Prototipo de Red de Sensores de Temperatura, Utilizando Módulos Xbee, Para Control de Incendios.

3 de diciembre de 2010

Pedro Canché Martín [canche_martin@hotmail.com]
Mario Marín Montoya [marin_mario@hotmail.com]
Universidad del Caribe
Cancún, Quintana Roo

Resumen

En este trabajo se crea un prototipo de red de sensores de temperatura, usando módulos de RF XBee y microprocesadores, integrados en dispositivos de bajo consumo y de baja transferencia de datos.

Palabras clave: Xbee, microcontrolador 16F877A, LM35, Red de sensores inalámbricos.

1. Introducción

El estado mexicano de Quintana Roo, ubicado en la península de Yucatán con una superficie aproximada de 10,200 km2; el 90 % de su superficie está cubierto por la selva y el restante 10 % por manglar.

Las selvas en Quintana Roo año con año se ven amenazados por los incendios, debido a factores meteorológicos como los causados por rayos, la aridez del terreno y por su puesto los descuidos humanos. Los factores meteorológicos condicionan de una forma decisiva el nivel de riesgo del incendio, por ejemplo, las condiciones críticas de altas temperaturas o viento fuerte, aumentan

la posibilidad de incendio forestal y propagarlos con suma facilidad.

Lo incendios forestales muchas veces al no ser detectados a tiempo, se propagan, causando áreas incendiadas más grandes, como en el 2005 con 53,619 hectáreas incendiadas; siendo ese año como el peor de la década, según estadísticas de [1].

Para la detección de los incendios se utilizan diversos medios de detección: Detección terrestre fija, detección terrestre móvil, Detección Aérea, Detección satelital y Detección de puntos de calor. Sin embargo esta última es costosa, debido a que se usa recursos tecnológicos del extranjero, por lo que falta tecnología menos costosa de origen mexicano que representen una valiosa ayuda en la preservación de extensiones de selvas y fauna endémica de la región, así como el impacto producido directamente, no solo en un estado o región, sino a nivel mundial.

Por lo descrito anteriormente, se propone construir un prototipo de red de sensores inalámbricos, basados en un microcontrolador, sensor de temperatura y módulos de Radio Frecuencia (RF) de bajo consumo, con baja tasa de envío de datos y maximizando la vida útil de sus baterías.

2. Sistemas analógicos y digitales: Xbee, microcontrolador 16F877A, LM35

Para la construcción del prototipo que se propone, es necesario conocer los aspectos teóricos del modulo de RF Xbee, microcontrolador 16F877A y del sensor de temperatura LM35.

2.1. *Modulo de RF Xbee*

Xbee y Xbee Pro son módulos de RF fabricado por MaxStream, y son de pequeñas dimensiones; esta creado para satisfacer las necesidades de bajo coste y consumo, exigido en las redes de sensores inalámbricos, y su pequeño tamaño lo hace de fácil integración a un PCB.

Los módulos Xbee proporcionan 2 formas de comunicación: Transmisión serial transparente (modo AT) y el modo API que provee muchas ventajas. Pueden ser configurados desde el PC utilizando el programa X-CTU o bien desde el microcontrolador.

Proveen comunicaciones inalámbricas confiables, ya que operan en la banda ISM 2.4 GHz con protocolo de comunicación 802.15.4 [1] y comunicarse en arquitecturas punto a punto, punto a multipunto o en una red mesh, permitiendo formar hasta 65535 combinaciones distintas de red.

La serie 1 de Xbee soportan topología estrella, son de corto alcance y mantienen un rendimiento inalámbrico excelente, proporcionando un control máximo sobre la red y un mínimo de latencia.

Cuenta con 6 convertidores analógico/digital y 8 entradas digitales además de 2 puertos UART. Las especificaciones y dimensiones del Xbee se encuentran en el anexo.

2.2. Microcontrolador 16F877A

Fabricado por la Familia Microchip[2], posee características que hacen a este microcontrolador un dispositivo muy versátil, eficiente y práctico. Este microcontrolador soporta el modo de comunicación serial, a través de dos pines destinados para este motivo, tiene amplia memoria para datos y programa. La memoria en este Pic es de tipo Flash; este tipo de memoria puede borrarse y reescribirse cuando sea necesario. Posee un set de instrucciones reducidos de tipo RISC, pero con las instrucciones necesarias para facilitar su manejo. Este microcontrolador se programa en lenguaje ensamblador.

2.3. Sensor de Temperatura LM35

Construido por National Semiconductors, el LM35 es un sensor de temperatura integrado en circuito, cuya tensión de salida es linealmente proporcional a la temperatura en grados Celsius, con una precisión calibrada de 1°C y un rango que abarca desde -55° a +150°C.

Por ejemplo:

- +1500mV = 150°C
- +250mV = 25°C
- -550mV = -55°C

[1] véase Apéndice A

[2] Microchip Technology Ic. is a leading provider of micro controller and analogue semiconductors

El LM35 funciona en el rango de alimentación comprendido entre 4 y 30 voltios. Se puede conectar a un conversor Analógico/Digital y tratar la medida digitalmente, almacenarla o procesarla con un microntrolador o similar.

Principales características se lista a continuación:

- Calibrado directamente en grados centígrados
- Factor de escala lineal +10.0 mV
- 0.5ºC de exactitud garantizable (a 25ºC)
- Adecuado para aplicaciones remotas
- Funciona desde 4 hasta 30 volts
- Menos de 60 mA de Corriente
- Bajo calentamiento, 0.8 ºC al aire libre
- Baja impedancia de salida, 0.1 W de carga de 1 mA

3. Generalidades del prototipo

El prototipo propuesto consta de 5 nodos: un nodo maestro o coordinador, y 4 nodos terminales o esclavos, que se comunican inalámbricamente y una computadora conectada a través del puerto serial con el nodo maestro.

La Figura 1 esquematiza el prototipo del proyecto, propuesto.

3.1. Topología del proyecto

Teniendo en cuenta distintas limitaciones como la accesibilidad y costo para conseguir los dispositivos y herramientas necesarias (Xbee, microcontrolador, programador, software, etc.); y al ser un modelo prototipo, se decidió demostrar la factibilidad y sencillez de formar una red a través de su implementación básica, por lo que se ha escogido la topología estrella.

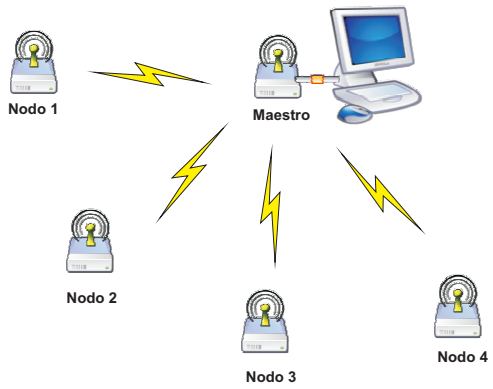

Figura 1: Representación del proyecto

3.2. Diseño del Hardware

El diseño del hardware como en cualquier prototipo, no puede faltar. A continuación se describen los diseños del hardware, del nodo maestro y esclavo requerido para el funcionamiento del prototipo de adquisición de datos de temperatura.

El prototipo de la red de sensores consta de dos partes.

3.2.1. Nodo Maestro

Esta parte tiene la tarea de mantener el enlace con el resto de los nodos; se encarga de enviar y recibir información de los nodos esclavos y este nodo a su vez enviarlo a la computadora para su visualización.
Este nodo consta de un microcontrolador que es el que realiza las peticiones, configura en tiempo real el modulo Xbee; para la comunicación serial se utiliza un conversor MAX232 para lograr la comunicación entre el nodo maestro y la computadora, para el envío de los datos.
Además complementan al nodo los siguientes componentes

- Led's indicadores de actividad
- Zumbador
- Display de 2x16

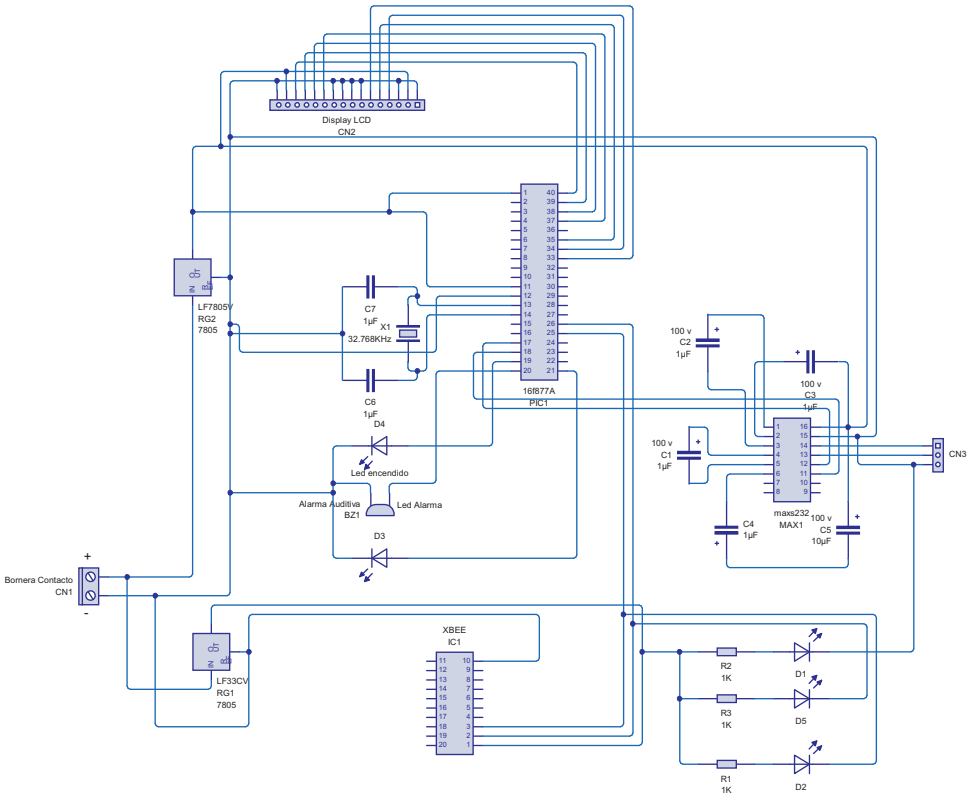

Figura 2: Diagrama de circuitos del nodo Maestro

3.2.1.1 Diagrama de circuito del nodo maestro El diagrama del circuito, está construido de acuerdo a las especificaciones de los fabricantes de los diversos componentes que la forman, tal como se muestra en la Figura 2.

3.2.2. Nodo Esclavo

Esta parte está formado por el modulo Xbee que en conjunto con el microcontrolador, responde la solicitud de temperatura hecha por el maestro. Este dispone del sensor de temperatura por el cual se obtienen los datos a enviar. Le complementa este circuito led's indicadores de actividad.

3.2.2.1 Diagrama de circuito del nodo esclavo Está construido de acuerdo a las especificaciones de los fabricantes de los diversos componentes que la forman, tal como se muestra en la Figura 3.

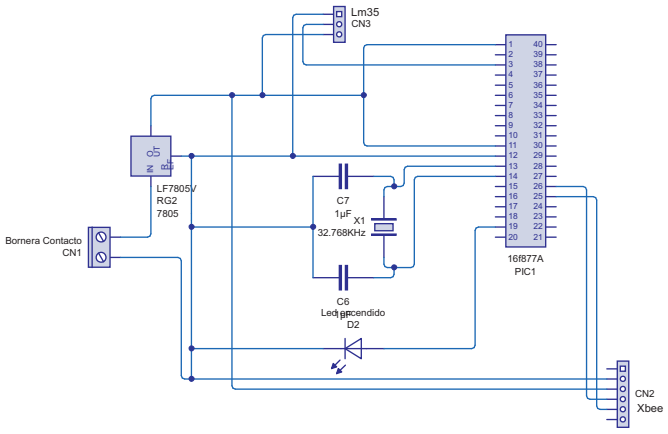

Figura 3: Diagrama de circuitos del nodo Maestro

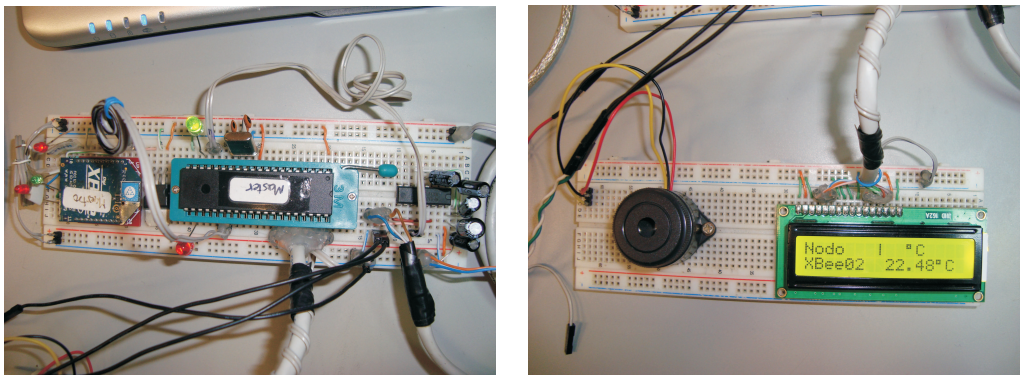

Figura 4: Diagrama de circuitos del nodo Maestro

3.2.3. Prototipo sobre protoboard

En la Figura 4 y 5, se muestran los prototipos montados sobre placas protoboard, con los componentes necesarios para su correcto funcionamiento.

3.3. Entorno de desarrollo del microcontrolador

El código de funcionamiento del microcontrolador fue desarrollado usando el software PIC C Compiler CCS PCDWHD, que combina el uso del lenguaje estructurado C, con salida en formato HEX soportado por el pic.

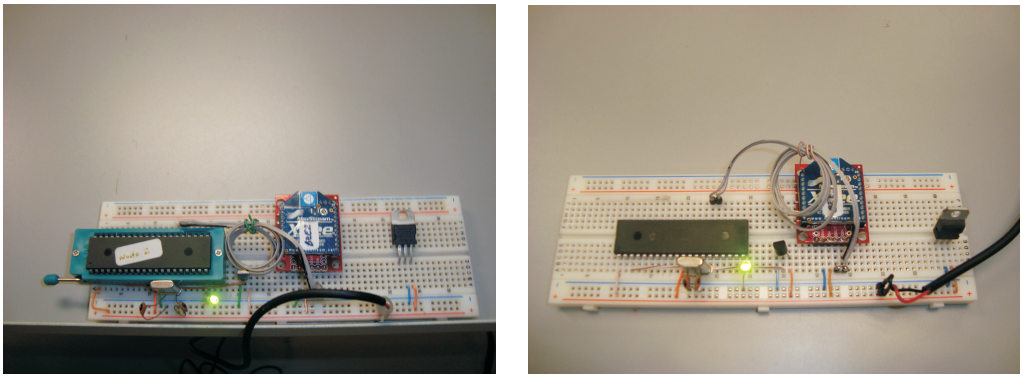

Figura 5: Diagrama de circuitos del nodo Esclavo

3.3.1. El programa

El programa que se aloja en el pic del nodo maestro, debe cumplir con los siguientes requisitos:

1. Al conectarse el nodo maestro a la computadora a través del puerto serial, el programa que reside en el pic, debe manejar envíos continuos de datos, a través de la salida del pic habilitada por software.

2. Al mismo tiempo, este programa debe configurar en tiempo real los valores del Xbee, para una comunicación transparente con el resto de los nodos. Mediante el uso de comandos AT. Como por ejemplo, la configuración del Canal, del Pan ID, el nombre del Xbee destino, entre otros. Si la configuración no resulto con éxito, debe enviar un mensaje de error.

3. El programa también debe solicitar datos de modo inalámbrico a los nodos esclavos enviando una cadena que cada nodo es capaz de reconocer.

4. 4. Enviar una alarma de sonido cuando un nodo no responda.

Para el caso del nodo esclavo, el programa alojado en el pic, debe cumplir con los siguientes requisitos:

1. Continuamente obtener datos del sensor de temperatura y hacer una conversión analógica digital.

2. Permanecer en escucha de la petición del nodo maestro, y responder esta.

El programa generado para cada uno de los nodos, se pueden ver en el Apéndice D y Apéndice E.

Finalmente, todos los datos que sean recibidos por el puerto serial de la computadora serán mostrados en una interfaz gráfica sobre un navegador en internet, para ser vista desde cualquier ubicación.

4. Resultados

Una vez que el prototipo está construido y que la programación este lista, el siguiente paso es probar si funciona.

4.1. Monitorización mediante terminal X-CTU e hyperterminal

Para hacer las pruebas de funcionamiento, se hace uso del terminal que viene en el software X-CTU. Una vez instalado y abierto el software, este es capaz de detectar el puerto disponible por el cual estamos conectados al nodo maestro. Una vez en el terminal, se enciende el circuito del nodo maestro junto con el resto de los nodos esclavos.

Una vez encendido todos los nodos, comprobamos que nodos esclavos están respondiendo, mediante la cadena nombre=nombre_nodo& temperatura=grados_centigrados, si un nodo no responde, la cadena es nombre=nombre_nodo& temperatura=-99.99, indicando un error.

La configuración para la transmisión de datos entre el nodo maestro y el hyperterminal, es de acuerdo a la configuración que aparece en el Cuadro 1.

En la Figura 6 podemos observar los datos que vamos recibiendo desde los nodos esclavos.

Cuando un dato no es recibido, se activa la alarma auditiva y visual de un led rojo, por espacio de un segundo.

4.2. Monitorización mediante la interfaz

Los datos que estamos obteniendo, y que hemos visto mediante el terminal, ahora serán canalizados a la interfaz web, para que sean desplegados, de

	valor
Bits por segundo:	9600
Bits de datos:	8
Paridad:	Ninguna
Bits de parada:	1
Control de flujo:	Ninguna

Cuadro 1: Datos para configuración de hyperterminal, para comunicación serial con el XBee

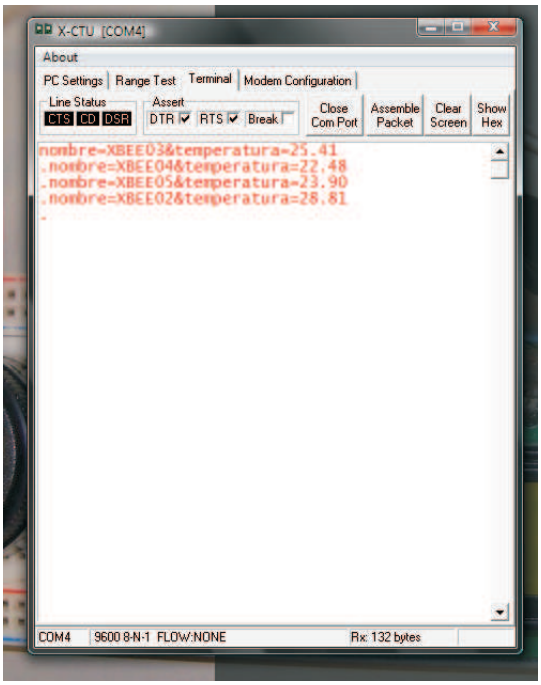

Figura 6: Salida a la terminal de X-CTU

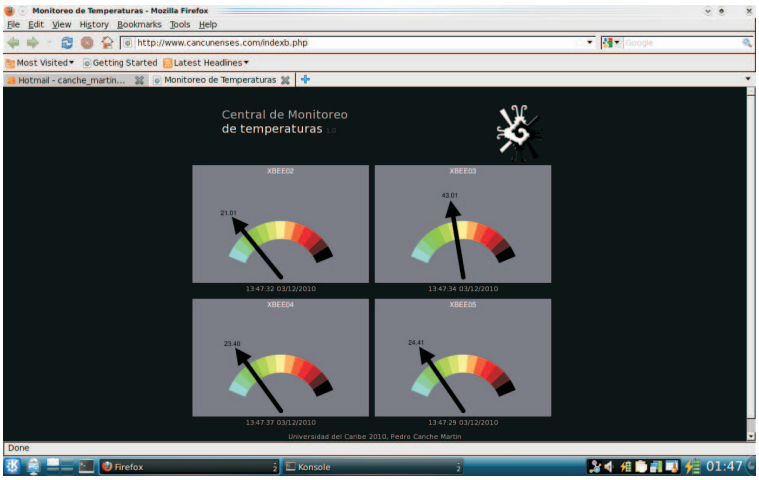

Figura 7: Salida a la terminal de X-CTU

tal modo que el usuario pueda identificarlo con suma facilidad, obteniendo los siguientes resultados, tal como se muestran en la figura 7.

5. Conclusiones

Se corroboró la factibilidad de la implementación Prototipo de red de sensores de temperatura, utilizando módulos Xbee, para la adquisición de datos de temperatura. Los resultados obtenidos tienen una mayor eficiencia debido a que pueden ser leídos desde cualquier, utilizando un explorador de internet, sin la necesidad de estar presente en el lugar donde son generados los datos.

Con las pruebas realizadas, se pudo comprobar que el uso de aparatos de enlace inalámbricos o de microondas no afectó el desempeño del prototipo, pues no presento perdidas de datos o señal.

Una vez construido el prototipo, se puede concluir que los microcontroladores y los Xbee son dispositivos muy sencillos, cuyas dimensiones, potencia y costos reducidos, sientan la base para otros proyectos similares, usando módulos de RF.

5.1. Trabajo futuro

El prototipo construido, es tan solo la base, de un proyecto más grande en el futuro, como las que se lista a continuación:

- Habilitar Comunicación GPRS
- Añadir al prototipo celdas fotovoltaicas, en sustitución de pilas alcalinas
- Implementación de redes Mesh
- Reducción de tamaño en los dispositivos
- Añadir más sensores a los nodos esclavos

Referencias

[1] Secretaría de Medio Ambiente y Recursos Naturales. http://www.semarnat.gob.mx

[2] United States Environment Protection Agency. http://www.epa.gov/iaq/voc.html

[3] Madruga, F. (2006) Contribución al desarrollo de sensores de temperatura y redes de sensores en tecnología de fibra óptica (Tesis Doctoral, Universidad de Cantabría, 2006).

[4] Barneda, I. F. (2008) ZigBee Aplicado a la transmisión de datos de sensores biomédicos (Memoria del proyecto de Ingeniería Técnica en Telecomunicaciones especialidad en sistemas electrónicos, Universitat Autonoma de Barcelona, 2008).

[5] Ramos, P. F. Redes de sensores inalámbricos (Profesor titular de la Universidad Politécnica de Valencia) [http://www.radiooptica.com]

[6] Redes de inalambricas de sensores para la detección de incendios forestales y monitoreo de variables de estado de combustible para el recurso forestal de la región de Valparaíso. http://www.gisincendiosforestales.cl/

[7] Product Manual v1.xAx - 802.15.4 Protocol for OEM RF ModulePart Numbers: XB24-...-001,XBP24-...-001 IEEE 802.15.4 OEM RF Modules byMaxStream

[8] Wi-Fi. Cómo construir una red inalámbrica. José A. Carballar. Ed. RaMa. 2003

[9] IEEE. A non-profit organization, IEEE is the world's largest professional association for the advancement of technology.© Copyright 2010

[10] Valverde Rebaza, Jorge C.(2007) El Estándar inalámbrico ZigBee. Universidad Nacional de Trujillo, Perú.

[11] Valdés Pérez, Fernando, Pallás Areny, Ramon. Microcontroladores: Fundamentos y aplicaciones con PIC. Ed. Marcombo S.A. de C.V. Madrid, España. 2007

[12] Microchip. PIC16F87XA Data Sheet 28/40/44Pin Enhaced Flash Microcontrollers

[13] Datasheet by National Semiconductor. LM35 Precision Centigrade Temperature Sensors. November 2000

Apéndice A. Especificaciones Xbee

Las especificaciones del Xbee, de acuerdo a su fabricante, se muestra en la Tabla

Especificación	Xbee
Rendimiento	
Rango Indoor/Urban	Mayor a 30 mts
Rango de linea de vista Outdoor	Mayor de 100 mts.
Potencia de transmisión	1mW
Tasa de Transmisión	250,000 bps
Tasa de transferencia interface serial (software seleccionable)	1200 - 115200 bps
Sensibilidad de recepción	-92 dBm (Tasa de error 1 %)
Requerimientos de Energía	
Voltaje	2.8-3.4 V
Transmisión típica	45 mA (@3.3V)
Recepción típica	50 mA (@3.3V)
Apagado	Menor a 10 μA
Generales	
Frecuencia de operación	ISM 2.4 GHz
Dimensiones	0.960"x 1.087"(2.438 cm x 2.761 cm)
Temperatura de operación	-40 a 85°C (Industrial)
Opciones de antena	Whip integrado, chip o conectro U. FL
Seguridad y Red	
Topologias de red	Punto a punto, Punto a multipunto y Peer to peer
Número de canales	16 Direct Sequence Channels
Opciones de direcciones	PAN ID, Canales y Direcciones

Especificaciones para el modulo Xbee

Apéndice B. Especificaciones microcontrolador 16F877A

Las especificaciones del microcontrolador, de acuerdo a su fabricante, se muestra en la siguiente tabla:

Características	Pic 16F877A
Frecuencia máxima	20 MHz
Memoria de programa flash	8Kb
RAM de datos	368
EEPRON de datos	256
Puertos E/S	A,B,C,D,E
Número de pines	40
Interrupciones	14
Timers	3
Modulos CCP	2
Comunicaciones serie	MSSP y USART
Comunicaciones paralelo	PSP
Lineas de Etrada de 10 bits	8
Juego de instrucciones	35 instrucciones
Longitud de la instrucción	14 bits
Arquitectura	HARVARD
CPU	RISC
Canales PWM	2

Características más relevantes

Apéndice C. Estándar 802.15.4

IEEE 802.15.41 es la base sobre la que se define la especificación de ZigBee, cuyo propósito es ofrecer una solución completa para este tipo de redes construyendo los niveles superiores de la pila de protocolos que el estándar no cubre.

El propósito del estándar es definir los niveles de red básicos para dar servicio a un tipo específico de red inalámbrica de área personal (WPAN) centrada en la habilitación de comunicación entre dispositivos con bajo coste y velocidad. Se enfatiza el bajo coste de comunicación con nodos cercanos y sin infraestructura, para favorecer aún más el bajo consumo. Para hacer una consulta más detallada, consultar en `http://standards.ieee.org/index.html`.

Apéndice D. Código fuente pic Maestro

```
#IF defined(__PCM__)   //directiva de preprocesador
//Codigo de configuración de funcionamiento
#use delay(clock=4000000)

#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8,stream=Xbee232)
#USE RS232(baud=9600,parity=N,xmit=PIN_C0,rcv=PIN_C1,bits=8,stream=p2,FORCE_SW)
#USE RS232(baud=9600,parity=N,xmit=PIN_C2,rcv=PIN_C3,bits=8,stream=pc232,FORCE_SW)

#ENDIF

#include <stdlib.h>

#define use_portb_lcd //definimos que queremos que salga pór el puerto B
#include <lcd.c>


char rcvchar;
char buffer[10];
int index=0;
int bool=0; //0 -> falso
          //1 -> verdadero


char ok[]={"OK"};
 int *q=1;
//int to=0;
//char timeout=0;


#int_RDA
void RDA_isr()
{
  while(rcvchar!=0x0d)
  {
    rcvchar=fgetc(Xbee232);
    fputc(rcvchar,p2);
    buffer[index]=rcvchar;
    index++;
  }
  rcvchar='\0';
  bool=1;
}

void Xbee_res()
{
  while(!bool)
  {  }
  bool=0;
}

int *Xbee_response(int16 tmo)
{
  while(!bool)
  {
    if(tmo)
    {
      fprintf(p2,"tmo vale antes de la resta %Lu\r",tmo);
      tmo--;
      if(!tmo)
      {
        q=0;
        bool=1;
        return (q);
      }
    }
  }
  bool=0;
  q=1;
  return (q);
}

void configura(int b)
{
    int error=0;
    char *p;

    fprintf(Xbee232,"+++");
    Xbee_res();
    p=strcmp(buffer,ok);
    if(!p)
    {
      bool=0;
      index=0;
    //aqui cambio el dato de mi xbee
      fprintf(Xbee232,"ATDLBEE%u \r",b);
      Xbee_res();
      p=strcmp(buffer,ok);
      if(!p)
      {
        error++;
        fprintf(pc232,"error ATDL\r");
      }
      bool=0;
      index=0;
      //aqui cambio el dato de mi xbee
      fprintf(Xbee232,"ATMYBEE1 \r");
      Xbee_res();
      p=strcmp(buffer,ok);
      if(!p)
      {
        error++;
        fprintf(pc232,"error ATMY\r");
      }

      bool=0;
      index=0;
      //aqui cambio el dato de mi xbee
      fprintf(Xbee232,"ATCHF \r");
      Xbee_res();
      p=strcmp(buffer,ok);
      if(!p)
      {
        error++;
        fprintf(pc232,"error ATCH\r");
      }

      bool=0;
      index=0;
       //aqui cambio el dato de mi xbee
      fprintf(Xbee232,"ATID3332 \r");
      Xbee_res();
      p=strcmp(buffer,ok);
      if(!p)
      {
        error++;
        fprintf(pc232,"error ATID\r");
      }

      bool=0;
      index=0;

      fprintf(Xbee232,"ATWR \r");
      Xbee_res();
      p=strcmp(buffer,ok);
      if(!p)
      {
        error++;
        fprintf(pc232,"error ATWR\r");
      }
      bool=0;
      index=0;

      fprintf(Xbee232,"ATCN \r");
      Xbee_res();
      p=strcmp(buffer,ok);
      if(!p)
      {
        error++;
        fprintf(pc232,"error ATCN\r");
      }
       bool=0;
       index=0;
    }
    else
    {
              fprintf(pc232,"no esta disponible el dispositivo:\r");
    }
}

char *quita_enter(char *buf)
{
  char *cadena;
  int i=0;
  while(buf[i]!=0x0D)
  {
    cadena[i]=buf[i];
    i++;
  }
  return cadena;
}


void LCD_Clear()
{
  lcd_putc("\f");
}

void main()
{
  int *q;
  int i=2;
  char *temperatura;
  LCD_Clear();
  lcd_init();
  lcd_gotoxy(1,1);
  lcd_putc("Red de Sensores");
  lcd_gotoxy(1,2);
  lcd_putc("P. Terminal");
  enable_interrupts(INT_RDA); //TENEMOS QUE HABILITAR LAS
INTERRUPCIONES PARA QUE FUNCIONE EL RDA
  enable_interrupts(GLOBAL);
  delay_ms(1000);
  output_high(PIN_D0);
  bool=0;
  delay_ms(100);

  while(true)
  {
    //corregimos el indice del xbee
    if(i==6)
    {
      i=2;
    }
    configura(i);
    fprintf(Xbee232,"gettemp \r");
    q=Xbee_response(20);
    temperatura=quita_enter(buffer);
    LCD_Clear();
    lcd_gotoxy(1,1);
    printf(lcd_putc,"Nodo   | Temp. ");

    if(q==1)
    {
      fprintf(pc232,"nombre=XBEE0%u&",i);
      fprintf(pc232,"temperatura=%s\r\n",temperatura);
      lcd_gotoxy(1,2);
      printf(lcd_putc,"XBee0%u  %s%CC",i,temperatura,0xDF);
    }
    else
    {
      fprintf(pc232,"nombre=XBEE0%u&",i);
      fprintf(pc232,"temperatura=-99.99\r\n");
      lcd_gotoxy(1,2);
      printf(lcd_putc,"XBee0%u  -99.99%CC",i,0xDF);
      output_high(PIN_D1);
      output_high(PIN_D2);
    }
    bool=0;
    index=0;
    delay_ms(1000);
    output_low(PIN_D1);
    output_low(PIN_D2);
    i++;
  }
}
```

Apéndice E. Código fuente pic esclavo

```
#IF defined(__PCM__)   //directiva de preprocesador
#include <16F877A.h>
#device adc=10 // convertimos a 10bits

#FUSES NOWDT,XT,NOPUT,NOPROTECT,NODEBUG,NOBROWNOUT,NOLVP,NOCPD,WRT_50%,HS

#use delay(clock=4000000)


#ENDIF

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <lcd.c>
#define use_portb_lcd //definimos que queremos que
salga pór el puerto B

#use
rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7
,bits=8,stream=Xbee232)
#use
rs232(baud=9600,parity=N,xmit=PIN_C2,rcv=PIN_C3
,bits=8,stream=p2,FORCE_SW)
/*Zona de declaración de variables de temperatura*/
   float valor=0.0;
   float const resoluc= (5.0/1023.0)*100;      //
Conversión es con 10 bits
   float temperatura=0.0;
/*declaraciones de la entrada por interrupcion*/
char rcvchar;
char buffer[10];
int index=0;
int bool=0; //0 -> falso
           //1 -> verdadero
char get[]={"gettemp"};

#int_RDA
void RDA_isr()
{
   while(rcvchar!=0x0d)
   {
      rcvchar=fgetc(Xbee232);
      fputc(rcvchar,p2);
      buffer[index]=rcvchar;
      index++;
   }
   rcvchar='\0';
   bool=1;
}

void Xbee_res()
{
   while(!bool)
   {

   }
   bool=0;
}


float sensor_Temp()
{
      valor = read_adc();
      temperatura= valor * resoluc;
      return temperatura;
}

void main()
{
   char *p;
   enable_interrupts(INT_RDA); //TENEMOS
QUE HABILITAR LAS INTERRUPCIONES PARA
QUE FUNCIONE EL RDA
   enable_interrupts(GLOBAL);
   setup_adc_ports(RA0_ANALOG);//definimos
que A0 || all_analog todos analogos
   setup_adc(ADC_CLOCK_INTERNAL);    //
Reloj interno para el ADC
   set_adc_channel(1);              // Canal 0 para
convertir

  output_high(PIN_D0);


   while(true)
   {
      Xbee_res();
      p=strcmp(buffer,get);
      fprintf(p2,"p=%s \r",p);

      if(p)
      {
         fprintf(p2,"La condicion es verdadera");
         fprintf(Xbee232,"%f\r",sensor_Temp());
         bool=0;
         index=0;
      }
   }
}
```