender a utilizarlo y no tener que empezar desde cero en este ámbito. En la parte de los radios los conocimientos de las materias de comunicación nos fueron muy útiles ya que los radios son complicados de programar y utilizar. En la parte del microcontrolador fueron necesarios nuestros conocimientos de electrónica digital y para las instalaciones en el faro de electrónica analógica.

Además de esto cabe mencionar que a pesar de que los radios son únicamente el medio de transmisión entre la base y el faro, parte de la aportación en el primer semestre fue, como ya se mencionó, entender, programar y aprender a utilizarlos, ya que estos radios de corte militar tienen muchas funciones que no son sencillas de entender ni de aplicar, por lo cual nos tomó algo de tiempo dominar la tecnología de estos radios para poder después utilizarlos a nuestra conveniencia.

De acuerdo a las pruebas que realizamos se puede decir que los resultados fueron satisfactorios con algunas fallas, además de esto cabe destacar que en la aplicación real sobre el faro habría que considerar diferentes factores para determinar el tipo de antena a utilizar así como el diseño de esta misma. Después de esto es necesario hacer un análisis del terreno, así como el análisis del clima durante un año completo, para poder así determinar la frecuencia óptima de transmisión. Una de las cosas que nos facilitaría un poco esta cuestión es que como sólo contamos con una base, uno de los puntos de este análisis siempre va a ser el mismo.

En conclusión general podemos decir que al momento de la aplicación real habrá cosas a considerar que se tendrán que hacer, pero también cabe mencionar que la mayor parte del trabajo ya está hecho además de la experiencia sobre el control de los radios y los problemas que pueden surgir al momento de realizar las instalaciones de los faros o del dispositivo que necesitemos monitorear.

6. TRABAJOS FUTUROS

En una entrevista en la marina se presentó el proyecto y surgieron bastantes sugerencias para trabajos futuros como agregar funciones a la aplicación gráfica que permita a los usuarios de esta aplicación realizar su trabajo de una forma eficiente, también se nos sugirió realizar un manual de usuario para los usuarios tanto de la aplicación gráfica como de las instalaciones realizadas en los faros y la tarjeta para poder ahorrar tiempo en la preparación de las personas que tendrán esto a su cargo, además de que llevaría demasiado tiempo el hacer que estas personas entiendan por sí mismas lo que ya está hecho. Otra de las sugerencias fue el dar una especie de capacitación para los técnicos que se encarguen de la operación de todo esto, así como preparar a los técnicos para que sepan qué hacer en caso de falla de los radios o cualquier inconveniente que se pueda tener ya que como se mencionó anteriormente el uso y programación de los radios no es nada sencillo.

Otro de los trabajos a futuro es el alimentar todas nuestras instalaciones sobre el faro con la planta de electricidad con la que cuentan los faros para que un intruso no pueda cortar la corriente que alimenta nuestro sistema de seguridad. Con esto si el intruso quiere cortar la electricidad para que no funcione la alarma de intrusos le será imposible.

Una opción alternativa es instalar sobre el faro un sistema de seguridad para que en caso de que la corriente sea cortada se active otra alarma, esto se haría alimentando un relevador normalmente abierto con una fuente independiente, y que al cortar esta alimentación se active la alarma de corte de corriente.

Además de esto quedaría como trabajo futuro el acceder a la base de datos que crea el radio que tiene cosas importantes como lo son horas a la que se transmitió más eficientemente y diversos factores, y lo que se hará con esto es usar esta base de datos creada por los mismos radios en nuestro beneficio, ya sea con la misma aplicación que tenemos desarrollada o creando una nueva para esta única función, ya que esta función de los radios nos da un valor agregado que podemos utilizar en nuestro beneficio.

Buscar aplicaciones para las que el sistema que hemos desarrollado nos pueda funcionar para monitorear, mejorar o ahorrar en costos de cualquier índole para darle otra dimensión a nuestro trabajo, logrando así convertirlo en un sistema universal.

Es necesario investigar qué otras funciones nos proporciona este nuevo faro que nos puedan servir para hacer algunas aplicaciones extras o mejorar las ya realizadas, ya que este faro es el que la marina piensa emplear en las instalaciones existentes. Estas funciones nos permitirán hacer y mejorar procesos para controlar el mismo. Cabe señalar que lo anterior, ya no nos serviría para aplicar este mismo proceso para otra aplicación, ya que esta aplicación no sería un faro.

7. ANEXO I

Especificaciones técnicas de los radios Codan NGT SR transceiver de nueva generación.

En la tabla 11.1 se observan as características técnicas de los radios Codan NGT SR transceiver de nueva generación.

Rango de Frecuencias	1.6-30 MHz Tx, 250 KHz-30 MHz Rx
Canales	400
Modo de operación	SSB (JBE, USB, LSB)
Estabilidad de Frecuencia	+/- 5 ppm (-30 a 60°C)
Voltaje primario	12 V DC.
Potencia primaria	Rx (sin señal) 1 A, Tx JBE voz 6 A JBE dos tonos 10-14 A
Sensibilidad del Receptor	-125 dbm.
Potencia de Transmisión	125 Watts.
Temperatura	-30 a 60°C.
Medidas y Peso	RF unit: 210mm x 270 mm x 65 mm, 3.3 kg. Handset: 65mm x 35mm x 130mm, 0.3 kg. Junction Box: 135mm x 117mm x 38mm, 0.4 kg.
ALE link quality data	100 canales hasta en 100 estaciones por 24 horas para un total de hasta 72675 datos, revolución local de 8 bits y remota de 5 bits.
Polvo	MIL-STD-810F method 510.4
Vibración	MIL-STD-810F method 514.5
Shock	MIL-STD-810F method 516.5
Interfaz de Computadora	RS232, 300-19200 baudios
Interfaz GPS	NMEA-0183 (4800 baudios, RS232)

Tabla 11.1 Especificaciones técnicas de los radios Codan NGT SR transceiver de nueva generación.

8. ANEXO II

Código en lenguaje C# de las aplicaciones de escritura y lectura en consola.

A continuación se observan las líneas de código de la aplicación de escritura en consola.

```
/*
            Instituto Tecnológico y de Estudiso Superiores de Monterrey
 *                          Campus Ciudad de México
 *                        Proyectos de Ingeniería I
 *                 Control a Distancia de un Faro Mediante HF
 *
 *          Autores: Alan Roberto Gómez Rosas    A00995349
 *                   Jorge Marcin Cornish        A00994748
 *                   Javier Rodríguez Lizardi    A00996092
 *                   Mauricio Orvañanos Riquelme A00993448
 *
 */
using System;
using ACOMPORTLib;

namespace ActiveComport
{

    class Class1
    {

        static public ComPort m_objComport;

        static public void WriteStr(System.String str)
        {
            m_objComport.WriteString(str);
            Console.WriteLine("-> " + str + "\n");
        }


          static public void ReadStr()
          {
              System.String str;
              str = "notempty";
              m_objComport.Sleep(200);
              while (str != "")
              {
                  str = m_objComport.ReadString();
                  if (str != "")
                      Console.WriteLine("<- " + str + "\n");
              }
          }
```

```
        [STAThread]
        static void Main(string[] args)
        {

            m_objComport = new ComPortClass();
            m_objComport.BaudRate = 9600;
            m_objComport.Device = "COM1";
            m_objComport.LogFile = "C:\\ComportLog.txt";

            m_objComport.Open();

            if (m_objComport.LastError == 0)
            {
                Console.WriteLine("Open: SUCCESS\n");
            }
        else
        {
            Console.WriteLine("Open failed, Error : " + m_objComport.GetErrorDescription(m_objComport.LastE
        }


        if (m_objComport.LastError == 0)
        {

            WriteStr("hangup");



            Console.WriteLine("Close\n");
            m_objComport.Close();
        }

        m_objComport.Sleep(20000);
    }
}
}
```

A continuación se observan las líneas de código de la aplicación de lectura en consola.

```
/*
            Instituto Tecnológico y de Estudiso Superiores de Monterrey
 *                          Campus Ciudad de México
 *                        Proyectos de Ingeniería I
 *                 Control a Distancia de un Faro Mediante HF
 *
 *          Autores: Alan Roberto Gómez Rosas    A00995349
 *                   Jorge Marcin Cornish        A00994748
 *                   Javier Rodríguez Lizardi    A00996092
 *                   Mauricio Orvañanos Riquelme A00993448
 *
 */
using System;
using ACOMPORTLib;

namespace ActiveComport
{

    class Class1
    {

        static public ComPort m_objComport;

        static public void WriteStr(System.String str)
        {
            m_objComport.WriteString(str);
            Console.WriteLine("-> " + str + "\n");
        }


          static public void ReadStr()
          {
              System.String str;
              str = "notempty";
              m_objComport.Sleep(200);
              while (str != "")
              {
                  str = m_objComport.ReadString();
                  if (str != "")
                      Console.WriteLine("<- " + str + "\n");
              }
          }
```

```
        [STAThread]
        static void Main(string[] args)
        {

            m_objComport = new ComPortClass();
            m_objComport.BaudRate = 9600;
            m_objComport.Device = "COM1";
            m_objComport.LogFile = "C:\\ComportLog.txt";

            m_objComport.Open();

            if (m_objComport.LastError == 0)
            {
                Console.WriteLine("Open: SUCCESS\n");
            }
        else
        {
            Console.WriteLine("Open failed, Error : " + m_objComport.GetErrorDescription(m_objComport.LastE
        }


        if (m_objComport.LastError == 0)
        {


            ReadStr();



            Console.WriteLine("Close\n");
            m_objComport.Close();
        }

        m_objComport.Sleep(20000);
    }
}
```

9. ANEXO III

Código en lenguaje C# de la aplicación gráfica de control remoto.

Código en lenguaje C# de la aplicación gráfica de control remoto desarrollada.

Identificación de usuario

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace Validacion
{


    partial class Form1 : Form
    {

       static public Faros ff = new Faros();

        public Form1()
        {
            InitializeComponent();

        }
        private void MostrarPosicion()
        {
            int iTotal = usuariosBindingSource.Count;
            int iPos;

            if (iTotal == 0)
                etPosicion.Text = "No registros";
            else
            {
                iPos = usuariosBindingSource.Position + 1;
                etPosicion.Text = iPos.ToString() + " de " +
                  iTotal.ToString();
            }
        }
        private void Form1_Load(object sender, EventArgs e)
        {


      this.usuariosTableAdapter.Fill(this.validacionDataSet.usuarios);
            MessageBox.Show("Escriba su nombre de usuario y presione el
            boton buscar usuario");
            SetDesktopLocation(0, 0);
            SetClientSizeCore(1024, 1024);

        }

        private void button1_Click(object sender, EventArgs e)
        {

            DataTable miTabla = validacionDataSet.usuarios;
```

```
        DataRowCollection cfilas = miTabla.Rows;
        DataRow[] filaBuscada;
        string NL = Environment.NewLine;
        string criterio = "Usuario Like '*" + ctUsuario.Text + "*'";
        filaBuscada = miTabla.Select(criterio);
        if (filaBuscada.GetUpperBound(0) == -1)
        {
            MessageBox.Show("Nombre de usuario Incorrecto",
              "Buscar");
            ctUsuario.Clear();
            return;
        }

        int i, j;
        for (i = 0; i <= filaBuscada.GetUpperBound(0); i++)
        {
            if (MessageBox.Show("Usuario Encontrado" + NL +
              (string)filaBuscada[i]["usuario"] + NL, "Buscar",
              MessageBoxButtons.YesNo) == DialogResult.Yes)
            {
                for (j = 0; j <= cfilas.Count - 1; j++)
                    if (cfilas[j].Equals(filaBuscada[i]))
                    {
                        usuariosBindingSource.Position = j;
                        MessageBox.Show("Escriba su Password y oprima
                          validar");
                    }
                break;
            }
        }
            }

    private void button1_Click_1(object sender, EventArgs e)
    {
        DataTable miTabla1 = validacionDataSet.usuarios;
        DataRowCollection cfilas1 = miTabla1.Rows;
        DataRow[] filaBuscada1;
        string NL1 = Environment.NewLine;
        string criterio1 = "Pasword Like '*" + ctPasword.Text + "*'";
        filaBuscada1 = miTabla1.Select(criterio1);
        if (filaBuscada1.GetUpperBound(0) == -1)
        {
            MessageBox.Show("Password Incorrecto", "Buscar");
            ctPasword.Clear();
            return;
        }

        int i, j;
        for (i = 0; i <= filaBuscada1.GetUpperBound(0); i++)
        {
            if (MessageBox.Show("Confirma que es usted el usuario" +
              NL1 + (string)filaBuscada1[i]["usuario"] + NL1,
              "Buscar", MessageBoxButtons.YesNo) == DialogResult.Yes)
            {
                for (j = 0; j <= cfilas1.Count - 1; j++)
                    if (cfilas1[j].Equals(filaBuscada1[i]))
                    {
                        usuariosBindingSource.Position = j;
```

```
                        ff.ShowDialog();
                        this.Close();

                    }
                break;
            }
        }
    }
    }

}
```

Tablero virtual

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace Validacion
{

    partial class Faros : Form
    {
        static public string estacion;
        static public string nombre;
        static public string ciudad;
        static public string estado;
        static public string coordenadas;
        static public string f2;


       static public Control cc = new Control();

        public Faros()
        {
            InitializeComponent();
        }

        private void Faros_Load(object sender, EventArgs e)
        {
            this.farosTableAdapter.Fill(this.farosDataSet.Faros);
            Faros.ActiveForm.Visible = false;
            SetDesktopLocation(0, 0);
            SetClientSizeCore(1024, 1024);
            btAñadir.Hide();
            btBorrar.Hide();
            btBuscar.Hide();
            ctBuscar.Hide();
            label6.Hide();
```

```
        BtnCerrarOpAv.Hide();
        pictureBox6.Hide();
        pictureBox2.Hide();
        pictureBox3.Hide();
        pictureBox4.Hide();
        pictureBox5.Hide();
        MostrarPosicion();


    }
    private void Form1_FormClosing(object sender,
        FormClosingEventArgs e)
    {
        if (farosDataSet.HasChanges())
        {
            farosTableAdapter.Update(farosDataSet.Faros);
            MessageBox.Show("Origen de datos actualizado");
        }
    }


    private void button1_Click(object sender, EventArgs e)
    {
        estacion = textBox4.Text;
        nombre = textBox1.Text;
        ciudad = textBox3.Text;
        estado = textBox2.Text;
        coordenadas = textBox5.Text;
        cc.ShowDialog();
    }
    private void MostrarPosicion()
    {
        int iTotal = farosBindingSource.Count;
        int iPos;

        if (iTotal == 0)
            etPosicion.Text = "No registros";
        else
        {
            iPos = farosBindingSource.Position + 1;
            etPosicion.Text = iPos.ToString() + " de " +
              iTotal.ToString();
        }
    }

    private void textBox7_TextChanged(object sender, EventArgs e)
    {

    }

    private void button8_Click(object sender, EventArgs e)
    {
        farosBindingSource.Position = farosBindingSource.Count - 1;
        pictureBox1.Hide();
        pictureBox6.Show();
        pictureBox2.Hide();
        pictureBox3.Hide();
        pictureBox4.Hide();
```

```
        pictureBox5.Hide();
        MostrarPosicion();

    }

    private void button7_Click(object sender, EventArgs e)
    {
        farosBindingSource.Position += 1;
        f2 = textBox4.Text;
        if (f2 == "STATION2")
        {
            pictureBox1.Hide();
            pictureBox6.Hide();
            pictureBox3.Hide();
            pictureBox4.Hide();
            pictureBox5.Hide();
            pictureBox2.Show();
        }
        if (f2 == "STATION3")
        {
            pictureBox1.Hide();
            pictureBox6.Hide();
            pictureBox3.Show();
            pictureBox4.Hide();
            pictureBox5.Hide();
            pictureBox2.Hide();
        }
        if (f2 == "STATION4")
        {
            pictureBox1.Hide();
            pictureBox6.Hide();
            pictureBox3.Hide();
            pictureBox4.Show();
            pictureBox5.Hide();
            pictureBox2.Hide();
        }
        if (f2 == "STATION5")
        {
            pictureBox1.Hide();
            pictureBox6.Hide();
            pictureBox3.Hide();
            pictureBox4.Hide();
            pictureBox5.Show();
            pictureBox2.Hide();
        }
        if (f2 == "STATION6")
        {
            pictureBox1.Hide();
            pictureBox6.Show();
            pictureBox3.Hide();
            pictureBox4.Hide();
            pictureBox5.Hide();
            pictureBox2.Hide();
        }
        MostrarPosicion();

    }
```

```
    private void button2_Click(object sender, EventArgs e)
    {
        DataTable miTabla = farosDataSet.Faros;
        DataRowCollection cfilas = miTabla.Rows;
        DataRow nuevaFila;
        try
        {
            //Nueva fila
            nuevaFila = miTabla.NewRow();
            //Datos por omision para las columnas de la nueva fila
            nuevaFila[0] = "Faro";                    //columna 0
            nuevaFila[1] = "Estado";                  //columna 0
            nuevaFila[2] = "Ciudad";                  //columna 0
            nuevaFila[3] = "Código";                  //columna 0
            nuevaFila[4] = "Coordenadas";             //columna 0
            nuevaFila[5] = "Observaciones";           //columna 0
            cfilas.Add(nuevaFila);
            btBuscar.PerformClick();  //hacer click en ultimo
            MostrarPosicion();
            //Usuario.Focus(); //enfocar la caja de texto Usuario
        }
        catch (ConstraintException ex)
        {
            //Capturar el posible error por clave duplicada
            MessageBox.Show(ex.Message);
        }
    }

    private void btBorrar_Click(object sender, EventArgs e)
    {
        DataRowView vistaFilaActual;
        string NL = Environment.NewLine;

        if (MessageBox.Show("¿Desea borrar este registro?" + NL,
"Buscar", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
DialogResult.Yes)
        {
            vistaFilaActual =
((DataRowView)farosBindingSource.Current);
            vistaFilaActual.Row.Delete();
            MostrarPosicion();
        }
    }



    private void button5_Click(object sender, EventArgs e)
    {
        farosBindingSource.Position = 0;
        MostrarPosicion();

    }

    private void button6_Click(object sender, EventArgs e)
    {
        farosBindingSource.Position -= 1;
        MostrarPosicion();
```

```
}

private void btBuscar_Click(object sender, EventArgs e)
{
    DataTable miTabla = farosDataSet.Faros;
    DataRowCollection cfilas = miTabla.Rows;
    DataRow[] filaBuscada; //matriz de filas
    string NL = Environment.NewLine;
    //Buscar en la columna nombre de cada fila
    string criterio = "Faro Like '*" + ctBuscar.Text + "*'";

    filaBuscada = miTabla.Select(criterio);

    if (filaBuscada.GetUpperBound(0) == -1)
    {
        MessageBox.Show("No se encontró ningún faro con ese
    nombre", "Buscar");
        return;
    }

    //Seleccionar de las filas encontradas la que buscamos
    int i, j;
    for (i = 0; i <= filaBuscada.GetUpperBound(0); i++)
    {
        if (MessageBox.Show("¿Es este el faro buscado?" + NL +
    (string)filaBuscada[i]["faro"] + NL, "Buscar",
    MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            //Si: mostrar en el formulario la fila seleccionada
            for (j = 0; j <= cfilas.Count - 1; j++)
                if (cfilas[j].Equals(filaBuscada[i]))
                {
                    farosBindingSource.Position = j;
                    MostrarPosicion();
                }
            break;
        }
    }
    ctBuscar.Clear();
}

private void button5_Click_1(object sender, EventArgs e)
{
    farosBindingSource.Position = 0;
    pictureBox1.Show();
    pictureBox6.Hide();
    pictureBox2.Hide();
    pictureBox3.Hide();
    pictureBox4.Hide();
    pictureBox5.Hide();
    MostrarPosicion();

}

private void button6_Click_1(object sender, EventArgs e)
{
    farosBindingSource.Position -= 1;
    f2 = textBox4.Text;
```

```
        if (f2 == "STATION2")
        {
            pictureBox1.Hide();
            pictureBox6.Hide();
            pictureBox3.Hide();
            pictureBox4.Hide();
            pictureBox5.Hide();
            pictureBox2.Show();
        }
        if (f2 == "STATION3")
        {
            pictureBox1.Hide();
            pictureBox6.Hide();
            pictureBox3.Show();
            pictureBox4.Hide();
            pictureBox5.Hide();
            pictureBox2.Hide();
        }
        if (f2 == "STATION4")
        {
            pictureBox1.Hide();
            pictureBox6.Hide();
            pictureBox3.Hide();
            pictureBox4.Show();
            pictureBox5.Hide();
            pictureBox2.Hide();
        }
        if (f2 == "STATION5")
        {
            pictureBox1.Hide();
            pictureBox6.Hide();
            pictureBox3.Hide();
            pictureBox4.Hide();
            pictureBox5.Show();
            pictureBox2.Hide();
        }
        if (f2 == "STATION1")
        {
            pictureBox1.Show();
            pictureBox6.Hide();
            pictureBox3.Hide();
            pictureBox4.Hide();
            pictureBox5.Hide();
            pictureBox2.Hide();
        }
        MostrarPosicion();

    }

    private void textBox1_TextChanged(object sender, EventArgs e)
    {

    }

    private void BtnOpAv_Click(object sender, EventArgs e)
    {
        btAñadir.Show();
        btBorrar.Show();
```

```
        btBuscar.Show();
        ctBuscar.Show();
        label6.Show();
        etPosicion.Hide();
        BtnCerrarOpAv.Show();
        pictureBox1.Hide();
        pictureBox6.Hide();
        pictureBox3.Hide();
        pictureBox4.Hide();
        pictureBox5.Hide();
        pictureBox2.Hide();
    }

    private void BtnCerrarOpAv_Click(object sender, EventArgs e)
    {
        btAñadir.Hide();
        btBorrar.Hide();
        btBuscar.Hide();
        ctBuscar.Hide();
        label6.Hide();
        BtnCerrarOpAv.Hide();
        etPosicion.Show();
        f2 = textBox4.Text;
        if (f2 == "STATION2")
        {
            pictureBox1.Hide();
            pictureBox6.Hide();
            pictureBox3.Hide();
            pictureBox4.Hide();
            pictureBox5.Hide();
            pictureBox2.Show();
        }
        if (f2 == "STATION3")
        {
            pictureBox1.Hide();
            pictureBox6.Hide();
            pictureBox3.Show();
            pictureBox4.Hide();
            pictureBox5.Hide();
            pictureBox2.Hide();
        }
        if (f2 == "STATION4")
        {
            pictureBox1.Hide();
            pictureBox6.Hide();
            pictureBox3.Hide();
            pictureBox4.Show();
            pictureBox5.Hide();
            pictureBox2.Hide();
        }
        if (f2 == "STATION5")
        {
            pictureBox1.Hide();
            pictureBox6.Hide();
            pictureBox3.Hide();
            pictureBox4.Hide();
            pictureBox5.Show();
            pictureBox2.Hide();
```

```
            }
            if (f2 == "STATION1")
            {
                pictureBox1.Show();
                pictureBox6.Hide();
                pictureBox3.Hide();
                pictureBox4.Hide();
                pictureBox5.Hide();
                pictureBox2.Hide();
            }
            if (f2 == "STATION6")
            {
                pictureBox1.Hide();
                pictureBox6.Show();
                pictureBox3.Hide();
                pictureBox4.Hide();
                pictureBox5.Hide();
                pictureBox2.Hide();
            }


        }
    }
}
```

Control del sistema de comunicaciones

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO.Ports;
using System.Threading;
using System.Text;
using System.Windows.Forms;
namespace Validacion
{
    public partial class Control : Form
    {
        char letra;
        string cadena;


        public Control()
        {
            InitializeComponent();
            Serial.Open();
            Serial.RtsEnable = true;
        }
```

```
private void Control_Load(object sender, EventArgs e)
{
    SetDesktopLocation(0, 0);
    SetClientSizeCore(1024, 1024);
    LbNombre.Text = "Faro: " + Faros.nombre;
    LbEstacion.Text = "Estación: " + Faros.estacion;
    LbUbicacion.Text = "Ubicación: " + Faros.ciudad + ", " +
    Faros.estado;
    LbCoordenadas.Text = "Coordenadas: " + Faros.coordenadas;
    LbRecibidos.Text = "Estatus del Faro: Inactivo";
    LbEnviados.Hide();
    TxtMensaje.Hide();
    BtnSend.Hide();
    BtnCerrar.Hide();
    txbRx.Hide();
    pictureBox1.Hide();
    LbRxT.Hide();
    BtnLamp2.Hide();
    BtnCambio.Hide();
    BtnRotacion.Hide();
    BtnLampara.Hide();
    BtnDetector.Hide();

    if (Faros.estacion == "STATION1")
    {
        pictureBox2.Show();
        pictureBox3.Hide();
        pictureBox4.Hide();
        pictureBox5.Hide();
        pictureBox6.Hide();
        pictureBox7.Hide();
    }
    if (Faros.estacion == "STATION2")
    {
        pictureBox2.Hide();
        pictureBox3.Show();
        pictureBox4.Hide();
        pictureBox5.Hide();
        pictureBox6.Hide();
        pictureBox7.Hide();
    }
    if (Faros.estacion == "STATION3")
    {
        pictureBox2.Hide();
        pictureBox3.Hide();
        pictureBox4.Show();
        pictureBox5.Hide();
        pictureBox6.Hide();
        pictureBox7.Hide();
    }
    if (Faros.estacion == "STATION4")
    {
        pictureBox2.Hide();
        pictureBox3.Hide();
        pictureBox4.Hide();
        pictureBox5.Show();
        pictureBox6.Hide();
```

```
            pictureBox7.Hide();

        }
        if (Faros.estacion == "STATION5")
        {
            pictureBox2.Hide();
            pictureBox3.Hide();
            pictureBox4.Hide();
            pictureBox5.Hide();
            pictureBox6.Show();
            pictureBox7.Hide();
        }
        if (Faros.estacion == "STATION6")
        {
            pictureBox2.Hide();
            pictureBox3.Hide();
            pictureBox4.Hide();
            pictureBox5.Hide();
            pictureBox6.Hide();
            pictureBox7.Show();
        }

    }

    private void Form1_FormClosing(object sender,
        FormClosingEventArgs e)
    {
        this.Close();
    }

    private void BtnEncendido_Click(object sender, EventArgs e)
    {
        string Texto;
        Texto = "pagecall "+ Faros.estacion + " \"a\"" + "\r";
        Serial.Write(Texto);
        LbRecibidos.Text = "Estatus del Faro: En operación";

    }

    private void BtnApagado_Click(object sender, EventArgs e)
    {
        string Texto;
        Texto = "pagecall "+ Faros.estacion + " \"e\"" + "\r";
        Serial.Write(Texto);
        LbRecibidos.Text = "Estatus del Faro: Apagado";

    }

    private void BtnLamp1_Click(object sender, EventArgs e)
    {
        string Texto;
        Texto = "alecall " + Faros.estacion + "\r";
        Serial.Write(Texto);
        LbRecibidos.Text = "Estatus del Faro: Solucionando falla";

    }
```

```
private void ReceivedData(object sender,
    System.IO.Ports.SerialDataReceivedEventArgs e)
{



    cadena = Serial.ReadLine();
    int tamaño=cadena.Length;
    LbRxT.Text = cadena + LbRxT.Text;
    for (int i = 0; i < tamaño;i++ )
    {
        if (cadena[i] == '"' && cadena[i + 1] != '\r')
        {
            letra = cadena[i+1];
        }
    }

    if (Faros.estacion=="STATION2")
    {
    switch (letra)
    {

        case 'i':
            LbRecibidos.Text = " Estatus del Faro: Intruso
          detectado";
            pictureBox1.Show();
            pictureBox3.Hide();
            break;
        case 'm':
            LbRecibidos.Text = " Estatus del Faro: Foco fundido";
            pictureBox1.Hide();
            break;
        case 'p':
            LbRecibidos.Text = " Estatus del Faro: Falla de
          rotación";
            pictureBox1.Hide();
            break;

    }
    }




}

private void BtnSend_Click(object sender, EventArgs e)
{
    string Texto;
    Texto = TxtMensaje.Text;
    txbRx.Text = ">";
    TxtMensaje.Text = "";
    Serial.Write(Texto);

}

private void TxtMensaje_TextChanged(object sender, EventArgs e)
{
```

```
}

private void BtnSeleccion_Click(object sender, EventArgs e)
{
    LbEnviados.Show();
    TxtMensaje.Show();
    BtnSend.Show();
    BtnCerrar.Show();
    BtnEncendido.Hide();
    BtnApagado.Hide();
    BtnLamp1.Hide();
    BtnLamp2.Hide();
    BtnCambio.Hide();
    BtnRotacion.Hide();
    BtnLampara.Hide();
    BtnDetector.Hide();
    LbRxT.Show();
    pictureBox1.Hide();
    pictureBox2.Hide();
    pictureBox3.Hide();
    pictureBox4.Hide();
    pictureBox5.Hide();
    pictureBox6.Hide();
    pictureBox7.Hide();

}

private void BtnCerrar_Click(object sender, EventArgs e)
{
    LbEnviados.Hide();
    TxtMensaje.Hide();
    BtnSend.Hide();
    BtnCerrar.Hide();
    BtnEncendido.Show();
    BtnApagado.Show();
    BtnLamp1.Show();
    LbRxT.Hide();
    if (Faros.estacion == "STATION1")
    {
        pictureBox2.Show();
        pictureBox3.Hide();
        pictureBox4.Hide();
        pictureBox5.Hide();
        pictureBox6.Hide();
        pictureBox7.Hide();
    }
    if (Faros.estacion == "STATION2")
    {
        pictureBox2.Hide();
        pictureBox3.Show();
        pictureBox4.Hide();
        pictureBox5.Hide();
        pictureBox6.Hide();
        pictureBox7.Hide();
    }
    if (Faros.estacion == "STATION3")
    {
```

```
                pictureBox2.Hide();
                pictureBox3.Hide();
                pictureBox4.Show();
                pictureBox5.Hide();
                pictureBox6.Hide();
                pictureBox7.Hide();
            }
            if (Faros.estacion == "STATION4")
            {
                pictureBox2.Hide();
                pictureBox3.Hide();
                pictureBox4.Hide();
                pictureBox5.Show();
                pictureBox6.Hide();
                pictureBox7.Hide();

            }
            if (Faros.estacion == "STATION5")
            {
                pictureBox2.Hide();
                pictureBox3.Hide();
                pictureBox4.Hide();
                pictureBox5.Hide();
                pictureBox6.Show();
                pictureBox7.Hide();
            }
            if (Faros.estacion == "STATION6")
            {
                pictureBox2.Hide();
                pictureBox3.Hide();
                pictureBox4.Hide();
                pictureBox5.Hide();
                pictureBox6.Hide();
                pictureBox7.Show();
            }

        }


    }
}
```

10. ANEXO IV

Código en lenguaje C del microcontrolador.

Código en lenguaje C del microcontrolador

```
/*****************************************************
Project :  Control y Monitoreo de un Faro a través de HF
Version :
Date    : 24/08/2007
Authors  : Jorge Marcin Cornish, Alan R. Gómez Rosas,
           Javier Rodríguez Lizardi y Mauricio Orvañanos Riquelme
Company : Instituto Tecnológico y de Estudios Superiores de Monterrey
           Campus Ciudad de México
Comments:


Chip type           : ATtiny2313V
Clock frequency     : 3.686000 MHz
Memory model        : Tiny
External SRAM size  : 0
Data Stack size     : 32
*****************************************************/

#include <tiny2313.h>

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

```

```
#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 32
unsigned char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

volatile unsigned char countms=0;
bit save=0;
bit finTX=0;

// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
char status,data;
status=UCSRA;
data=UDR;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
   {
    if (data == '"') save=1;
    if (save == 1)
    {
    ++countms;
         if (countms <= 2)
         {
         rx_buffer[rx_wr_index]=data;
                 if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
                 if (++rx_counter == RX_BUFFER_SIZE)
                 {
                 rx_counter=0;
                 rx_buffer_overflow=1;
                 };
         };
         if (countms == 2)
         {
          finTX=1;
          countms=0;
          save=0;
         };
    };
    };
}
```

```
83
84  #ifndef _DEBUG_TERMINAL_IO_
85  // Get a character from the USART Receiver buffer
86  #define _ALTERNATE_GETCHAR_
87  #pragma used+
88  char getchar(void)
89  {
90  char data;
91  while (rx_counter==0);
92  data=rx_buffer[rx_rd_index];
93  if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
94  #asm("cli")
95  --rx_counter;
96  #asm("sei")
97  return data;
98  }
99  #pragma used-
100 #endif
101
102 // USART Transmitter buffer
103 #define TX_BUFFER_SIZE 8
104 char tx_buffer[TX_BUFFER_SIZE];
105
106 #if TX_BUFFER_SIZE<256
107 unsigned char tx_wr_index,tx_rd_index,tx_counter;
108 #else
109 unsigned int tx_wr_index,tx_rd_index,tx_counter;
110 #endif
111
112 // USART Transmitter interrupt service routine
113 interrupt [USART_TXC] void usart_tx_isr(void)
114 {
115 if (tx_counter)
116    {
117    --tx_counter;
118    UDR=tx_buffer[tx_rd_index];
119    if (++tx_rd_index == TX_BUFFER_SIZE) tx_rd_index=0;
120    };
121 }
122
123 #ifndef _DEBUG_TERMINAL_IO_
124 // Write a character to the USART Transmitter buffer
125 #define _ALTERNATE_PUTCHAR_
126 #pragma used+
127 void putchar(char c)
128 {
```

```
while (tx_counter == TX_BUFFER_SIZE);
#asm("cli")
if (tx_counter || ((UCSRA & DATA_REGISTER_EMPTY)==0))
   {
   tx_buffer[tx_wr_index]=c;
   if (++tx_wr_index == TX_BUFFER_SIZE) tx_wr_index=0;
   ++tx_counter;
   }
else
   UDR=c;
#asm("sei")
}
#pragma used-
#endif

// Standard Input/Output functions
#include <stdio.h>

// Declare your global variables here

void main(void)
{
// Declare your local variables here
unsigned char var=0;
// Crystal Oscillator division factor: 1
#pragma optsize-
CLKPR=0x80;
CLKPR=0x00;
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+
#endif

// Input/Output Ports initialization
// Port A initialization
// Func2=In Func1=In Func0=In
// State2=T State1=T State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;

// Port D initialization
// Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0A output: Disconnected
// OC0B output: Disconnected
TCCR0A=0x00;
TCCR0B=0x00;
TCNT0=0x00;
OCR0A=0x00;
OCR0B=0x00;
```

```
191
192 // Timer/Counter 1 initialization
193 // Clock source: System Clock
194 // Clock value: Timer 1 Stopped
195 // Mode: Normal top=FFFFh
196 // OC1A output: Discon.
197 // OC1B output: Discon.
198 // Noise Canceler: Off
199 // Input Capture on Falling Edge
200 // Timer 1 Overflow Interrupt: Off
201 // Input Capture Interrupt: Off
202 // Compare A Match Interrupt: Off
203 // Compare B Match Interrupt: Off
204 TCCR1A=0x00;
205 TCCR1B=0x00;
206 TCNT1H=0x00;
207 TCNT1L=0x00;
208 ICR1H=0x00;
209 ICR1L=0x00;
210 OCR1AH=0x00;
211 OCR1AL=0x00;
212 OCR1BH=0x00;
213 OCR1BL=0x00;
214
215 // External Interrupt(s) initialization
216 // INT0: Off
217 // INT1: Off
218 // Interrupt on any change on pins PCINT0-7: Off
219 GIMSK=0x00;
220 MCUCR=0x00;
221
222 // Timer(s)/Counter(s) Interrupt(s) initialization
223 TIMSK=0x00;
224
225 // Universal Serial Interface initialization
226 // Mode: Disabled
227 // Clock source: Register & Counter=no clk.
228 // USI Counter Overflow Interrupt: Off
229 USICR=0x00;
230
231 // USART initialization
232 // Communication Parameters: 8 Data, 1 Stop, No Parity
233 // USART Receiver: On
234 // USART Transmitter: On
235 // USART Mode: Asynchronous
236 // USART Baud rate: 9600
237 UCSRA=0x00;
238 UCSRB=0xD8;
239 UCSRC=0x06;
240 UBRRH=0x00;
241 UBRRL=0x17;
242
243 // Analog Comparator initialization
244 // Analog Comparator: Off
245 // Analog Comparator Input Capture by Timer/Counter 1: Off
246 ACSR=0x80;
247
248 // Global enable interrupts
249
250 #asm("sei")
251
```

```
252  while (1)
253        {
254
255        if (finTX == 1)
256        {
257
258          finTX=0;
259          var=getchar();
260          var2=getchar();
261
262
263        //putchar(var);
264        }
265
266        }
267  }
```

11. ANEXO V

Hoja de especificaciones del microcontrolador ATtiny2313V.

ATtiny2313V

Features

- Utilizes the AVR® RISC Architecture
- AVR – High-performance and Low-power RISC Architecture
  - 120 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 24 MIPS Throughput at 24 MHz
- Data and Non-volatile Program and Data Memories
  - 2K Bytes of In-System Self Programmable Flash
    Endurance 10,000 Write/Erase Cycles
  - 128 Bytes In-System Programmable EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 128 Bytes Internal SRAM
  - Programming Lock for Flash Program and EEPROM Data Security
- Peripheral Features
  - One 8-bit Timer/Counter with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare and Capture Modes
  - Four PWM Channels
  - On-chip Analog Comparator
  - Programmable Watchdog Timer with On-chip Oscillator
  - USI – Universal Serial Interface
  - Full Duplex USART
- Special Microcontroller Features
  - debugWIRE On-chip Debugging
  - In-System Programmable via SPI Port
  - External and Internal Interrupt Sources
  - Low-power Idle, Power-down, and Standby Modes
  - Enhanced Power-on Reset Circuit
  - Programmable Brown-out Detection Circuit
  - Internal Calibrated Oscillator
- I/O and Packages
  - 18 Programmable I/O Lines
  - 20-pin PDIP, 20-pin SOIC, and 32-pin MLF
- Operating Voltages
  - 1.8 - 5.5V (ATtiny2313)
- Speed Grades
  - ATtiny2313V: 0 - 6 MHz @ 1.8 - 5.5V, 0 - 12 MHz @ 2.7 - 5.5V
  - ATtiny2313: 0 - 12 MHz @ 2.7 - 5.5V, 0 - 24 MHz @ 4.5 - 5.5V
- Power Consumption Estimates
  - Active Mode
    1 MHz, 1.8V: 300 µA
    32 kHz, 1.8V: 20 µA (including oscillator)
  - Power-down Mode
    < 0.2 µA at 1.8V

8-bit **AVR®** Microcontroller with 2K Bytes In-System Programmable Flash

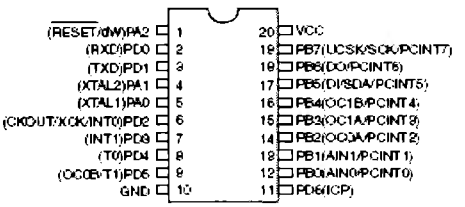
ATtiny2313/V

Preliminary Summary

Pin Configurations

Figure 1. Pinout ATtiny2313

PDIP/SOIC



Overview

The ATtiny2313 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATtiny2313 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Pin Descriptions

VCC Digital supply voltage.

GND Ground.

Port A (PA2..PA0) Port A is a 3-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port A pins that are externally pulled low will source current if the pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port A also serves the functions of various special features of the ATtiny2313 as listed on page 52.

Port B (PB7..PB0) Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATtiny2313 as listed on page 52.

Port D (PD6..PD0) Port D is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATtiny2313 as listed on page 55.

$\overline{\text{RESET}}$ Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 15 on page 33. Shorter pulses are not guaranteed to generate a reset. The Reset input is an alternate function for PA2 and dW.

XTAL1 Input to the inverting Oscillator amplifier and input to the internal clock operating circuit. XTAL1 is an alternate function for PA0.

XTAL2 Output from the inverting Oscillator amplifier. XTAL2 is an alternate function for PA1.

Figure 3.

12. ANEXO VI

Hoja de especificaciones del microcontrolador ATMega16.

ATMega16

Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- Nonvolatile Program and Data Memories
  - 16K Bytes of In-System Self-Programmable Flash
    Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - 512 Bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 1K Byte Internal SRAM
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    8 Single-ended Channels
    7 Differential Channels in TQFP Package Only
    2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad MLF
- Operating Voltages
  - 2.7 - 5.5V for ATmega16L
  - 4.5 - 5.5V for ATmega16
- Speed Grades
  - 0 - 8 MHz for ATmega16L
  - 0 - 16 MHz for ATmega16
- Power Consumption @ 1 MHz, 3V, and 25°C for ATmega16L
  - Active: 1.1 mA
  - Idle Mode: 0.35 mA
  - Power-down Mode: < 1 µA

8-bit **AVR®** Microcontroller with 16K Bytes In-System Programmable Flash

ATmega16
ATmega16L

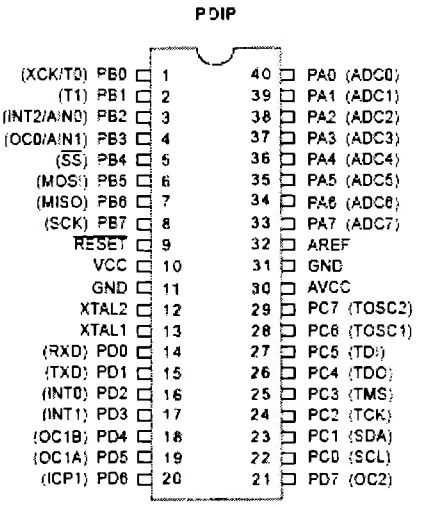
Summary

2466HS–AVR

Note: This is a summary document. A complete doc
is available on our Web site at www.atmel.com.

Pin Configurations

Figure 1. Pinouts ATmega16



Pin Descriptions

VCC Digital supply voltage.

GND Ground.

Port A (PA7..PA0) Port A serves as the analog inputs to the A/D Converter.

Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B (PB7..PB0)	Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port B also serves the functions of various special features of the ATmega16 as listed on page 56.
Port C (PC7..PC0)	Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs. Port C also serves the functions of the JTAG interface and other special features of the ATmega16 as listed on page 59.
Port D (PD7..PD0)	Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port D also serves the functions of various special features of the ATmega16 as listed on page 61.
$\overline{\text{RESET}}$	Reset Input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 15 on page 36. Shorter pulses are not guaranteed to generate a reset.
XTAL1	Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.
XTAL2	Output from the inverting Oscillator amplifier.
AVCC	AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.
AREF	AREF is the analog reference pin for the A/D Converter.

13. ANEXO VII

Hoja de especificaciones del foto detector MRD300.

MRD300

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

Photo Detectors
Transistor Output

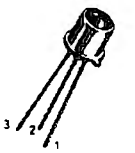
MRD300
MRD310*
*Motorola Preferred Device

PHOTO DETECTORS
TRANSISTOR OUTPUT
NPN SILICON

The MRD300 and MRD310 are designed for applications requiring radiation sensitivity and stable characteristics.

Features:
- Popular TO-18 Type Package for Easy Handling and Mounting
- Sensitive Throughout Visible and Near Infrared Spectral Range for Wider Application
- Minimum Light Current 4 mA at H = 5 mW/cm$_2$ (MRD300)
- External Base for Added Control
- Annular Passivated Structure for Stability and Reliability

Applications:
- Industrial Processing and Control
- Shaft or Position Readers
- Optical Switching
- Remote Control
- Light Modulators
- Punched Card Readers
- Logic Circuits
- Counters



CASE 82-05
METAL
STYLE 1

MAXIMUM RATINGS (T_A = 25°C unless otherwise noted)

MAXIMUM RATINGS (T_A = 25°C unless otherwise noted)

Rating	Symbol	Value	Unit
Collector-Emitter Voltage	V_{CEO}	50	Volts
Emitter-Collector Voltage	V_{ECO}	7	Volts
Collector-Base Voltage	V_{CBO}	80	Volts
Total Device Dissipation @ T_A = 25°C Derate above 25°C	P_D	250 2.27	mW mW/°C
Operating Temperature Range	T_A	−55 to +125	°C
Storage Temperature Range	T_{stg}	−65 to +150	°C

STATIC ELECTRICAL CHARACTERISTICS (T_A = 25°C unless otherwise noted)

Characteristic	Symbol	Min	Typ	Max	Unit
Collector Dark Current (V_{CE} = 20 V, H = 0) T_A = 25°C T_A = 100°C	I_{CEO}	— —	5 4	25 —	nA μA
Collector-Base Breakdown Voltage (I_C = 100 μA)	$V_{(BR)CBO}$	80	120	—	Volts
Collector-Emitter Breakdown Voltage (I_C = 100 μA)	$V_{(BR)CEO}$	50	85	—	Volts
Emitter-Collector Breakdown Voltage (I_E = 100 μA)	$V_{(BR)ECO}$	7	8.5	—	Volts

OPTICAL CHARACTERISTICS (T_A = 25°C unless otherwise noted)

Light Current (V_{CC} = 20 V, R_L = 10 Ohms) Note 1	MRD300 MRD310	I_L	4 1	7 3.5	— —	mA
Light Current (V_{CC} = 20 V, R_L = 100 Ohms) Note 2	MRD300 MRD310	I_L	— —	2.5 0.8	— —	mA
Photo Current Rise Time (Note 3) (R_L = 100 Ohms, I_L = 1 mA peak)		t_r	—	2	2.5	μs
Photo Current Fall Time (Note 3) (R_L = 100 Ohms, I_L = 1 mA peak)		t_f	—	2.5	4	μs

NOTES 1. Radiation flux density (H) equal to 5 mW/cm^2 emitted from a tungsten source at a color temperature of 2870 K
2. Radiation flux density (H) equal to 0.5 mW/cm^2 (pulsed) from a GaAs (gallium-arsenide) source at λ = 940 nm
3. For unsaturated response time measurements, radiation is provided by pulsed GaAs (gallium-arsenide) light-emitting diode (λ = 940 nm) with a pulse width equal to or greater than 10 microseconds (see Figure 2) I_L = 1 mA peak

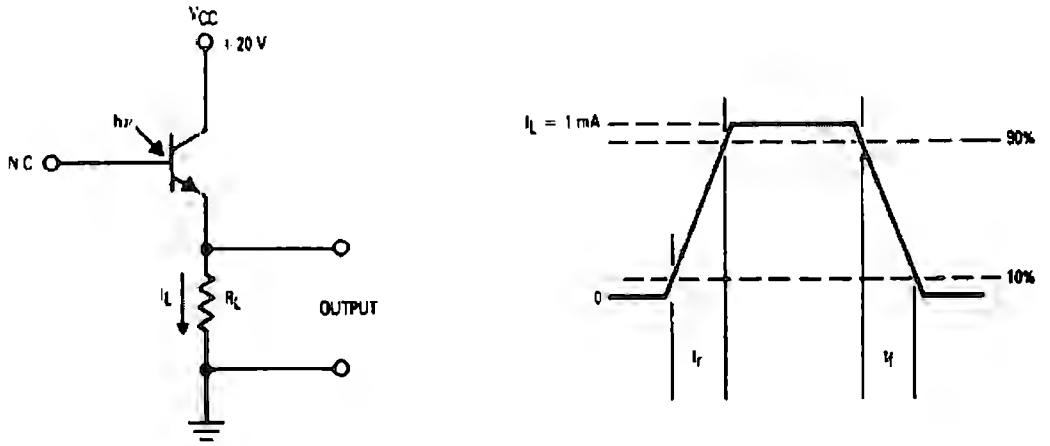


Figure 9. Pulse Response Test Circuit and Waveform

14. ANEXO VIII

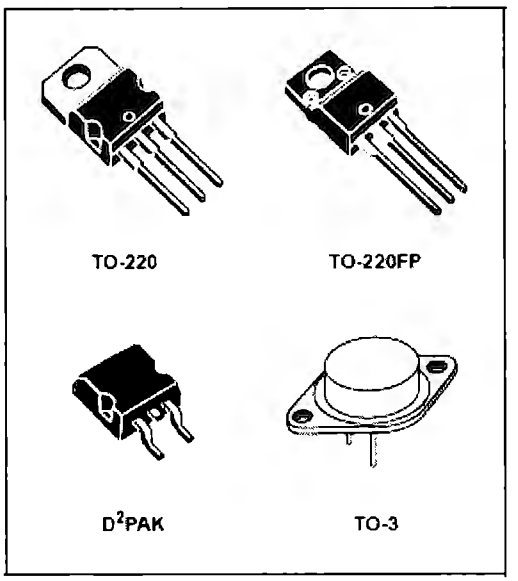
Hoja de especificaciones del regulador de voltaje L7805CV.

L7805CV

L7800 SERIES

POSITIVE VOLTAGE REGULATORS

- OUTPUT CURRENT TO 1.5A
- OUTPUT VOLTAGES OF 5; 5.2; 6; 8; 8.5; 9; 12; 15; 18; 24V
- THERMAL OVERLOAD PROTECTION
- SHORT CIRCUIT PROTECTION
- OUTPUT TRANSITION SOA PROTECTION



DESCRIPTION

The L7800 series of three-terminal positive regulators is available in TO-220, TO-220FP, TO-3 and D^2PAK packages and several fixed output voltages, making it useful in a wide range of applications. These regulators can provide local on-card regulation, eliminating the distribution problems associated with single point regulation. Each type employs internal current limiting, thermal shut-down and safe area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltage and currents.

L7800 SERIES

ABSOLUTE MAXIMUM RATINGS

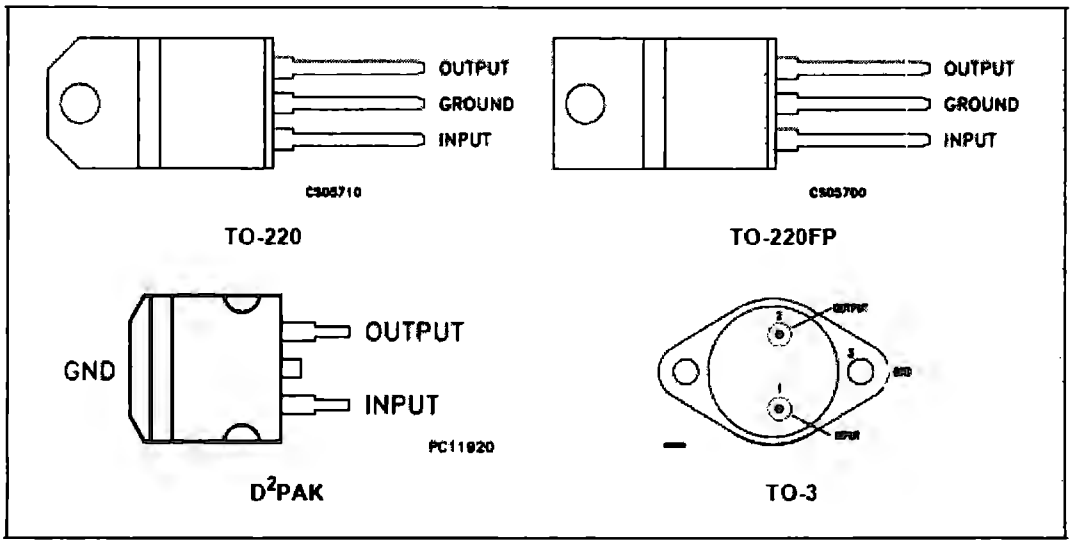
Symbol	Parameter[2]		Value	Unit
V_I	DC Input Voltage	for V_O= 5 to 18V	35	V
		for V_O= 20, 24V	40	
I_O	Output Current		Internally Limited	
P_{tot}	Power Dissipation		Internally Limited	
T_{stg}	Storage Temperature Range		-65 to 150	°C
T_{op}	Operating Junction Temperature Range	for L7800	-55 to 150	°C
		for L7800C	0 to 150	

Absolute Maximum Ratings are those values beyond which damage to the device may occur. Functional operation under these condition is not implied.

THERMAL DATA

Symbol	Parameter	D^2PAK	TO-220	TO-220FP	TO-3	Unit
$R_{th\,j\text{-}case}$	Thermal Resistance Junction-case Max	3	5	5	4	°C/W
$R_{th\,j\text{-}amb}$	Thermal Resistance Junction-ambient Max	62.5	50	60	35	°C/W

CONNECTION DIAGRAM (top view)



15. ANEXO IX

Hoja de especificaciones del Optointerruptor.

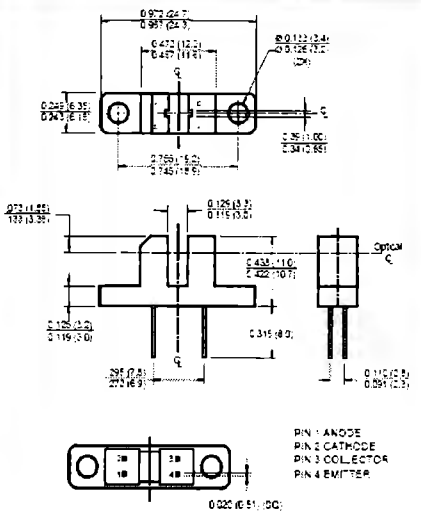
Optointerruptor

FAIRCHILD
SEMICONDUCTOR®

H21A1 / H21A2 / H21A3

PHOTOTRANSISTOR OPTICAL INTERRUPTER SWITCH

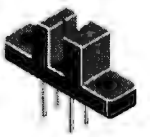
PACKAGE DIMENSIONS



NOTES:

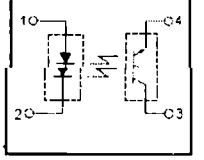
1. Dimensions for all drawings are in inches (mm).
2. Tolerance of ± .010 (.25) on all non-nominal dimensions unless otherwise specified.

DESCRIPTION



The H21A1, H21A2 and H21A3 consist of a gallium arsenide infrared emitting diode coupled with a silicon phototransistor in a plastic housing. The packaging system is designed to optimize the mechanical resolution, coupling efficiency, ambient light rejection, cost and reliability. The gap in the housing provides a means of interrupting the signal with an opaque material, switching the output from an "ON" to an "OFF" state.

SCHEMATIC



FEATURES

- Opaque housing
- Low cost
- .035" apertures
- High $I_{C(ON)}$

1. Derate power dissipation linearly 1.33 mW/°C above 25°C.
2. RMA flux is recommended.
3. Methanol or isopropyl alcohols are recommended as cleaning agents.
4. Soldering iron tip 1/16" (1.6mm) minimum from housing.

ABSOLUTE MAXIMUM RATINGS (T_A = 25°C unless otherwise specified)

Parameter	Symbol	Rating	Unit
Operating Temperature	T_{OPR}	-55 to +100	°C
Storage Temperature	T_{STG}	-55 to +100	°C
Soldering Temperature (Iron)(2,3 and 4)	T_{SOL-I}	240 for 5 sec	°C
Soldering Temperature (Flow)(2 and 3)	T_{SOL-F}	260 for 10 sec	°C
INPUT (EMITTER) Continuous Forward Current	I_F	50	mA
Reverse Voltage	V_R	6	V
Power Dissipation (1)	P_D	100	mW
OUTPUT (SENSOR) Collector to Emitter Voltage	V_{CEO}	30	V
Emitter to Collector Voltage	V_{ECO}	4.5	V
Collector Current	I_C	20	mA
Power Dissipation (T_C = 25°C)(1)	P_D	150	mW

ELECTRICAL / OPTICAL CHARACTERISTICS (T_A =25°C)(All measurements made under pulse condition)							
PARAMETER	**TEST CONDITIONS**	**SYMBOL**	**DEVICES**	**MIN**	**TYP**	**MAX**	**UNITS**
INPUT (EMITTER) Forward Voltage	I_F = 60 mA	V_F	All	—	—	1.7	V
Reverse Breakdown Voltage	I_R = 10 µA	V_R	All	6.0	—	—	V
Reverse Leakage Current	V_R = 3 V	I_R	All	—	—	1.0	µA
OUTPUT (SENSOR) Emitter to Collector Breakdown	I_F = 100 µA, Ee = 0	BV_{ECO}	All	6.0	—	—	V
Collector to Emitter Breakdown	I_C = 1 mA, Ee = 0	BV_{CEO}	All	30	—	—	V
Collector to Emitter Leakage	V_{CE} = 25 V, Ee = 0	I_{CEO}	All	—	—	100	nA
COUPLED On-State Collector Current	I_F = 5 mA, V_{CE} = 5 V	$I_{C(ON)}$	H21A1	0.15	—	—	mA
			H21A2	0.30	—	—	
			H21A3	0.60	—	—	
	I_F = 20 mA, V_{CE} = 5 V		H21A1	1.0	—	—	
			H21A2	2.0	—	—	
			H21A3	4.0	—	—	
	I_F = 30 mA, V_{CE} = 5 V		H21A1	1.9	—	—	
			H21A2	3.0	—	—	
			H21A3	5.5	—	—	
Saturation Voltage	I_F = 20 mA, I_C = 1.8 mA	$V_{CE(SAT)}$	H21A2/3	—	—	0.40	V
	I_F = 30 mA, I_C = 1.8 mA		H21A1	—	—	0.40	V
Turn-On Time	I_F = 30 mA, V_{CC} = 5 V, R_L = 2.5 KΩ	t_{on}	All	—	8	—	µs
Turn-Off Time	I_F = 30 mA, V_{CC} = 5 V, R_L = 2.5 KΩ	t_{off}	All	—	50	—	µs

16. REFERENCIAS

Referencias bibliográficas:

[1] Devendra K., Misra (2004). ***Radio-Frecuency and Microwave Communication Circuits***. New Jersey: John Wiley.

[2] Carr, Joseph J (2002). ***RF Components and Circuits***. Oxford: Newnes.

[3] Blake, Roy (2002). ***Electronic Communication Systems***. Albany, N.Y.: Delmar/Thomson Learning.

[4] Pozar, David M (2001). ***Microwave and RF Wireless Systems***. New York: John Wiley.

[5] Hickman, Ian (1995). ***Manual Práctico de Radiofrecuencia***. Madrid: Paraninfo.

[6] Miguel Rodriguez Gómez-Stern, Marco Antonio Besteiro Gorostizaga (2002). ***Desarrollo de aplicaciones .Net con Visual C#***. Madrid: McGraw-Hill.

[7] Bradley L. Jones (2004). ***Sams teach yourself the C# language in 21 days***. Indianapolis: Sams.

[8] Ceballos Sierra, Fco. Javier (2006). ***Enciclopedia de Microsoft Visual C#***. México: Alfaomega Grupo Editor.

[9] Petzold, Charles (2002). ***Programación en Microsoft Windows con C#***. España: McGraw-Hill.

[10] Price, Jason; Gunderloy, Mike (2002). ***Mastering Visual C# .Net***. London: Sybex.

[11] Robinson, Simon (2004). ***Professional C#***. Indianapolis: Wiley Publishing, Inc.

Referencias de internet:

[12] ***Datos de los puertos***. [En línea] disponible en: http://fox.presidencia.gob.mx
Visitado en marzo de 2007.

[13] ***Radios marca Codan, modelo NGT SR transceiver***. [En línea] disponible en:
http://www.mobilecomms.com.au/Products/Codan/easy.htm
Visitado en marzo de 2007.

[14] ***Usos de HF***. [En línea] disponible en:
http://www.voacap.com/itshfbc-help/icepac-tech-intro.html
Visitado en marzo de 2007

[15] ***ICEPAC***. [En línea] disponible en:

http://www.voacap.com/documents/GLane_ICEPAC.pdf
Visitado en marzo de 2007.

[16] ***Programa HAARP.*** [En línea] disponible en: http://www.haarp.alaska.edu
Visitado en mazo de 2007.

[17] ***SUPERDARN.*** [En línea] disponible en: http://superdarn.jhuapl.edu/
Visitado en marzo 2007.

[18] Goldberg, B. ***HF Radio Data Transmission. U.S. Army Signal Res. And Dev. Lab., Fort Monmouth, NJ, USA.*** Volume: 9, Issue: 1. Páginas 21- 28. [En línea] disponible en: Base de datos de IEEE (Biblioteca digital) Visitado en marzo de 2007.

[19] Itoh, Tatsuo. ***RF Technologies For Low Power Wireless Communications.*** New York: IEEE: Wiley Interscience, 2001. [En línea] disponible en: Base de datos de IEEE (Biblioteca digital) Visitado en marzo de 2007.

[20] ***Ventajas de HF.*** [En línea] disponible en:
http://www.mobilecomms.com.au/Products/Codan/easy.htm
Visitado en marzo de 2007.

[21] ***Desventajas de HF.*** [En línea] disponible en:
http://www.mobilecomms.com.au/Products/Codan/easy.htm
Visitado en marzo de 2007.

[22] ***Características principales de los radios marca Codan, modelo NGT SR transceiver.*** [En línea] disponible en:
http://www.mobilecomms.com.au/Products/Codan/easy.htm
Visitado en marzo de 2007.

[23] ***Características avanzadas de los radios marca Codan, modelo NGT SR transceiver.*** [En línea] disponible en:
http://www.mobilecomms.com.au/Products/Codan/easy.htm
Visitado en marzo de 2007.