Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

USO DEL PROTOCOLO 1-WIRE CON COMPROBACIÓN DE REDUNDANCIA CÍCLICA APLICADO A LA MEDICIÓN DE TEMPERATURA Y CONTROL DE ACCESO

Mynor Geovanny Mendoza Ordóñez
Asesorado por el Ing. Guillermo Antonio Puente Romero

Guatemala, enero de 2015

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

USO DEL PROTOCOLO 1-WIRE CON COMPROBACIÓN DE REDUNDANCIA CÍCLICA APLICADO A LA MEDICIÓN DE TEMPERATURA Y CONTROL DE ACCESO

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

MYNOR GEOVANNY MENDOZA ORDÓÑEZ
ASESORADO POR EL ING. GUILLERMO ANTONIO PUENTE ROMERO

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO ELECTRÓNICO

GUATEMALA, ENERO DE 2015

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Angel Roberto Sic García
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Inga. Elvia Miriam Ruballos Samayoa
VOCAL IV	Br. Narda Lucía Pacay Barrientos
VOCAL V	Br. Walter Rafael Véliz Muñoz
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	PhD. Enrique Edmundo Ruiz Carballo
EXAMINADOR	Ing. Otto Fernando Andrino González
EXAMINADOR	Ing. Julio César Solares Peñate
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

USO DEL PROTOCOLO 1-WIRE CON COMPROBACIÓN DE REDUNDANCIA CÍCLICA APLICADO A LA MEDICIÓN DE TEMPERATURA Y CONTROL DE ACCESO

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha 8 de agosto de 2014.

Mynor Geovanny Mendoza Ordóñez

Guatemala, 10 de noviembre de 2014.

Ing. Carlos Eduardo Guzmán Salazar
Coordinador de Área de Electrónica
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC.

Ingeniero Guzmán:

Por este medio me permito dar aprobación al Trabajo de Graduación titulado: **"USO DEL PROTOCOLO 1-WIRE CON COMPROBACIÓN DE REDUNDANCIA CÍCLICA APLICADO A LA MEDICIÓN DE TEMPERATURA Y CONTROL DE ACCESO"**, desarrollado por el estudiante Mynor Geovanny Mendoza Ordoñez con carné No. 2009-14992, ya que considero que cumple con los requisitos establecidos, por lo que el autor y mi persona somos responsables del contenido y conclusiones del mismo.

Sin otro particular, aprovecho la oportunidad para saludarlo.

Atentamente,

Ing. Guillermo Antonio Puente Romero
ASESOR
Colegiado 5898

Guillermo A. Puente R.
INGENIERO ELECTRONICO
COL. # 5898

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

Ref. EIME 58 2014
Guatemala, 13 de NOVIEMBRE 2014.

Señor Director
Ing. Guillermo Antonio Puente Romero
Escuela de Ingenieria Mecànica Elèctrica
Facultad de Ingenieria, USAC.

Señor Director:

Me permito dar aprobación al trabajo de Graduación titulado: USO DEL PROTOCOLO 1-WIRE CON COMPROBACIÓN DE REDUNDANCIA CÍCLICA APLICADO A LA MEDICIÓN DE TEMPERATURA Y CONTROL DE ACCESO, del estudiante, Mynor Geovanny Mendoza Ordóñez , que cumple con los requisitos establecidos para tal fin.

Sin otro particular, aprovecho la oportunidad para saludarle.

Atentamente,
ID Y ENSEÑAD A TODOS

Ing. Carlos Eduardo Guzmán Salazar
Coordinador Área Electrónica

SIO

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

REF. EIME 58. 2014.

El Director de la Escuela de Ingeniería Mecánica Eléctrica, después de conocer el dictamen del Asesor, con el Visto Bueno del Coordinador de Área, al trabajo de Graduación del estudiante; MYNOR GEOVANNY MENDOZA ORDÓÑEZ titulado: USO DEL PROTOCOLO 1-WIRE CON COMPROBACIÓN DE REDUNDANCIA CÍCLICA APLICADO A LA MEDICIÓN DE TEMPERATURA Y CONTROL DE ACCESO, procede a la autorización del mismo.

Ing. Guillermo Antonio Puente Romero

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

GUATEMALA, 20 DE NOVIEMBRE 2,014.

Universidad de San Carlos
De Guatemala

Facultad de Ingeniería
Decanato

DTG. 014.2015

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica, al Trabajo de Graduación titulado: **USO DEL PROTOCOLO 1-WIRE CON COMPROBACIÓN DE REDUNDANCIA CÍCLICA APLICADO A LA MEDICIÓN DE TEMPERATURA Y CONTROL DE ACCESO,** presentado por el estudiante universitario **Mynor Geovanny Mendoza Ordoñez,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, se autoriza la impresión del mismo.

IMPRÍMASE:

Ing. Murphy Olympo Paiz Recinos
Decano

Guatemala, 23 de enero de 2015

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERÍA

/gdech

ACTO QUE DEDICO A:

Dios	Por darme las oportunidades que me llevaron hasta este triunfo y estar siempre a mi lado. A Dios sea la gloria.
Mi madre	Carmen Ordóñez, por ser el mejor ejemplo de fortaleza, paciencia, disciplina y amor que Dios ha puesto en mí camino y, por nunca dejar de creer en mí.
Mis sobrinas	Zain y Sharon, por darle alegría a la familia y darme un motivo para querer ser una mejor persona cada día.

AGRADECIMIENTOS A:

La Universidad de San Carlos de Guatemala	Por abrirme sus puertas y darme la oportunidad de culminar exitosamente mis estudios universitarios.
Facultad de Ingeniería	Por darme la oportunidad de formarme como ingeniero electrónico.
Startrack S. A.	Por el apoyo brindado en mi formación profesional.
Ing. Guillermo Puente	Por su tiempo y conocimiento como profesional para el asesoramiento en la elaboración del presente documento.
Didier Tenas	Por ser una persona de excelencia y por su apoyo en el transcurso de la carrera.
Amigos y compañeros de proyectos	Por su grata compañía en el transcurso de la carrera, y por el apoyo brindado en cada proyecto que logramos desarrollar.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
A	Amperio
°C	Grado Celsius
kΩ	Kilo-ohm
Mhz	Megahertz
mW	Micro Watt
µC	Microcontrolador
µF	Microfaradio
µs	Microsegundo
mA	Miliamperio
ms	Milisegundo
nF	Nano faradio
Ω	Ohm
Rx	Receptor
Tx	Transmisor
V	Volt

GLOSARIO

ABS Sistema antibloqueo de ruedas. Dispositivo que hace variar la fuerza de frenado de un vehículo, para evitar que los neumáticos pierdan la adherencia con el suelo.

BASIC Código simbólico de instrucciones de propósito general para principiantes, es una familia de lenguajes de programación de alto nivel.

CAN Controller area network. Protocolo de comunicaciones, serie que soporta control distribuido en tiempo real con un alto nivel de seguridad y multiplexación.

CCP Capture/Compare/PWM. Módulo del microcontrolador PIC, se asocia con el registro de control CCPxCON y un registro de datos CCPRx.

CI Circuito integrado. Pastilla pequeña de material semiconductor sobre la cual se fabrican circuitos electrónicos, generalmente mediante fotolitografía y que está protegida dentro de un encapsulado de plástico o cerámica.

Cifrado	Método que permite aumentar la seguridad de un mensaje o de un archivo mediante la codificación del contenido, de manera que solo pueda leerlo la persona que cuente con la clave de cifrado adecuada para descodificarlo.
Codec	Codificador-decodificador. Describe una especificación desarrollada en software, hardware o una combinación de ambos, capaz de transformar un archivo con un flujo de datos o una señal.
CPU	Unidad central de procesamiento. Hardware dentro de un computador u otros dispositivos programables, que interpreta las instrucciones de un programa de ordenador mediante la realización de las operaciones básicas aritméticas, lógicas y de entrada/salida del sistema.
CRC	Verificación por redundancia cíclica. Código de detección de errores usado para detectar cambios en los datos.
DC	Corriente continua. Se refiere al flujo continuo de carga eléctrica a través de un conductor entre dos puntos de distinto potencial, que no cambia de sentido con el tiempo.

DOW CRC	Dallas One Wire CRC. Método de detección de errores usado por los dispositivos fabricados por Dallas semiconductor.
DSP	Procesamiento digital de señales. Manipulación matemática de una señal de información para modificarla o mejorarla en algún sentido.
EEPROM	ROM programable y borrada eléctricamente. Tipo de memoria ROM que puede ser programada, borrada y reprogramada eléctricamente.
EPROM	ROM programable borrable. Tipo de memoria ROM no volátil que puede borrarse solamente mediante exposición a una fuerte luz ultravioleta.
Estándar	Proceso, protocolo o técnica utilizada para hacer algo concreto.
Firmware	Bloque de instrucciones de máquina para propósitos específicos grabado en una memoria, que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo. Es un software que maneja físicamente al hardware.
Half-Duplex	Conexión en la que los datos fluyen en una u otra dirección, pero no las dos el mismo tiempo.

Handshaking	Proceso automatizado de negociación que establece de forma dinámica los parámetros de un canal de comunicaciones establecido entre dos entidades antes de que comience la comunicación normal por el canal.
Hardware	Partes tangibles de un sistema informático.
HVAC	Calefacción, ventilación y aire acondicionado. Sistema de control que se aplica a la regulación de un sistema de calefacción y/o aire acondicionado.
ID	Símbolo que identifica de forma única un objeto o registro.
ISO	Organización internacional de normalización. Organismo encargado de promover el desarrollo de normas internacionales de fabricación, comercio y comunicación para todas las ramas industriales a excepción de la eléctrica y la electrónica.
Kbps	Unidad de velocidad de transferencia de datos igual a 1000 bits por segundo.
LAN	Red de área local. Red informática de computadoras interconectadas dentro de un área limitada.

LCD	Pantalla de cristal líquido. Pantalla delgada y plana formada por un número de pixeles en color o monocromos colocados delante de una fuente de luz o reflectora.
LSB	Bit menos significativo. Posición de bit en un número binario que tiene el menor valor.
MicroLAN™	Red de dispositivos que se conectan mediante el protocolo 1-Wire.
MSB	Bit más significativo. Posición de bit en un número binario que tiene el mayor valor.
OR	Puerta lógica digital que implementa la disyunción lógica.
OSI	Interconexión de sistemas abiertos. Modelo de red descriptivo creado por la ISO. Es un marco de referencia para la definición de arquitecturas en la interconexión de los sistemas de comunicaciones.
PCB	Printed circuit board. Superficie constituida por caminos o pistas de material conductor laminadas sobre una base no conductora.

ROM	Memoria de solo lectura. Medio de almacenamiento utilizado en ordenadores y dispositivos electrónicos, que permite solo la lectura de la información y no su escritura, independientemente de la presencia o no de una fuente de energía
Software	Equipamiento lógico o soporte lógico de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas.
T	Temperatura. Registro de la memoria de trabajo que consta de 16 bits, dividida en TH y TL, donde se guarda la información de la conversión hecha por el dispositivo cuando se realiza la medición de temperatura.
TH	Temperatura alta. Parte del registro de la memoria de trabajo de los dispositivos esclavos 1-Wire capaz de realizar mediciones de temperatura, donde se guarda los 8 bits más significativos del resultado de la conversión.
TL	Temperatura baja. Parte del registro de la memoria de trabajo de los dispositivos esclavos 1-Wire capaz de realizar mediciones de temperatura, donde se guarda los 8 bits menos significativos del resultado de la conversión.

RESUMEN

El presente trabajo de graduación pretende explicar detalladamente el uso del protocolo 1-Wire y demostrar su uso mediante la implementación en la resolución de un problema común en el trasporte y almacenaje de productos en cuartos refrigerados: la medición de temperatura y el control de acceso.

En el primer capítulo se expone conceptos fundamentales de un protocolo de comunicación y de los componentes que se utilizarán para realizar el prototipo del dispositivo de medición de temperatura y control de acceso.

En el capítulo dos se describe el protocolo 1-Wire, desde cómo escribir o leer un bit por medio de dicho protocolo, hasta la descripción de los comandos ROM y comandos de funciones.

En el capítulo tres se muestra el método de detección de errores denominado Comprobación de redundancia cíclica y sus características principales. Se referencia cómo implementar el método en el bus 1-Wire por medio de software.

El capítulo cuatro hace referencia a la propiedad de red del protocolo 1-Wire, se define mediante el modelo de red OSI, se expone cada una de las capas y se explica detalladamente el proceso de direccionamiento de los diferentes dispositivos esclavos en la red.

En el capítulo cinco se presenta el diseño del prototipo de un dispositivo para la medición de temperatura y control de acceso en cuartos refrigerados.

OBJETIVOS

General

Presentar el uso del protocolo 1-Wire con comprobación de redundancia cíclica y su aplicación en la medición de temperatura y control de acceso.

Específicos

1. Exponer los conceptos básicos de un protocolo de comunicación y de los elementos que servirán para el desarrollo del prototipo de un dispositivo para medición de temperatura y control de acceso.

2. Presentar los conceptos del protocolo 1-Wire.

3. Mostrar los conceptos sobre los cuales se fundamenta el método Comprobación de redundancia cíclica.

4. Exponer los fundamentos de una LAN 1-Wire.

5. Presentar el diseño de un prototipo de dispositivo para la medición de temperatura y el control de acceso en cuartos refrigerados.

INTRODUCCIÓN

En los últimos años, la demanda de sensores, actuadores, o cualquier otro dispositivo que se refiera al control de un proceso ha crecido. En la búsqueda de un protocolo robusto, con alta inmunidad al ruido eléctrico, con un método sobre detección de errores, que utilice la mínima cantidad de conductores para la conexión entre ellos y su controlador, Dallas Semiconductor, Inc. ha creado el protocolo de comunicación 1-Wire.

1-Wire es un protocolo de comunicación bidireccional, Half-Duplex, que transporta datos y alimenta a los dispositivos que se conecten al bus por medio de una única línea o cable y tierra. Acepta solamente un maestro en el bus y uno o más esclavos. El protocolo incluye un sistema de direccionamiento. Cada dispositivo de red 1-Wire tiene un identificador de 64 bits codificado dentro del mismo, esto sirve como su dirección.

Todos los dispositivos 1-Wire son diseñados para operar en un entorno de red. Esto amplía el campo de aplicaciones a una mayor capacidad de almacenamiento, y para el almacenamiento de datos distribuidos usando solo una línea de datos común para el maestro. En relación con el Modelo de Interconexiones de Sistemas Abiertos, el protocolo de red 1-Wire cuenta con cinco capas, las cuales son: Presentación, Transporte, Red, Enlace y Física.

Utilizando todas las características antes mencionadas del protocolo 1-Wire, se propuso y realizó el diseño de un prototipo de dispositivo para la medición de temperatura y control de acceso en cuartos refrigerados, el cual utiliza todas las ventajas que brinda este protocolo de comunicación.

1. MARCO TEÓRICO

El presente capítulo expone, con el fin de crear una base teórica sobre el tema, los conceptos básicos sobre el protocolo de comunicación y canal de comunicación. Se presenta también, información acerca de los dispositivos DS18B20, DS1990, el microcontrolador PIC18F2550 y del dispositivo LCD. Asimismo, la información de los dispositivos mencionados anteriormente, porque servirá de referencia en el desarrollo del diseño del hardware y software, del prototipo de dispositivo para la medición de temperatura y control de acceso a cuartos refrigerados.

1.1. Protocolo de comunicaciones

En informática y telecomunicación, un protocolo de comunicaciones es un conjunto de reglas y normas que permiten que dos o más entidades de un sistema de comunicación se comuniquen entre ellos para transmitir información por medio de cualquier tipo de variación de una magnitud física. Se trata de las reglas o el estándar que define la sintaxis, semántica y sincronización de la comunicación, así como posibles métodos de recuperación de errores. Los protocolos pueden ser implementados por hardware, software, o una combinación de ambos.

1.1.1. Propiedades típicas

Si bien los protocolos pueden variar mucho en propósito y sofisticación, la mayoría especifica una o más de las siguientes propiedades:

- Detección de la conexión física subyacente (con cable o inalámbrica), o la existencia de otro punto final o nodo.
- *Handshaking*.
- Negociación de varias características de la conexión.
- Cómo iniciar y finalizar un mensaje.
- Procedimientos en el formateo de un mensaje.
- Qué hacer con mensajes corruptos o formateados incorrectamente (corrección de errores).
- Cómo detectar una pérdida inesperada de la conexión, y qué hacer entonces.
- Terminación de la sesión y/o conexión.
- Estrategias para mejorar la seguridad (autenticación, cifrado).
- Cómo se construye una red física.
- Cómo los computadores se conectan a la red.

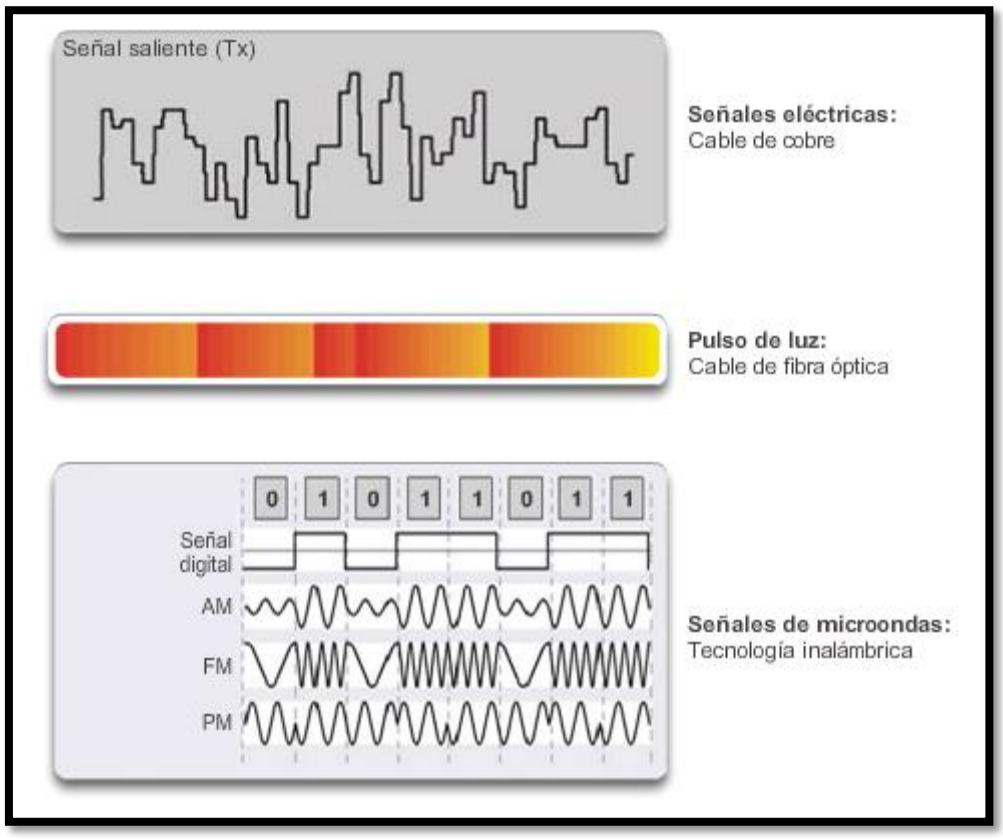
1.2. Canal de comunicación

Es el medio de transmisión por el que viajan las señales portadoras de información emisor y receptor. Es frecuente referenciarlo también como canal de datos. En el modelo OSI, el canal de comunicación se encuentra en la capa física. Existen tres formatos básicos de medios de red. La capa física produce la representación y las agrupaciones de bits para cada tipo de medio de la siguiente manera:

- Cable de cobre: las señales son patrones de pulsos eléctricos.
- Cable de fibra óptica: las señales son patrones de luz.
- Conexión inalámbrica: las señales son patrones de transmisiones de microondas. En la figura 1 se muestran ejemplos de señalización para medios inalámbricos, de cobre y de fibra óptica.

Para habilitar la interoperabilidad de la capa física, los organismos de estandarización rigen todos los aspectos de estas funciones.

Figura 1. **Representación de los datos según el medio**



Fuente: https://static-course-assets.s3.amazonaws.com/ITN50ES/module4/index.html#4.1.2.2. Consulta: 16 de agosto de 2014.

1.3. Microcontrolador

Un microcontrolador, abreviado µC, es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. Un microcontrolador incluye en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida.

Algunos microcontroladores pueden utilizar palabras de cuatro bits y funcionan a velocidad de reloj con frecuencias tan bajas como 4 kHz, con un consumo de baja potencia (mW). Por lo general, tendrá la capacidad para mantener la funcionalidad a la espera de un evento como pulsar un botón o de otra interrupción, el consumo de energía durante el estado de reposo (reloj de la CPU y los periféricos de la mayoría) puede ser solo nano vatios, lo que hace que muchos de ellos sean adecuados para aplicaciones con batería de larga duración. Otros microcontroladores pueden servir para roles de rendimiento crítico, donde sea necesario actuar más como un procesador digital de señal (DSP), con velocidades de reloj y consumo de energía más altos.

Cuando es fabricado el microcontrolador, no contiene datos en la memoria ROM. Para que pueda controlar algún proceso es necesario generar o crear y luego grabar en la EEPROM o equivalente del microcontrolador algún programa, el cual puede ser escrito en lenguaje ensamblador u otro lenguaje para microcontroladores; sin embargo, para que el programa pueda ser grabado en la memoria del microcontrolador, debe ser codificado en sistema numérico hexadecimal, que es finalmente el sistema que hace trabajar al microcontrolador cuando este es alimentado con el voltaje adecuado y asociado a dispositivos analógicos y discretos para su funcionamiento.

1.3.1. Características generales

Los microcontroladores están diseñados para reducir el costo económico y el consumo de energía de un sistema en particular. Por eso el tamaño de la unidad central de procesamiento, la cantidad de memoria y los periféricos incluidos dependerán de la aplicación. El control de un electrodoméstico sencillo, como una batidora, utilizará un procesador muy pequeño (4 u 8 bits), porque sustituirá a un autómata finito. En cambio, un reproductor de música y/o vídeo digital (MP3 o MP4) requerirá de un procesador de 32 o de 64 bits y de uno o más codecs de señal digital (audio y/o video). El control de un sistema de frenos ABS se basa, normalmente en un microcontrolador de 16 bits, al igual que el sistema de control electrónico del motor en un automóvil.

Los microcontroladores representan la inmensa mayoría de los chips de computadoras vendidos, sobre un 50 por ciento son controladores simples y el restante corresponde a DSP más especializados. Mientras se pueden tener uno o dos microprocesadores de propósito general en casa (Ud. está usando uno para esto), usted tiene distribuidos seguramente entre los electrodomésticos de su hogar una o dos docenas de microcontroladores. Pueden encontrarse en casi cualquier dispositivo electrónico como automóviles, lavadoras, hornos microondas, teléfonos, etc.

Un microcontrolador difiere de una unidad central de procesamiento normal, debido a que es más fácil convertirla en una computadora en funcionamiento, con un mínimo de circuitos integrados externos de apoyo. La idea es que el circuito integrado se coloque en el dispositivo, enganchado a la fuente de energía y de información que necesite, y eso es todo. Un microprocesador tradicional no le permitirá hacer esto, ya que espera que todas estas tareas sean manejadas por

otros chips. Hay que agregarle los módulos de entrada y salida (puertos) y la memoria para almacenamiento de información.

Un microcontrolador típico tendrá un generador de reloj integrado y una pequeña cantidad de memoria de acceso aleatorio y/o ROM/EPROM/EEPROM/flash, con lo que para hacerlo funcionar todo lo que se necesita son unos pocos programas de control y un cristal de sincronización. Los microcontroladores disponen, generalmente, también de una gran variedad de dispositivos de entrada/salida, como convertidor analógico digital, temporizadores, UARTs y buses de interfaz serie especializados, como I2C y CAN. Frecuentemente, estos dispositivos integrados pueden ser controlados por instrucciones de procesadores especializados. Los modernos microcontroladores, frecuentemente incluyen un lenguaje de programación integrado; como BASIC, que se utiliza bastante con este propósito.

Los microcontroladores negocian la velocidad y la flexibilidad para facilitar su uso. Debido a que se utiliza bastante sitio en el chip para incluir funcionalidad, como los dispositivos de entrada/salida o la memoria que incluye el microcontrolador, se prescindirá de cualquier otra circuitería.

1.3.1.1. Microcontrolador PIC18F2550

Es un producto de Microchip Technology Inc. Este microcontrolador, que es de arquitectura Harvard, es ideal para aplicaciones de baja potencia (en el orden de nano vatios), y para aplicaciones de conectividad, ya que cuenta con los siguientes puertos para la comunicaciones con otros periféricos: FS-USB (12 Mbit / s), I ² C ™ y SPI ™ (hasta 10 Mbit / s) y un puerto (EUSART).

Las grandes cantidades de memoria RAM para almacenamiento temporal y la memoria de programa flash mejorada hacen que sea ideal para el control integrado y aplicaciones de monitoreo que requieren conexión periódica, y la conexión USB de este microcontrolador lo hace ideal para carga/descarga de datos o actualización de firmware.

1.3.1.1.1. Características

- Full Speed USB 2.0 (12Mbit/s) interface
- 1K byte Dual Port RAM + 1K byte GP RAM
- Full Speed Transceiver
- 16 Endpoints (IN/OUT)
- Streaming Port
- Internal Pull Up resistors (D+/D-)
- 48 MHz performance (12 MIPS)
- Pin-to-pin compatible with PIC16C7X5

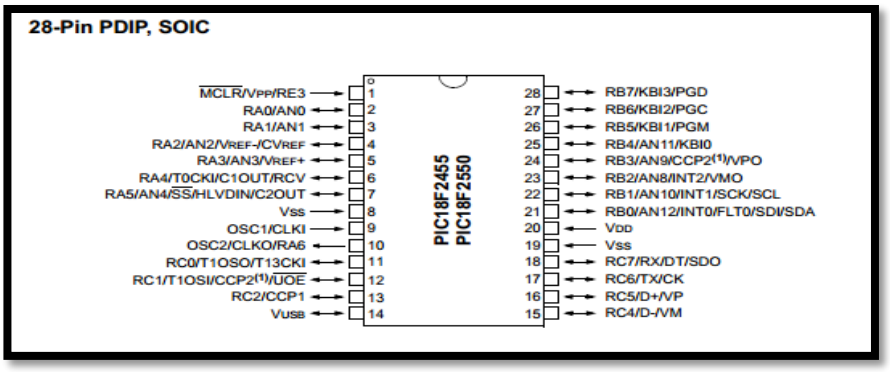
La tabla I muestra las principales características del microntrolador PIC18F2550. La figura 2 describe el diagrama de pines, el cual servirá para hacer el programa del controlador del bus 1-Wire. Asimismo, la figura 3 detalla el diagrama de bloques del microcontrolador, el cual indica los módulos internos, buses y las entradas y salidas.

Tabla I. **Características de PIC18F2550**

Parameter Name	Value
Program Memory Type	Flash
Program Memory (KB)	32
CPU Speed (MIPS)	12
RAM Bytes	2,048
Data EEPROM (bytes)	256
Digital Communication Peripherals	1-UART, 1-A/E/USART, 1-SPI, 1-I2C1-MSSP(SPI/I2C)
Capture/Compare/PWM Peripherals	1 CCP, 1 ECCP
Timers	1 x 8-bit, 3 x 16-bit
ADC	13 ch, 10-bit
Comparators	2
USB (ch, speed, compliance)	1, Full Speed, USB 2.0
Temperature Range (C)	-40 to 85
Operating Voltage Range (V)	2 to 5.5
Pin Count	40

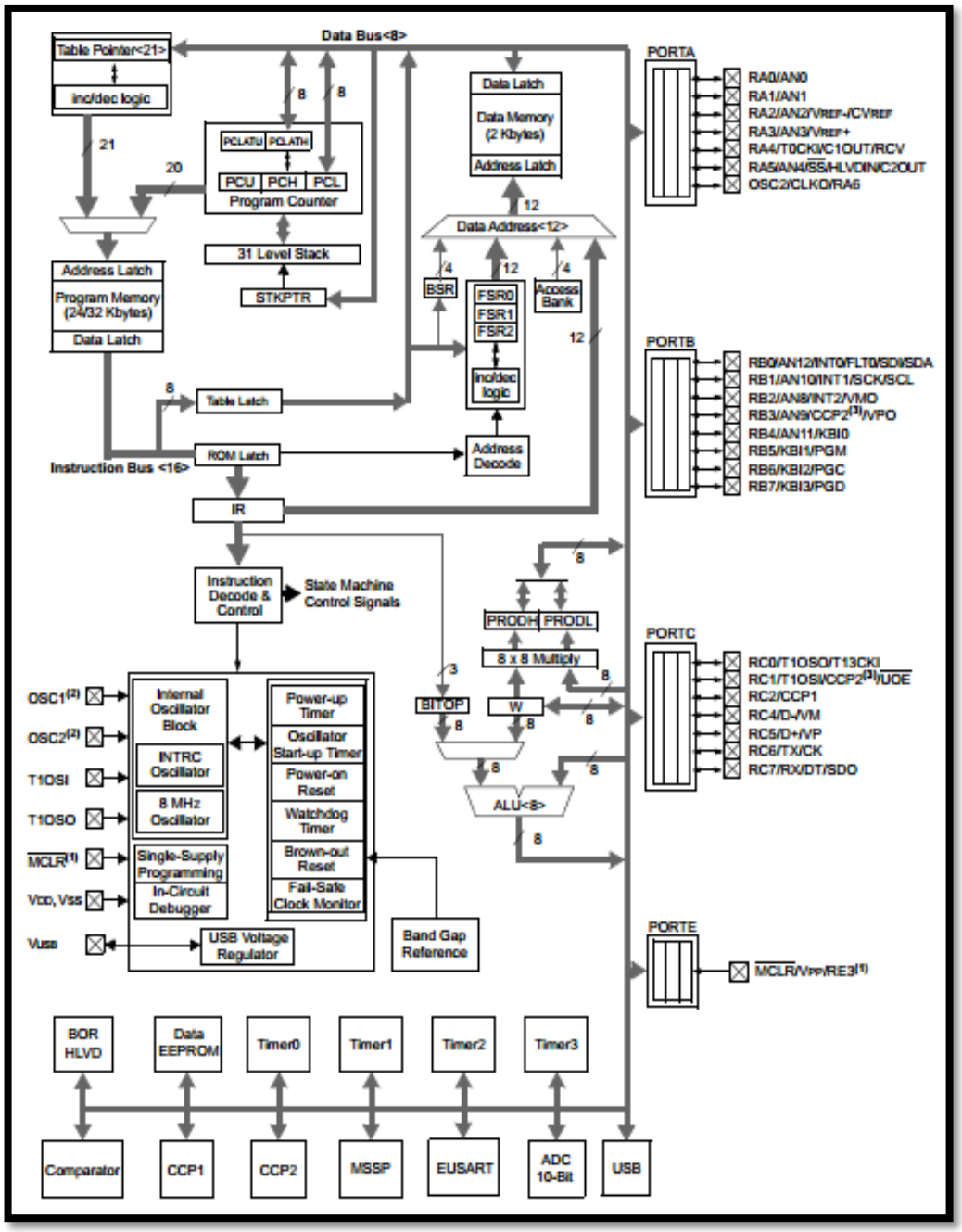
Fuente: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010300. Consulta: 16 de agosto de 2014.

Figura 2. **Diagrama de pines PIC18F2550**



Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39632e.pdf. Consulta: 16 de agosto de 2014.

Figura 3. **Diagrama de bloques PIC18F2550**



Fuente: http://ww1.microchip.com/downloads/en/DeviceDoc/39632e.pdf. Consulta: 16 de agosto de 2014.

1.4. Sensor de temperatura DS18B20

El termómetro digital DS18B20 es un dispositivo que proporciona un dato de temperatura de 9 a 12 bits, y tiene una función de alarma programable en memoria no volátil, de límite inferior y superior de temperatura. El DS18B20 se comunica a través del bus 1-Wire, que por definición requiere solo de una línea de datos y otra de tierra para la comunicación con un microcontrolador central. Tiene un rango de temperatura de funcionamiento de -55 °C a +125 °C, con una precisión de ± 0,5 °C en el rango de -10 °C a +85 °C. Además, el DS18B20 puede derivar energía directamente de la línea de datos (parasite power mode), eliminando la necesidad de una fuente de alimentación externa.

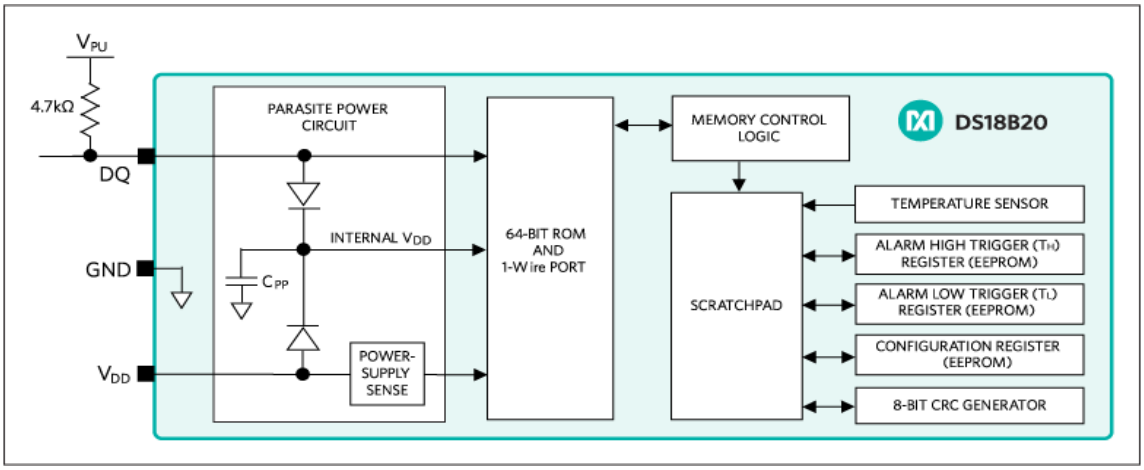
Cada uno de los DS18B20 tiene un código de serie único de 64 bits, que permite a múltiples DS18B20 funcionen en un mismo bus 1-Wire. Por lo tanto, es fácil de usar en un microprocesador para controlar muchos DS18B20 distribuidos en un área muy grande. Las aplicaciones que pueden beneficiar con esto son los controles ambientales HVAC, sistemas de control de temperatura interior en los edificios, equipos o maquinarias y la supervisión de procesos y sistemas de control.

1.4.1. Principales características

- Interfaz única 1-Wire, que requiere una única línea para su comunicación y alimentación.
- Cada dispositivo tiene un único número serial de 64 bits almacenada en una memoria ROM interna.
- Conexión multipunto, lo que simplifica el censado de temperatura en forma distribuida.
- No requiere componentes externos.

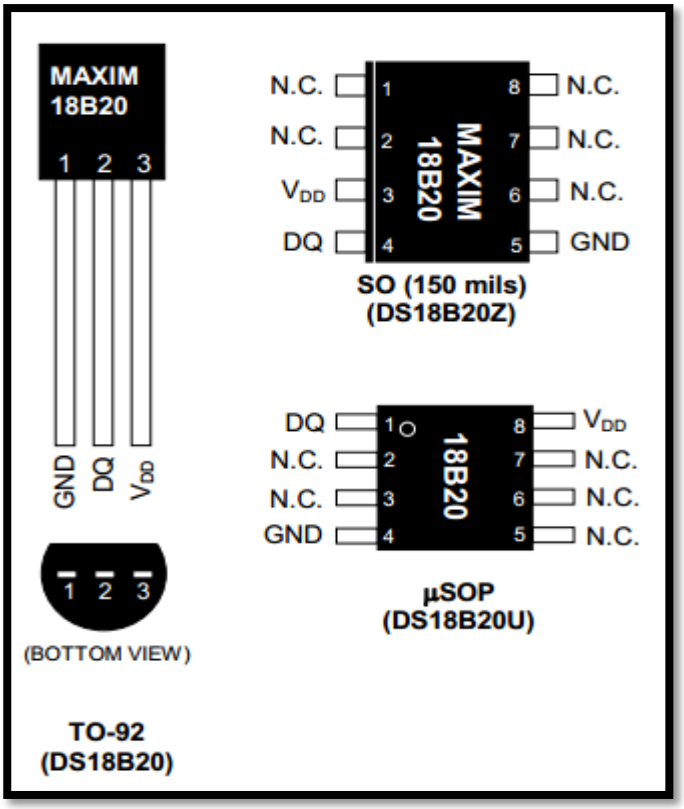
- Rango de alimentación de 3.0 V a 5,5 V
- Rango de temperatura de -55 °C a +125 °C (-67 °F a +257 °F).
- Precisión de ± 0,5 °C en el rango de -10 °C a +85 °C.
- La precisión del termómetro puede ser seleccionado por el usuario de 9 a 12 bits.
- La conversión de la temperatura con una resolución de 12 bits tarda 750 ms (tiempo máximo).
- El usuario puede definir alarmas de altos y bajos de temperatura y son grabadas en la memoria ROM.
- Disponible en empaquetado 8-Pin SO (150 milésimas), 8-Pin µSOP, y 3-Pin TO-92.

Figura 4. **Diagrama de bloques DS18B20**



Fuente: www.maximintegrated.com/en/products/analog/sensors-and-sensor-interface/DS18B20.html. Consulta: 17 de agosto de 2014.

Figura 5. **Diagrama de pines DS18B20**



Fuente: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf. Consulta: 17 de agosto de 2014.

1.4.2. Memoria

La memoria del DS18B20 está organizada como se muestra en la figura 6. Consta de una SRAM con una sección formada por una memoria EEPROM no volátil, donde se almacenan los valores límites para las alarmas de temperatura (TH y TL) y el registro de configuración. Tomar en cuenta que, si no se utilizan la función de alarma del DS18B20, TL y TH pueden ser utilizados como memoria de propósito general.

El byte 0 y el byte 1 de la memoria de trabajo contienen el LSB y MSB del registro donde se guarda el valor de la conversión de temperatura, estos bytes son solo de lectura. Los bytes 2 y 3 proveen acceso a los registros TH y TL. El byte 4 contiene los datos del registro de configuración acá es donde se define la resolución del dato de la medición de temperatura. La resolución se escoge variando los bits 6 y 5 únicamente. Esto se muestra en la tabla II.

Tabla II. **Configuración de resolución DS18B20**

BIT 6	BIT 5	Resolución (Bits)	Máximo tiempo de conversión
0	0	9	93.75 ms
0	1	10	187.5ms
1	0	11	375ms
1	1	12	750ms

Fuente: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf. Consulta: 26 de agosto de 2014.

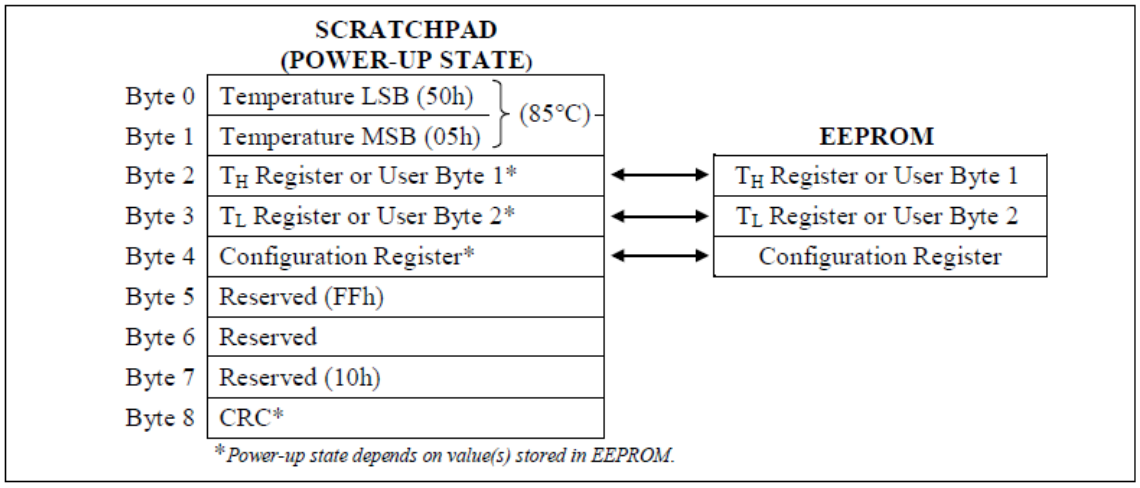
En el byte 4 solo se varían los bits 6 y 5 para escoger la resolución. Los bits 0, 1, 2, 3, 4 y 7 no deben ser sobrescritos, porque son utilizados para uso interno del DS18B20. El byte 8 de la memoria de trabajo es solo de lectura, y contiene el código CRC para los bytes del 0 al 7 de la memoria de trabajo.

Los datos son escritos en los bytes 2, 3 y 4 de la memoria de trabajo, usando el comando WRITE SCRATCHPAD. Los datos deben ser transmitidos al DS18B20 empezando por el bit menos significativo del byte 2. Para verificar la integridad de los datos, la memoria de trabajo debe ser leído, después de que los datos se escriben. Cuando se lee la memoria de trabajo, los datos son transferidos a través del bus 1-Wire comenzando con el bit menos significativo del byte 0. Para transferir TH y TL y el registro de configuración de la memoria

de trabajo a la EEPROM, el maestro debe usar el comando COPY SCRATCHPAD.

Los datos en los registros de la EEPROM se retienen, aun cuando el dispositivo esté apagado, en el encendido los datos de la EEPROM se vuelven a cargar en el área de la memoria de trabajo correspondiente. Los datos también se pueden recargar de la EEPROM a la memoria de trabajo en cualquier momento usando el comando RECALL $E^{2.}$

Figura 6. **Memoria DS18B20**



Fuente: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf. Consulta: 26 de agosto de 2014.

1.5. iButton

Es un chip de ordenador, que tiene un único número de serie de 64 bits, protegido por una gruesa lata de acero inoxidable de 16 milímetros. Debido a este contenedor único y duradero, la información puede viajar con una persona u objeto a cualquier lugar que vayan. El dispositivo iButton de acero se puede

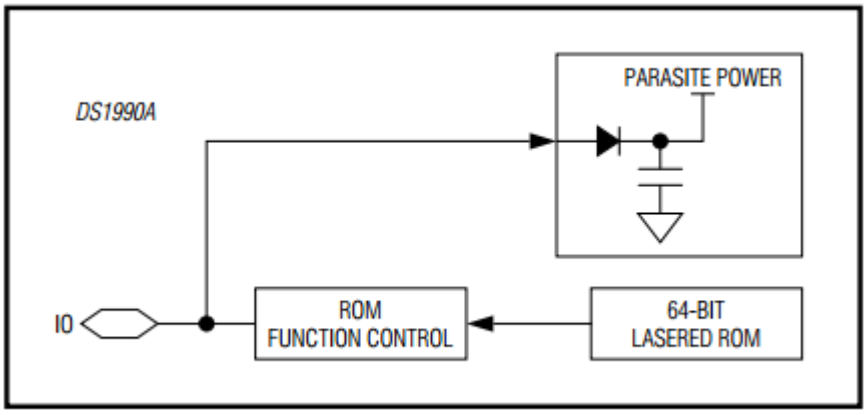
montar prácticamente en cualquier lugar, ya que es lo suficientemente resistente como para soportar entornos hostiles, tanto interiores como exteriores. Es lo suficientemente pequeño y se puede adjuntar a un llavero, anillo, reloj u otros artículos personales, y se usa a diario para aplicaciones tales como: control de acceso a edificios y equipos, administración de activos y varias tareas de registro de datos portátil.

Figura 7. **iButton**

Fuente: http://www.maximintegrated.com/en/app-notes/index.mvp/id/3808. Consulta: 17 de agosto de 2014.

El diagrama de bloques de la figura 8 muestra las mayores funciones del dispositivo, el iButton toma la energía que necesita para operar desde la línea IO como se indica en el bloque Parasite Power. La unidad de control de la función ROM incluye una interfaz 1-Wire y la lógica para implementar los comandos de función ROM, y acceder al código único de 64 bits almacenada en la ROM del dispositivo. Comparando la figura 4 con la 8, se puede observar la simpleza del iButton frente al DS18B20.

Figura 8. **Diagrama de bloques iButton**



Fuente: http://datasheets.maximintegrated.com/en/ds/.pdf. Consulta: 26 de agosto de 2014.

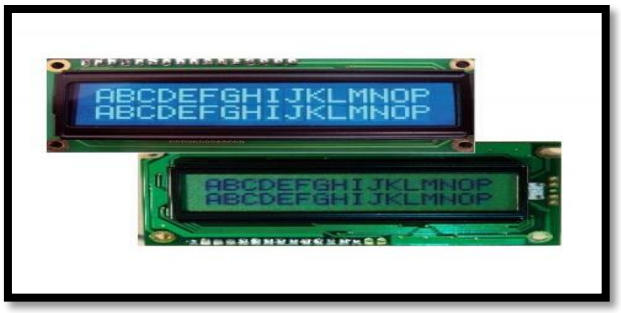
1.5.1. Componentes de un iButton

Un dispositivo iButton utiliza acero inoxidable como interfaz de comunicaciones electrónicas. Cada lata tiene un contacto de datos, llamada tapa, y un contacto a tierra, denominada base. Cada uno de estos contactos está conectado al chip de silicio en el interior. La tapa es la parte superior de la lata; la base forma los lados y la parte inferior de la lata. Los dos contactos están separados por una arandela de polipropileno. El iButton utiliza el protocolo 1-Wire para su comunicación, con solo tocar el dispositivo iButton a los dos contactos descritos anteriormente, puede comunicarse con él a través del protocolo 1-Wire. La interfaz 1-Wire tiene dos velocidades de comunicación: la velocidad estándar a 16 kbps, y la velocidad overdrive a 125 kbps.

1.6. Pantalla de cristal líquido

La definición más clara de un LCD es una pantalla de cristal líquido que visualiza ciertos caracteres. Para poder hacer funcionar un LCD, debe estar conectado a un circuito impreso en el que estén integrados los controladores del display y los pines para la conexión del display. Sobre el circuito impreso se encuentra el LCD en sí, rodeado por una estructura metálica que lo protege. En total se pueden visualizar 2 líneas de 16 caracteres cada una, es decir, 2x16=32 caracteres. A pesar de que el display solo puede visualizar 16 caracteres por línea, puede almacenar en total 40 por línea. Es el usuario el que especifica qué 16 caracteres son los que se van a visualizar. Tiene un consumo de energía de menos de 5 mA y son ideales para dispositivos que requieran una visualización pequeña o media. El LCD dispone de una matriz de 5x8 puntos para representar cada caracter. En total se pueden representar 256 caracteres diferentes. 240 caracteres están grabados dentro del LCD y representan las letras mayúsculas, minúsculas, signos de puntuación, números, entre otros.

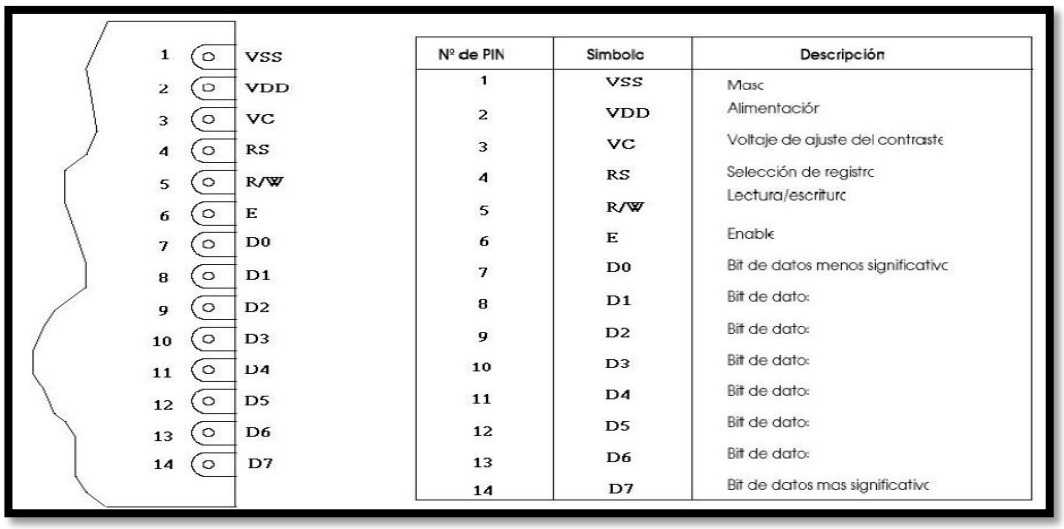
Figura 9. **Display LCD**



Fuente: http://espelectronicdesign.com/Tutoriales/Imagenes_manual_pantalla_LCD/LCD_16x2_COMBO_B.jpg. Consulta: 17 de agosto de 2014.

A continuación se observa en la figura 10 la asignación de los pines en una pantalla LCD.

Figura 10. **Asignación de pines LCD 4X20**



Nº de PIN	Simbolo	Descripción
1	VSS	Masc
2	VDD	Alimentación
3	VC	Voltaje de ajuste del contraste
4	RS	Selección de registro
5	R/W	Lectura/escritura
6	E	Enable
7	D0	Bit de datos menos significativo
8	D1	Bit de datos
9	D2	Bit de datos
10	D3	Bit de datos
11	D4	Bit de datos
12	D5	Bit de datos
13	D6	Bit de datos
14	D7	Bit de datos mas significativo

Fuente: http://server-die.alc.upv.es/asignaturas/lsed/2002-03/Pantallas_LCD/LCD.pdf. Consulta: 7 de agosto de 2014.

Los pines 1 y 2 son los utilizados para la alimentación del módulo LCD. La tensión utilizada es de 5 voltios. El pin 3 se utiliza para ajustar el contraste de la pantalla LCD. Por medio de un potenciómetro se regula la intensidad de los caracteres, a mayor tensión mayor intensidad. Se suele utilizar un potenciómetro de unos 10 kΩ o 20 kΩ, que regulará la misma tensión que se emplea para la alimentación. El pin 4 se utiliza para indicar al bus de datos si la información que le llega es una instrucción o, por el contrario, es un caracter. Si RS=0 indicara que en el bus de datos hay presente una instrucción, y si RD=1, indicará que tiene un caracter alfanumérico. El pin 5 es el de escritura o lectura. Si está a cero el módulo escribe en pantalla el dato que haya en el bus de datos, y si está a uno se leerá lo que hay en el bus de datos. El pin 6 es encargado de hacer que el

módulo LCD funcione, o por el contrario no acepte órdenes de funcionamiento. Cuando E=0 no se podrá utilizar el display y cuando E=1 se podrán transferir datos y realizar las demás operaciones. Las pines del 7 al 14 son los del bus de datos.

2. EL PROTOCOLO 1-WIRE

Este protocolo fue producido por Dallas Semiconductors, Inc., para permitir que los dispositivos de otros fabricantes se conectaran a las redes de bus serial 1-Wire. Un bus serie es un vínculo para transportar datos de un bit a la vez entre un microcontrolador (master) y una serie de dispositivos de control (esclavos). Dallas Semiconductors es ahora propiedad de Maxim, que sigue promoviendo el estándar (desde 2010).

Un bus es un cable de conexión de muchos dispositivos entre sí para la comunicación. Sin embargo, este no es el significado del nombre del producto. 1-Wire se refiere a la capacidad del bus para suministrar energía a los dispositivos, así como un enlace de comunicaciones. El sistema se aplica en la práctica con dos alambres: datos y tierra.

1-Wire es un protocolo bidireccional, Half-Duplex, que transporta datos y alimenta a los dispositivos que se conecten al bus por medio de una única línea o cable y tierra. Acepta solamente un maestro en el bus y uno o más esclavos, por lo que se dice que es multipunto. El protocolo incluye un sistema de direccionamiento. Cada dispositivo de la MicroLan debe tener un ID de 64 bits codificado en la misma. Esto sirve como su dirección. El ID es único e incluye un código de la familia de 8 bits que indica el tipo de dispositivo y la funcionalidad.

Los tipos de dispositivos atendidos por el bus 1-Wire incluyen sensores, en particular para los termómetros y estaciones meteorológicas. El sistema también se aplica en llaves electrónicas. En esta categoría, 1-Wire está integrado en una unidad llamada iButton, que se asemeja a una pila de reloj.

2.1. Señalización 1-Wire

El protocolo 1-Wire puede funcionar en dos velocidades de transmisión de datos.

- Estándar: 15.4 kbps
- Overdrive: 125 kbps

En este documento se utilizará solamente la velocidad estándar. 1-Wire funciona con señales que varían entre 0 y de 3 a 5 V DC. La comunicación se realiza con intervalos de tiempo que varía entre alto y bajo en tiempos determinados, para producir así la representación de uno y cero. Un intervalo de tiempo es el uso controlado de impulsos de baja duración de tiempo para codificar los unos y ceros binarios. El intervalo de tiempo depende de la velocidad de transmisión a la que se decida enviar la data.

- Velocidad estándar: 60 µs
- Velocidad overdrive: 8 µs

En el protocolo 1-Wire, para garantizar la integridad de los datos, se tiene varios tipos de señales. Se definen mediante este protocolo las siguientes señales: el pulso de reset, el pulso de presencia, escritura de 0, escritura de 1, lectura de 0, lectura de 1. El maestro del bus inicializa todas estas señales, con excepción del pulso de presencia.

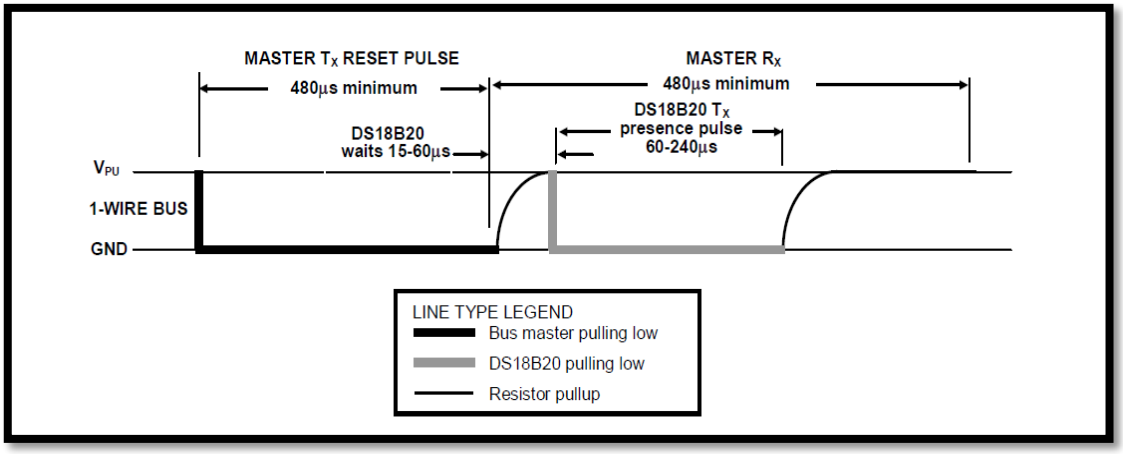
Los dispositivos esclavos 1-Wire pueden ser alimentados de dos maneras: usando el pin VDD, o en modo Parasite Power. El modo Parasite Power es muy utilizado, porque permite con un único cable alimentar y mantener comunicación

entre el maestro y el esclavo. Como se verá a continuación, algunos comandos 1-Wire responden de manera diferente dependiendo del modo de alimentación.

2.1.1. Proceso de inicialización

Toda la comunicación con el protocolo 1-Wire comienza con una secuencia de inicialización que consiste en un pulso de reset desde el maestro seguido por un pulso de presencia desde el esclavo. Esto se ilustra en la figura 11. Cuando el esclavo envía el pulso de presencia en respuesta al pulso de reset, el esclavo indica al maestro que está conectado al bus y listo para funcionar. Durante la secuencia de inicialización, el maestro del bus transmite el pulso de reset (TX), poniendo en bajo el bus 1-Wire durante un mínimo de 480 µs. El maestro de bus luego libera el bus y entra en modo de recepción (RX). Cuando se libera el bus, la resistencia de *pull-up* con valor de 4.7 kΩ pone el bus 1-Wire en alto. Cuando el esclavo detecta este flanco ascendente, espera de 15 µs a 60 µs y luego transmite un pulso de presencia poniendo el bus 1-Wire en bajo durante un tiempo de 60 µs a 240 µs.

Figura 11. **Proceso de inicialización**



Fuente: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.
Consulta: 19 de agosto de 2014.

2.1.2. Escritura/lectura

El maestro del bus escribe datos en el esclavo durante los intervalos de tiempo de escritura, y lee datos del esclavo durante los intervalos de tiempo de lectura. Un único bit de datos es transmitido a través del bus durante un intervalo de tiempo.

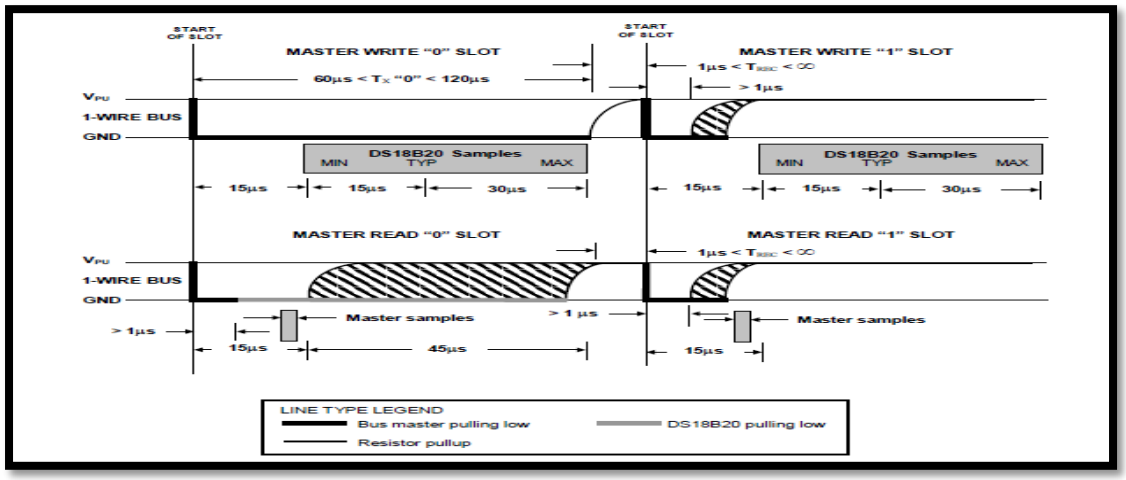
2.1.2.1. Intervalo tiempo de escritura

Existen dos tipos de intervalo de tiempo de escritura: el intervalo de tiempo de escritura de 1 y el intervalo de tiempo de escritura de 0. El maestro del bus utiliza el intervalo de tiempo de escritura de 1 para escribir un 1 en dispositivo esclavo, y utiliza el intervalo de tiempo de escritura de 0, para escribir un 0 en el dispositivo esclavo. Todos los intervalos de tiempo de escritura deben tener un mínimo de 60 µs de duración con un mínimo de un tiempo de recuperación 1 µs

entre los intervalos de escritura individuales. Ambos tipos de intervalos de tiempo de escritura son iniciados por el maestro poniendo el bus 1-Wire en bajo (ver figura 12).

Para generar un intervalo de escritura de 1, se debe poner el bus en bajo, el maestro debe liberar el bus después de 15 us. Cuando se libera el bus, la resistencia de *pull-up* de 4.7 kΩ pondrá en alto el bus. Para generar un intervalo de escritura de 0, después de poner el bus en bajo, el maestro del bus, debe continuar manteniendo en bajo el bus durante in tiempo mínimo de 60 µs. El dispositivo esclavo muestrea el bus 1-Wire durante una ventana que tiene una duración de 15 µs a 60 µs, después de que el maestro inicializa el intervalo de tiempo de escritura. Si el bus está en alto durante la ventana de muestreo, el dispositivo esclavo lee un 1, y si el bus se encuentra en bajo, un cero será leído por el dispositivo esclavo.

Figura 12. **Intervalos de escritura/lectura**



Fuente: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.

Consulta: 21 de agosto de 2014.

2.1.2.2. Intervalo de tiempo lectura

Los dispositivos esclavos en el protocolo 1-Wire solo pueden transmitir datos al maestro del bus cuando este empieza los intervalos de lectura. Por lo tanto, el maestro debe generar espacios de tiempo inmediatamente después de emitir el comando READ SCRATCHPAD [BEh], READ POWER SUPPLY [B4h], de modo que el esclavo puede proporcionar los datos solicitados. Además, el maestro puede generar intervalos de tiempo de lectura después de enviar el comando CONVERT T [44h] o RECALL E2 [B8h], para preguntar el estado de la operación. Todos los intervalos de tiempo de lectura deben tener un mínimo de 60 µs con un mínimo de 1 µs de tiempo de recuperación entre los intervalos de tiempo.

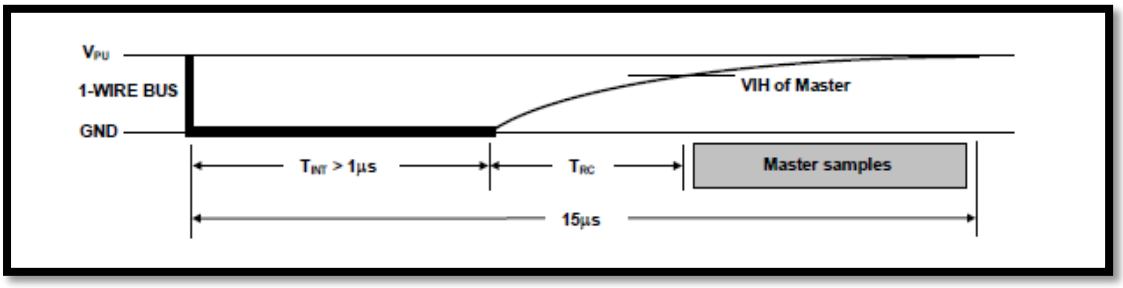
Un intervalo de tiempo de lectura es iniciada por el dispositivo maestro poniendo el bus 1-Wire en bajo por un mínimo de 1 µs y luego liberando el bus (ver figura 12). Después de que el maestro inicia el intervalo de tiempo de lectura, el esclavo comenzará a transmitir un 1 o 0 en el bus. El esclavo del bus transmite un 1 dejando el bus en alto y transmite un 0 poniendo en bajo el bus. Cuando se transmite un 0, el esclavo debe liberar el bus al final del intervalo de tiempo y el bus debe ser puesto en alto por la resistencia de *pull-up*.

La salida de datos del esclavo 1-Wire es válido para 15 µs después del flanco descendente que inicio el intervalo de tiempo de lectura. Por lo tanto, el maestro debe liberar el bus y luego probar el estado de bus dentro de 15 µs desde el inicio de la intervalo de lectura.

La figura 13 muestra que la suma de TINIT, TRC y TSAMPLE debe ser inferior a 15 µs para un intervalo de tiempo de lectura. La figura 14 muestra el margen de temporización del sistema se maximiza manteniendo TINIT y TRC tan

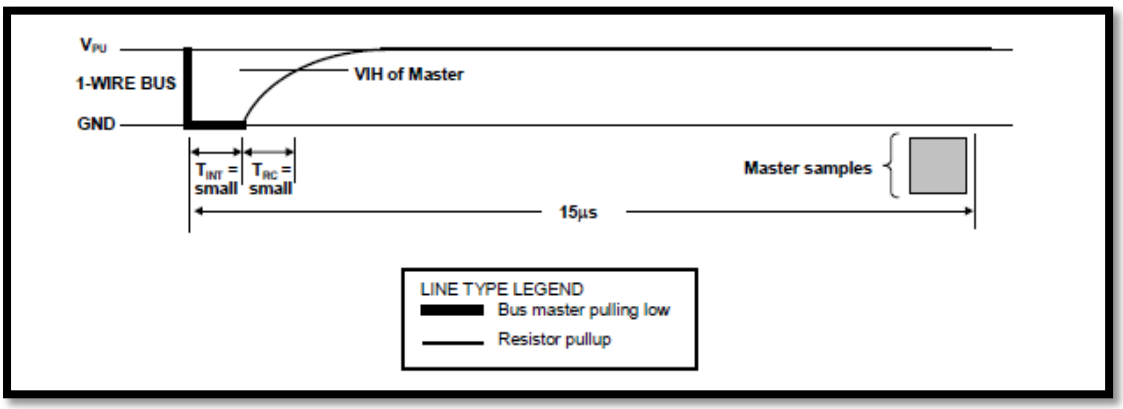
corto como sea posible y localizando el tiempo de muestra durante los intervalos de tiempo de lectura hacia el final del periodo de 15 µs.

Figura 13. **Detalle de tiempo de lectura del maestro**



Fuente: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.
Consulta: 21 de agosto de 2014.

Figura 14. **Recomendación para 1 tiempo de lectura del maestro**

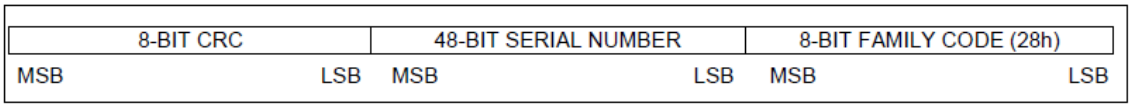


Fuente: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.
Consulta: 21 de agosto de 2014.

2.2. Código ROM

Cada dispositivo esclavo 1-Wire contiene un código único de 64 bits guardado en la memoria ROM. Los 8 bits menos significativos del código ROM indican a que familia pertenece el esclavo (sensor, llave, actuador, etc.). Los siguientes 48 bits contienen un número único de serie. Los 8 bits más significativos contienen una comprobación de redundancia cíclica que es calculada de los primeros 56 bits del código ROM.

Figura 15. **Código ROM**



Fuente: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.

Consulta: 26 de agosto de 2014.

2.3. Secuencia de transmisión

La secuencia de transmisión de datos entre el maestro y el esclavo en el protocolo 1-Wire requiere de tres pasos.

- Inicialización
- Comandos ROM
- Comando de funciones

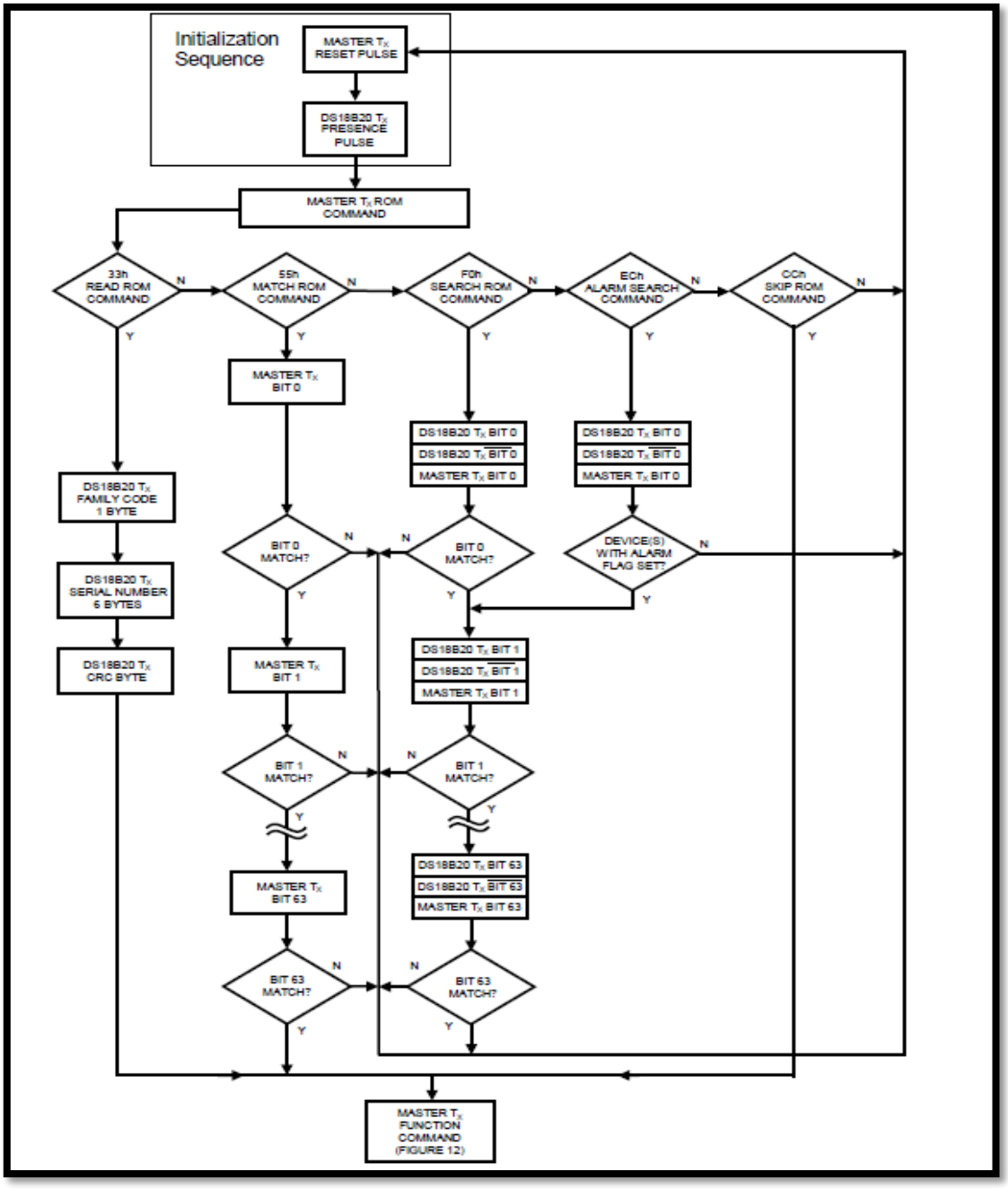
2.3.1. Inicialización

Todas las transmisiones que se realicen a través del bus 1-Wire comienzan con una secuencia de inicialización, la cual consiste en un pulso de reset transmitido por el maestro del bus seguido por pulsos de presencia transmitido por los esclavos en el bus. El pulso de presencia permite al maestro del bus saber que los dispositivos esclavos, como el DS18B20 están listos para la operar. El detalle de los pulsos de reset y presencia está detallada en la figura 11.

2.3.2. Comandos ROM

Después de que el maestro del bus ha detectado un pulso de presencia, puede emitir un comando ROM. Estos operan con los códigos únicos de 64 bits de longitud que cada uno de los esclavos tiene guardado en la ROM, y permite al maestro reconocer y singularizar un dispositivo en específico de los muchos que pudiesen estar conectados al bus en ese momento. Estos comandos también permiten que el maestro determine cuántos y qué tipos de dispositivos están presentes en el bus, y si algún dispositivo ha presentado alguna situación de alarma. Para el DS18B20 y DS1990A existen cinco comandos ROM, y cada uno es de 8 bits de longitud. El maestro debe emitir un comando ROM adecuado antes de emitir un comando de función, la figura 16 muestra un diagrama de flujo del funcionamiento de los comandos ROM en los dispositivos producidos por la empresa Maxim Integrated.

Figura 16. **Diagrama de flujo comandos ROM**



Fuente: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.

Consulta: 22 de agosto de 2014.

2.3.2.1. SEARCH ROM [F0h]

Cuando el sistema inicia por primera vez y empieza a funcionar, el protocolo 1-Wire indica que el maestro debe pedir que los esclavos conectados al bus en ese momento se identifiquen. Para ello debe de solicitar el código ROM de cada uno de los dispositivos conectados al bus, esto ayuda al maestro a identificar los dispositivos, ver qué clase de dispositivo es y cuántos están conectados al bus en ese instante.

El maestro aprende los códigos por medio de un proceso de deshabilitación de dispositivos esclavos, para llevar a cabo un ciclo de SEARCH ROM, tantas veces como sea necesario como para identificar a todos los dispositivos esclavos. Si solo hay un esclavo en el bus el comando READ ROM se puede utilizar en lugar del comando SEARCH ROM. Después de cada ciclo SEARCH ROM, el maestro del bus debe de volver al paso 1, inicialización.

2.3.2.2. READ ROM [33h]

Este comando solo se puede utilizar cuando hay un esclavo en el bus, permite al maestro del bus leer el código único de 64 bits almacenado en la ROM, sin necesidad de utilizar el procedimiento SEARCH ROM. Si se utiliza este comando cuando hay más de un esclavo presente en el bus, una colisión de datos se producirá cuando todos los esclavos intenten responder al mismo tiempo.

2.3.2.3. MARCH ROM [55h]

El comando MARCH ROM, seguido por una secuencia de código ROM de 64 bits, permite que el maestro del bus se comunique con un dispositivo esclavo en específico, en un bus multipunto o singular. Solo el esclavo que coincide exactamente con la secuencia de código ROM de 64 bits responderá al comando de función, emitido por el maestro, todos los demás esclavos del bus esperará un pulso de reset.

2.3.2.4. SKIP ROM [CCh]

El maestro puede usar este comando para direccionar todos los dispositivos conectados al bus simultáneamente, sin necesidad de enviar ningún tipo información de código ROM. Por ejemplo, el maestro puede hacer que si la red están conectados solo sensores de temperatura (DS18B20), puede ordenar que todos inicien la conversión de temperatura simultáneamente mediante la emisión del comando SKIP ROM seguido de un comando CONVERT T [44h]. En este caso se ahorra tiempo al permitir que le maestro pueda enviar la orden a varios sensores conectados sin necesidad de direccionarlos.

Tomar en cuenta que el comando READ SCRATCHPAD [BEh], se puede ejecutar después de SKIP ROM, solo si hay un único dispositivo en la red, de lo contrario se deberá usar MACH ROM seguido de READ SCRATCHPAD para direccionar a qué sensor se está pidiendo la lectura de temperatura. Un comando SKIP ROM seguido de un READ SCRATCHPAD provocará una colisión de datos en el bus si hay más de un esclavo, porque múltiples dispositivos intentarán transmitir al mismo tiempo.

2.3.2.5. ALARM SEARCH [ECh]

Este comando opera de la misma manera que SEARCH ROM excepto que solo esclavos con bandera en alto en señal de alarma responderán. Este comando permite, que dispositivos como el DS18B20, den señales de alerta de temperatura máxima y mínima en conversiones recientes de temperatura. Después de cada comando ALARM SEARCH, seguido por el intercambio de datos, el maestro del bus debe volver al paso de inicialización.

2.3.3. Comando de funciones

Estos son específicos para cada uno de los modelos de dispositivos esclavos, en esta sección se hará énfasis a los comandos de funciones del sensor de temperatura DS18B20.

Después de que el maestro del bus ha utilizado un comando ROM para abordar un dispositivo esclavo, en este caso específico un DS18B20, con el que desea comunicarse, el maestro puede emitir un comando de función DS18B20. Estos permiten al maestro escribir y leer desde la memoria EEPROM del DS18B20, iniciar conversiones de temperatura y determinar el modo de fuente de alimentación. Los comandos de función DS18B20 que se describen a continuación se resumen en la tabla III y se ilustran en el diagrama de flujo en la figura 17.

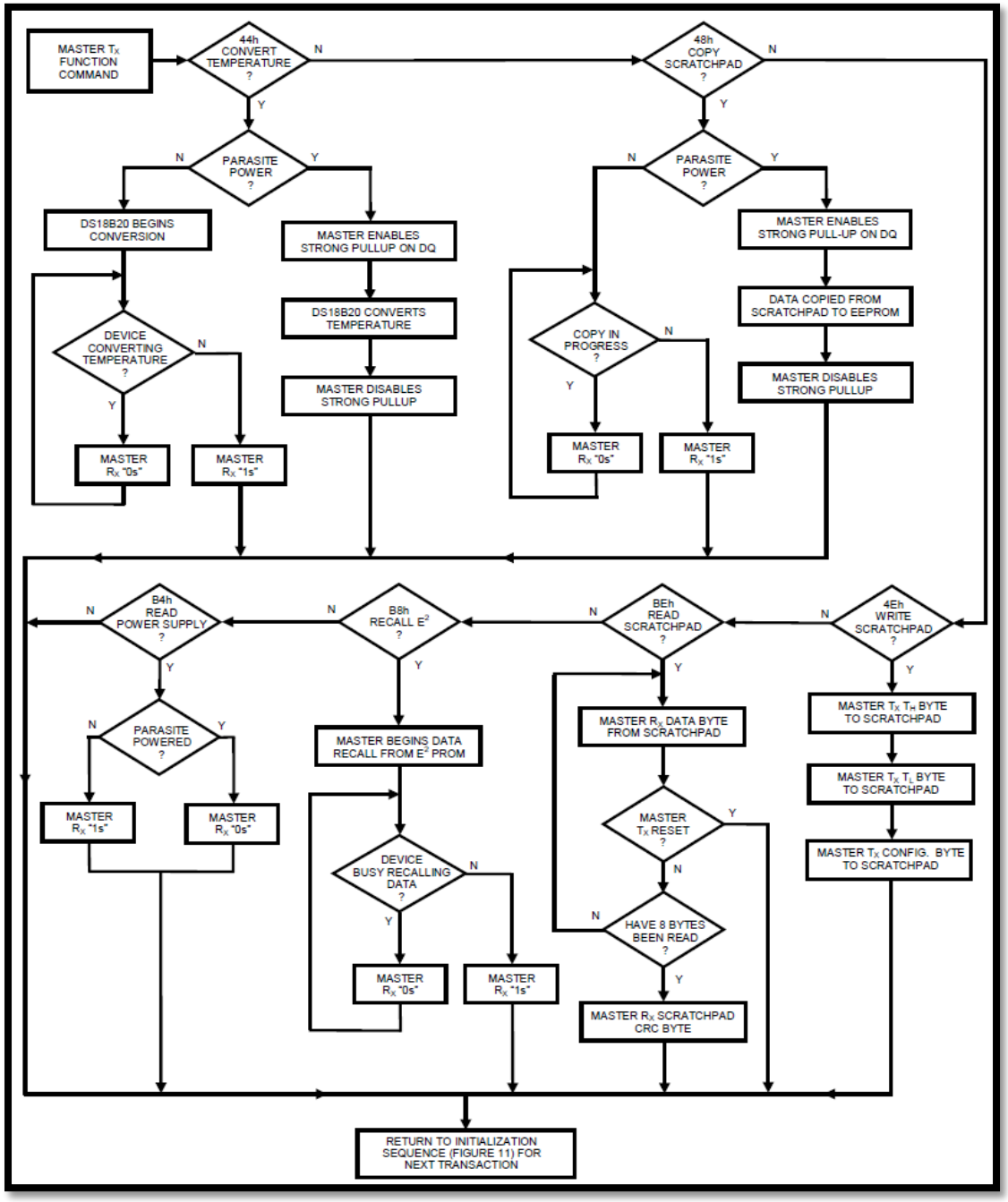
Tabla III. **Conjunto de comandos de función DS18B20**

COMMAND	DESCRIPTION	PROTOCOL	1-Wire BUS ACTIVITYAFTER COMMAND IS ISSUED	NOTES
TEMPERATURE CONVERSION COMMANDS				
Convert T	Initiates temperature conversion.	44h	DS18B20 transmits conversion status to master (not applicable for parasite-powered DS18B20s).	1
MEMORY COMMANDS				
Read Scratchpad	Reads the entire scratchpad including the CRC byte.	BEh	DS18B20 transmits up to 9 data bytes to master.	2
Write Scratchpad	Writes data into scratchpad bytes 2, 3, and 4 (T_H, T_L, and configuration registers).	4Eh	Master transmits 3 data bytes to DS18B20.	3
Copy Scratchpad	Copies T_H, T_L, and configuration register data from the scratchpad to EEPROM.	48h	None	1
Recall E^2	Recalls T_H, T_L, and configuration register data from EEPROM to the scratchpad.	B8h	DS18B20 transmits recall status to master.	
Read Power Supply	Signals DS18B20 power supply mode to the master.	B4h	DS18B20 transmits supply status to master.	

Fuente: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.

Consulta: 23 de agosto de 2014.

Figura 17. **Diagrama de flujo comando de funciones**



Fuente: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.

Consulta: 23 de agosto de 2014.

2.3.3.1. CONVERT T [44h]

Este comando inicia una sola conversión de temperatura. Tras la conversión, el valor de la temperatura se almacena en el registro de temperatura de 2 bytes y el DS18B20 vuelve a su estado de reposo de baja potencia. Si el dispositivo se utiliza en el modo de alimentación parasita (parasite power mode), dentro de 10 µs (como máximo), después de que este comando es usado el maestro debe habilitar un *strong pull-up* en el bus 1-Wire durante el tiempo en que se realice la conversión de temperatura dentro del DS118B20. Si el DS18B20 es alimentado por una fuente externa, el maestro puede emitir los intervalos de tiempo de lectura después de usar el comando CONVER T, y el DS18B20 responderá transmitiendo un 0 mientras que la transmisión de la temperatura esté en curso y un 1 cuando se realiza la conversión. En el modo Parasite Power, esta técnica de notificación no se puede utilizar, ya que el bus es puesto en alto por un *strong pull-up* durante la conversión.

2.3.3.2. WRITE SCRATCHPAD [4Eh]

Este comando permite al maestro escribir 3 bytes de datos en la memoria de trabajo del DS18B20. El primero se escribe en el registro TH (byte 2 de la memoria de trabajo), el segundo se escribe en el registro TL (byte 3), y el tercero se escribe en el registro de configuración (byte 4).

Los datos deben ser transmitidos bit a bit, comenzando con el menos significativo. Los tres bytes deben ser escritos antes de que el maestro transmita un reset en el bus, de lo contrario los datos podrían corromperse.

2.3.3.3. READ SCRATCHPAD [BEh]

Este comando permite que el maestro del bus lea el contenido de la memoria de trabajo. La transferencia de datos comienza con el bit menos significativo, el byte 0 y continúa a través de la memoria de trabajo hasta que el byte 9 sea leído. El maestro debe emitir un reinicio para terminar la lectura en cualquier momento si solo se requiere leer un segmento de la memoria de trabajo.

2.3.3.4. COPY SCRATCHPAD [48h]

Este comando copia el contenido de la memoria de trabajo TH, TL y registro de configuración, bytes 2, 3 y 4 a la EEPROM del DS18B20. Si el dispositivo se está utilizando en modo Parasite Power, 10 µs después de emitir el comando, el maestro debe permitir un *strong pull-up* en el bus 1-Wire por lo menos 10 ms.

2.3.3.5. RECALL E^2 [B8h]

Este comando llama los valores de disparo de la alarma de temperatura (TH y TL), y los datos de configuración guardados en la EEPROM y coloca los datos en los bytes 2,3 y 4 en la memoria de trabajo.

El dispositivo maestro puede emitir los intervalos de tiempo de lectura seguido del comando RECALL E^2 y el DS18B20 indicará el estado de la memoria mediante la transmisión de 0, mientras que la recuperación de los datos esté en curso y 1 si el proceso ha sido un éxito. La operación de recuperación se realiza automáticamente en el arranque, por lo que los datos válidos están disponible en la memoria de trabajo tan pronto como se aplica potencia al dispositivo.

2.3.3.6. READ POWER SUPLY [B4h]

El dispositivo maestro emite este comando seguido de intervalo de tiempo de lectura para determinar si algún DS18B20 conectado al bus usa el modo Parasite Power. Durante el intervalo de tiempo de lectura, el DS18B20 que se encuentre en modo Parasite Power, deberá poner el bus en bajo, y los DS18B20 externamente alimentados permitirán al bus mantenerse en alto.

3. COMPROBACIÓN DE REDUNDANCIA CÍCLICA

Una serie de datos pueden ser comprobados para determinar si existen errores de diferentes maneras. Una de las formas es incluir un bit adicional de comprobación en cada paquete para indicar si un error ha ocurrido.

Para paquetes de caracteres de 8-bits ASCII, por ejemplo, un bit adicional se adjunta a cada caracter ASCII indicando si este contiene errores. Supóngase que el dato consiste en una cadena de bits, como esta, 11010001. Un noveno bit deberá agregarse a la trama, de manera que el número total de bits que son 1 sea siempre un número impar. Por lo tanto, un 1 se añadirá y el paquete de datos se convertirá en 111010001.

El caracter subrayado indica el valor del bit de paridad requerido para hacer el paquete completo de 9 bits tenga un número impar de bits. Si el dato recibido es 111010001, entonces se puede asumir que la información es correcta., sin embargo, si el dato recibido fuera 111010101, donde el séptimo bit de la izquierda ha sido recibido de forma incorrecta, el número total de 1 ya no es impar y una condición de error ha sido detectada, y se tomarían acciones oportunas.

Este tipo de esquemas es llamado paridad impar. Similarmente el número total de 1’s, también podrían ser elegidos para ser siempre igual a un número par, en consecuencia, el esquema ahora adoptaría el termino paridad par. Estos esquemas son limitados a detectar un número impar o par de errores de bit, sin embargo, en el ejemplo anterior, si el dato fue corrompido y el paquete se convierte en 111011101 donde el bit 6 y 7 contando desde la izquierda son erróneos, la comprobación de paridad impar aparece correcta, sin embargo, el

error puede pasar desapercibo si se utiliza paridad par o impar. Este tipo de comprobación de errores solamente funciona cuando hay solamente hay un error en la trama de datos.

En la transmisión de información una de las más importantes investigaciones que se realiza es la de tratar de averiguar la forma en que se puedan detectar errores en los datos transmitidos para así garantizar la fidelidad de los datos recibidos. En este documento se utilizará uno de los métodos más usados por la facilidad y fidelidad en sus resultados, se trata de la comprobación de redundancia cíclica, pero enfocado a los dispositivos creados por Dallas Semiconductor que utilicen el protocolo 1-wire para su comunicación.

3.1. Dallas Semiconductor 1-Wire CRC

El esquema de detección más eficiente para localizar errores en un flujo de datos en serie, con la mínima cantidad de hardware es la comprobación de redundancia cíclica (CRC). Las operaciones y propiedades de una función CRC usadas en los productos de Dallas Semiconductor, se presentarán a continuación, sin entrar en los detalles matemáticos en lo que respecta a probar declaraciones y descripciones.

La CRC puede entenderse más fácilmente teniendo en cuenta la función, ya que en realidad sería construido en hardware, generalmente representado como un arreglo de registro de desplazamiento con retroalimentación como el mostrado en la figura 18. Alternativamente el CRC se muestra a veces como una expresión polinómica en función de X, con coeficientes binarios para cada uno de los términos. Los coeficientes corresponden directamente a los caminos de realimentación mostrados en la implementación del registro de desplazamiento.

El número de estados en el registro de desplazamiento para la descripción de hardware o el orden más alto del coeficiente presente en la expresión polinómica, indica la magnitud del valor de CRC que se calcula. Los códigos CRC que se utilizan habitualmente en las comunicaciones de datos digitales incluyen la CRC-16 y la CRC-CCITT, cada uno de los cuales calcula un valor de CRC de 16 bits.

La magnitud del Dallas semiconductor 1-Wire CRC (DOW CRC) es de ocho bits, que se utiliza para comprobar el código ROM de 64 bits escrito en cada producto 1-Wire y la información de la memoria de trabajo de los dispositivos esclavos. Este código ROM consiste en un código de familia de 8 bits escrito en el byte menos significativo, un número de serie único de 48 bits escritos en los próximos seis bytes y un valor CRC que se calcula basándose en los 56 bits anteriores de ROM y que luego es escrito en el byte más significativo.

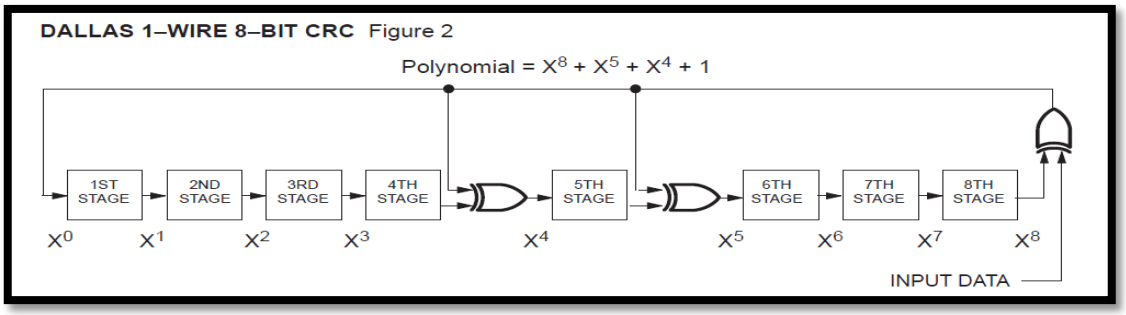
La ubicación de los caminos de realimentación representados por la compuertas OR exclusivas, o la presencia de los coeficientes de la expresión polinómica determinar las propiedades de la CRC y la capacidad del algoritmo para localizar ciertos tipos de errores en los datos. Los tipos de errores que puede detectar el DOW CRC son los siguientes.

- Cualquier número impar de errores en cualquier lugar dentro de la trama de datos.
- Todos los errores de doble bit en cualquier lugar dentro de la trama de datos.
- Cualquier grupo de errores que puedan estar contenidos en una ventana de 8 bits. (1-8 bits incorrectos).
- La mayoría de los grandes grupos de errores.

3.1.1. Implementación de DOW CRC

La entrada de datos es una OR exclusiva con la salida de la octava etapa del registro de desplazamiento, el cual es considerado matemáticamente como un circuito de división. La entrada de datos es el dividendo y el registro de desplazamiento con retroalimentación actúa como el divisor. El cociente resultante se descarta, y residuo es el valor de la CRC para ese flujo particular de datos de entrada, que reside en el registro de desplazamiento después de que el último bit de datos ha sido desplazado dentro del registro. Desde la aplicación de registro de desplazamiento es obvio que el resultado final (valor CRC) es dependiente, de una manera muy compleja, en la historia pasada de los bits presentados. Por lo tanto se necesitaría una combinación extremadamente rara de errores para que este método no detectara que el dato está corrupto.

Figura 18. **Dallas 1-Wire 8-bit CRC**



Fuente: web.archive.org/web/20090127003835/http://www.maxim-ic.com/products/ibutton/ibuttons/standard.pdf. Consulta: 30 de agosto de 2014.

El ejemplo en la figura 18 calcula el valor de CRC, después de cada bit de datos es presentado. Se calculará el valor del CRC de un código ROM de 64 bits. Supóngase tener el código 0200000001B81Ch y se intenta encontrar su CRC. El registro de desplazamiento es siempre reseteado, quedando con valor en 0, al

inicio de cada cálculo. El cálculo comienza con el bit menos significativo de la ROM de 64 bits, que es el 02h, código de familia a la que pertenece, en este ejemplo. Después de que todos los 56 bits de datos sean ingresados al registro (número de serie + código de familia), el valor que contiene el registro de desplazamiento es A2h, que es el valor del DOW CRC para esta entrada de datos.

Si el valor de CRC que se ha calculado (A2h en este ejemplo), ahora es usado como entrada para el registro de desplazamiento para los siguientes 8 bits de datos, el resultado final en el registro de desplazamiento después de ingresar los 64 bits de datos deberá ser 00h. Esta propiedad es siempre verdadera para el algoritmo DOW CRC.

Si cualquier valor de 8 bits que aparece en el registro de desplazamiento se utiliza también como los siguientes ocho bits en el flujo de entrada, entonces el resultado que aparece en el registro de desplazamiento después del octavo bit de datos será siempre 00h.

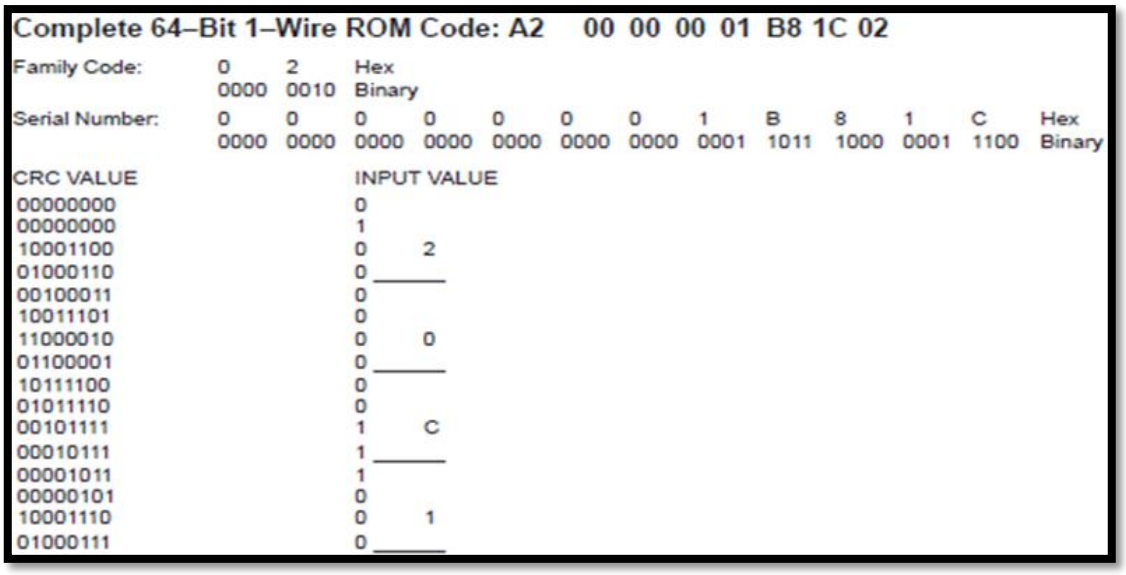
Esto puede explicarse por la observación de que el contenido de la octava etapa del registro de desplazamiento es siempre igual al bit de datos de entrada, haciendo que la salida de la puerta XOR que controla la primera etapa del registro de desplazamiento siempre igual a un 0 lógico. Esto hace que el registro de desplazamiento simplemente cambie a 0 de izquierda conforme cada bit de dato es ingresado, hasta que todo el registro se llena con 0 después del octavo bit. La estructura de 1-Wire64-bit ROM de Dallas semiconductor, usa esta propiedad para simplificar el diseño del hardware de un dispositivo usado para leer la ROM.

El registro de desplazamiento en el host se borra y luego los 64 bit ROM son leídos, incluyendo el valor CRC. Si la lectura ha sido correcta, el registro de

desplazamiento es de nuevo 0, que es una condición fácil de detectar. Si un valor distinto de cero se mantiene en el registro de desplazamiento, la operación de lectura debe repetirse.

La figura 19 muestra el comportamiento del registro de corrimiento. Se observa que la información debe ingresar comenzando por el bit menos significativo.

Figura 19. **Ejemplo para cálculo para DOW CRC**

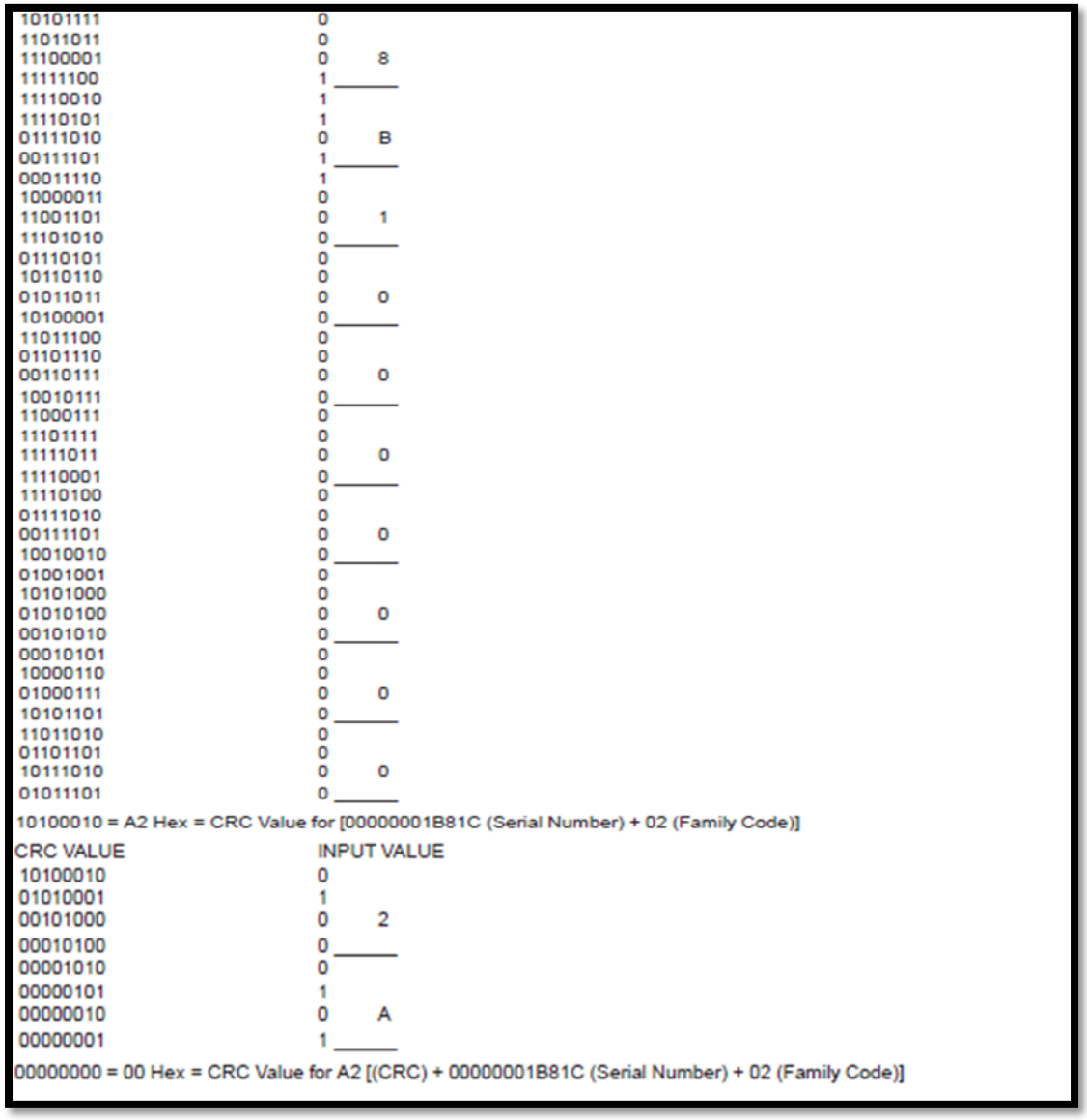


Complete 64–Bit 1–Wire ROM Code: A2 00 00 00 01 B8 1C 02

Family Code:	0	2	Hex										
	0000	0010	Binary										
Serial Number:	0	0	0	0	0	0	0	1	B	8	1	C	Hex
	0000	0000	0000	0000	0000	0000	0000	0001	1011	1000	0001	1100	Binary

CRC VALUE	INPUT VALUE	
00000000	0	
00000000	1	
10001100	0	2
01000110	0	
00100011	0	
10011101	0	
11000010	0	0
01100001	0	
10111100	0	
01011110	0	
00101111	1	C
00010111	1	
00001011	1	
00000101	0	
10001110	0	1
01000111	0	

Continuación de la figura 19.



```
10101111          0
11011011          0
11100001          0      8
11111100          1 ______
11110010          1
11110101          1
01111010          0      B
00111101          1 ______
00011110          1
10000011          0
11001101          0      1
11101010          0 ______
01110101          0
10110110          0
01011011          0      0
10100001          0 ______
11011100          0
01101110          0
00110111          0      0
10010111          0 ______
11000111          0
11101111          0
11111011          0      0
11110001          0 ______
11110100          0
01111010          0
00111101          0      0
10010010          0 ______
01001001          0
10101000          0
01010100          0      0
00101010          0 ______
00010101          0
10000110          0
01000111          0      0
10101101          0 ______
11011010          0
01101101          0
10111010          0      0
01011101          0 ______
10100010 = A2 Hex = CRC Value for [00000001B81C (Serial Number) + 02 (Family Code)]
CRC VALUE         INPUT VALUE
10100010          0
01010001          1
00101000          0      2
00010100          0 ______
00001010          0
00000101          1
00000010          0      A
00000001          1 ______
00000000 = 00 Hex = CRC Value for A2 [(CRC) + 00000001B81C (Serial Number) + 02 (Family Code)]
```

Fuente: Fuente: web.archive.org/web/20090127003835/http://www.maxim-ic.com/products/ibutton/ibuttons/standard.pdf. Consulta: 31 de agosto de 2014.

En el proyecto al cual se dedica este documento, se utilizará el DOW CRC para comprobar los datos del ROM de 8 bytes donde se almacena el código único de cada dispositivo y para corroborar los datos de la memoria de trabajo del DS18B20, que es de 9 bytes.

4. LAN 1-WIRE

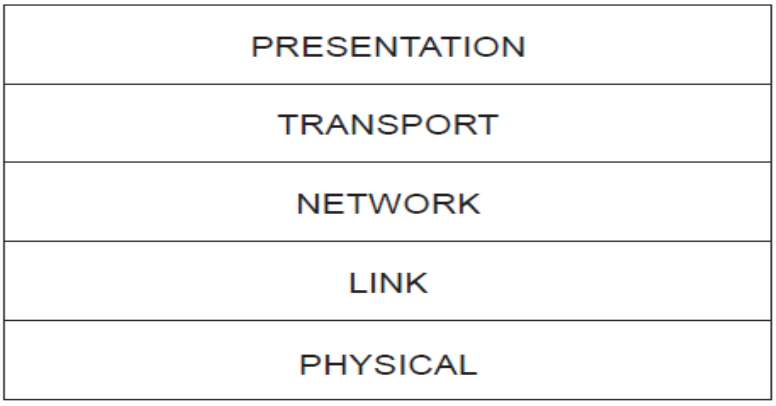
Todos los dispositivos 1-Wire son diseñados para operar en un entorno de red. Esto amplía el campo de aplicaciones a una mayor capacidad de almacenamiento, y para el almacenamiento de datos distribuidos usando solo una línea de datos común para el maestro. Las redes siempre requieren números de identificación de todos los nodos de la red. Los dispositivos esclavos tienen un código único guardado en la ROM, que es adecuado como identificador de nodo, por lo que el usuario no tiene que preocuparse por conflictos con los identificadores de los nodos. La interfaz Open Drain del bus 1-Wire, evita problemas potenciales si se producen conflictos en el bus, de hecho, la interfaz de datos 1-Wire, es en realidad una LAN 1-Wire (MicroLAN™), con todas las características necesarias para el funcionamiento de un bus multipunto con un solo maestro.

4.1. Modelo de la arquitectura del protocolo de red 1-Wire

El software o firmware que gestiona la transferencia de datos hacia y desde los dispositivos 1-Wire está relacionado con la Organización Internacional para la Estandarización (ISO), que hacer referencia al modelo de interconexión de sistemas abiertos (OSI), que especifica un protocolo de capas que tiene hasta siete capas, indicados como física, enlace, red, transporte, sesión, presentación y aplicación.

En relación a esto, el protocolo de red 1-Wire tiene 5 capas, las cuales son: Presentación, Transporte, Red, Enlace y Física.

Figura 20. **Protocolo de Red 1-Wire**



Fuente: web.archive.org/web/20090127003835/http://www.maxim-ic.com/products/ibutton/ibuttons/standard.pdf. Consulta: 05 de septiembre de 2014.

4.1.1. Capa física

Esta capa define las características eléctricas, niveles lógicos de voltajes y la sincrónica en las comunicaciones entre el maestro y los dispositivos esclavos.

Este tema se hace referencia en la sección 2.1 Señalización 1-Wire.

4.1.2. Capa de enlace

Esta capa define las funciones básicas de comunicación de los dispositivos 1-Wire. Las funciones de reset, pulso de presencia, y transferencia de bit (escritura y lectura). Los detalles de estas funciones también están en la sección 2.1. Señalización 1-Wire.

4.1.3. Capa de red

Esta capa se encarga de la identificación de cada uno de los dispositivos y de la capacidad de la asociación en red. Como se ha mencionado anteriormente, cada dispositivo esclavo 1-Wire, tiene un código único de identificación guardado en la ROM. Los detalles del código ROM están en la sección 2.2. Debido a la ROM, todos los comandos que se refieren a la capa de red, también son llamados comandos ROM. Los comandos ROM son estudiados en detalle en la sección 2.3.2. Comandos ROM.

4.1.4. Capa de transporte

Esta capa es la responsable de la transferencia de datos entre los segmentos no-ROM de los dispositivos esclavos y el maestro, y la transferencia de datos desde la memoria de trabajo, a las áreas de almacenamiento finales y registros especiales de la memoria del dispositivo esclavo. Los comandos que se encargan de esto son llamados de funciones. Debido a que hay una gran variedad de dispositivos esclavos en el mercado, y cada uno tiene funciones específicas, los comandos de funciones podrían variar entre uno y otro. Se estudian los comandos de funciones del dispositivo DS18B20 en la sección 2.3.3 Comando de funciones.

4.1.5. Capa de presentación

Las capas de enlace, red y transporte son la base de la capa de presentación. Esta capa se encarga de la representación de la información. Permite cifrar datos, actúa como traductor entre los dispositivos de la red.

4.2. Descripción de comandos

Para operar independientemente y también, parte del bus, los dispositivos 1-Wire soportan los siguientes comandos de red ROM: READ ROM, SKIP ROM, MATCH ROM y SEARCH ROM. Después de la ejecución de cualquier comando ROM, la capa de transporte es alcanzada. El comando READ ROM (33h) es usado para identificar un dispositivo en el bus 1-Wire o para averiguar si varios dispositivos están conectados al mismo tiempo.

Después de enviar este comando, el maestro del bus deberá generar 64 intervalos de tiempo de lectura. Si varios dispositivos están conectados al bus, ninguna lectura proporcionará un CRC válido, en este caso, el comando SEARCH ROM (F0h), debe ser utilizado para determinar el contenido de la ROM de los dispositivos antes de que puedan ser direccionados.

Si el contenido de la ROM no nos interesa porque solamente existe un dispositivo conectado al bus, la búsqueda se puede omitir mediante el comando SKIP ROM (CCh), Inmediatamente después de este comando, el dispositivo alcanza la capa de transporte.

El comando MATCH ROM es utilizado para direccionar un solo dispositivo si varios están conectados en paralelo. El contenido ROM actúa como la dirección de cada dispositivo para así activar exactamente ese dispositivo. Que un mismo código ROM de 64 bits, se repita en algún dispositivo, es casi imposible debido al estricto control en su fabricación. Si dos dispositivos tuvieran los mismos códigos de serie, tendrían códigos de familia distintos. De esta manera cualquier confusión o conflicto es evitado.

El comando MATCH ROM (55h), requiere el contenido de la ROM del dispositivo que se quiere direccionar, para ser enviados por el maestro durante los 64 intervalos de tiempo siguientes el comando. La secuencia de los bits debe de ser igual que como cuando fueron entregados al ser leída la ROM del dispositivo al que se quiera direccionar, es decir, los bits menos significativos primero, comenzando con el código de familia, seguido del número de serie y, por último el CRC. Todos los dispositivos cuyo contenido ROM no coincida con el código enviado permanecerán inactivos hasta recibir un pulso de reset.

4.3. Comando SEARCH ROM

Si el maestro no llegara a conocer el número serial de los dispositivos conectados al bus 1-Wire, es posible direccionar un único dispositivo a la vez. Esto es posible usando el comando SEARCH ROM (F0h). Este comando actúa como READ ROM combinado con MATCH ROM. Todos los dispositivos envían el valor real y el complemento del bit ROM actual, durante dos intervalos de tiempo de lectura después del comando SEARCH ROM. Si todos los dispositivos tienen un 0 como valor en el bit actual, la lectura será 0 1.Si el valor del bit en la posición actual fuese 1, el resultado seria 1 0. Si ambos, 1 y 0 se producen en esta posición, la lectura se traduciría en 0 y 0, indicando conflicto.

El maestro ahora tiene que enviar el valor del bit, 1 o 0, para seleccionar los dispositivos que seguirán en el proceso de selección. Todos los dispositivos descartados estarán inactivos hasta recibir un pulso de reset. Después de la primera etapa de selección, 63 ciclos de lectura/selección seguirán, hasta que finalmente el maestro ha aprendido el código ROM de un dispositivo y simultáneamente la direcciona. Cada etapa de selección consiste en dos intervalos de tiempo de lectura y un intervalo de tiempo de escritura.

El proceso completo de aprendizaje y simultáneo direccionamiento es aproximadamente tres veces la longitud del comando MATCH ROM, pero permite la selección de todos los dispositivos conectados de forma secuencial, sin conocer los valores ROM de antemano. En una aplicación en donde existen varios dispositivos conectados al bus 1-Wire, es más eficiente para el maestro evaluar todos los contenidos de la ROM con el comando SEARCH ROM y luego usar el comando MATCH ROM para direccionar un dispositivo en específico. Si la aplicación requiere la identificación y comunicación constante con los nuevos dispositivos porque son conectados y desconectados del bus constantemente, será necesario utilizar el comando SEARCH ROM, para identificar y direccionar cada nuevo dispositivo conectado.

De todos los comandos ROM el comando SEARCH ROM es el más complejo de utilizar. El siguiente ejemplo se utiliza para ilustrar el uso del comando paso a paso.

Cuatro dispositivos son conectados al bus 1-Wire. El contenido binario almacenado en la ROM es:

- Dispositivo 1: XXXXXX10101100
- Dispositivo 2: XXXXXX01010101
- Dispositivo 3: XXXXXX10101111
- Dispositivo 4: XXXXXX10001000

Las X representan los bits superiores restantes. Se muestran los 8 bits más bajos de los contenidos de la ROM de cada dispositivo. El bit menos significativo es el que está más a la derecha en esta representación. El proceso de búsqueda se ejecuta como sigue:

a) El maestro comienza la secuencia de inicialización con un pulso de reset. Los dispositivos responderán con un pulso de presencia.

b) El maestro entonces emitirá el comando SEARCH ROM.

c) El maestro lee un bit del bus 1-Wire. Cada dispositivo responderá, colocando el valor del primer bit de su respectivo código ROM en el bus 1-Wire. Dispositivos 1 y 4 deberán colocar 0 en el bus, es decir, pondrán en bajo el bus. Dispositivos 2 y 3 deberán mandar 1 permitiendo que la línea se mantenga en alto. El resultado final es el de aplicar AND lógica a todos los dispositivos de la línea; por lo tanto, el resultado es 0. El maestro pasará a leer el siguiente bit, ya que el comando SEARCH ROM está siendo ejecutado, todos los dispositivos responderán a esta segunda lectura colocando el complemento del primer bit de su respectivo dato ROM en el bus 1-Wire. Dispositivos 1 y 4 deberán mandar 1 y los 2 y 3 deberán mandar un 0. Por lo tanto, el bus 1-Wire deberá ponerse en bajo. El maestro lee de nuevo un 0, como el complemento del primer bit de datos ROM. Esto le dice al maestro que hay dispositivos en el bus que tienen un 0 en la primera posición y otros que tienen 1. Si todos los dispositivos tenían un 0 en esa posición, la lectura hubiese sido 0 1. Si todos los dispositivos tenían un 1 en esa posición, la lectura hubiese sido 1 0.

d) El maestro ahora decide escribir un 0 en el bus 1-Wire. Esto deselecciona los dispositivos 2 y 3 para el resto de la búsqueda, dejando solamente los dispositivos 1 y 4 particionado en el proceso de búsqueda.

e) El maestro realiza dos lecturas más y recibe un 0 y un 1. Esto indica que todos los dispositivos activos tienen un 0 en esa posición en el código ROM.

f) El maestro, entonces escribe un 0 para mantener activos los dispositivos 1 y 4.

g) El maestro ejecuta dos lecturas y recibe 0 0. Esto indica nuevamente que existe 1 y 0 en el tercer bit de la ROM de los dispositivos activos.

h) El maestro de nuevo escribe un 0. Esto deshabilita el dispositivo 1, dejando el dispositivo 4 como el único dispositivo activo.

i) Las siguientes lecturas hasta el final del código ROM no presentarán bits de conflicto, por lo tanto, directamente el dispositivo seleccionado le dirá el código ROM al maestro. Después de haber aprendido cualquier bit ROM nuevo, el maestro tiene que volver a enviar este bit para mantener seleccionado el dispositivo. Tan pronto como todos los bits de la ROM del dispositivo son conocidos y el último bit es reenviado por el maestro, el dispositivo está listo para aceptar un comando de la capa de transporte.

j) El maestro debe aprender el código ROM de los otros dispositivos. Por lo tanto, se inicia otra secuencia de búsqueda ROM, repitiendo los pasos desde a hasta g.

k) En el paso h, ahora se escribirá un 1. Esto deshabilita el dispositivo 4, dejando al dispositivo 1 activo.

l) Al igual que en el paso i, se seguirá leyendo hasta el final del código, sin que se encuentren bits de conflicto. Esto completa la segunda secuencia SEARCH ROM, donde el maestro ha aprendido otro código ROM.

m) El maestro debe aprender otro código ROM, por lo tanto se debe comenzar otra secuencia de búsqueda ROM, repitiendo los pasos desde a hasta c.

n) En el paso d, ahora se deberá escribir un 1. Esto desactiva los dispositivos 1 y 4, dejando activos los dispositivos 2 y 3.

o) El maestro manda dos ciclos de lectura y recibe dos ceros, indicando un bit de conflicto.

p) El maestro nuevamente decide escribir un 0. Esto deshabilita el dispositivo 3, dejando activo solamente el dispositivo 2.

q) Al igual que en el paso i, las siguientes lecturas hasta finalizar el código ROM, no presentaran bit de conflicto. Esto completa la tercera secuencia de búsqueda ROM, donde el maestro ha aprendido otro código ROM.

r) El maestro debe aprender otro código ROM, entonces se debe iniciar otra secuencia de búsqueda ROM mediante la repetición de los pasos m al o.

s) En el paso p, ahora se debe escribir un 1, esto deshabilita el dispositivo 2, dejando el dispositivo 3 activo.

t) Como en el paso q, se seguirá leyendo hasta finalizar el código ROM sin tener ningún bit de conflicto. Esto completa la cuarta secuencia de búsqueda ROM, donde el maestro ha aprendido el último código ROM.

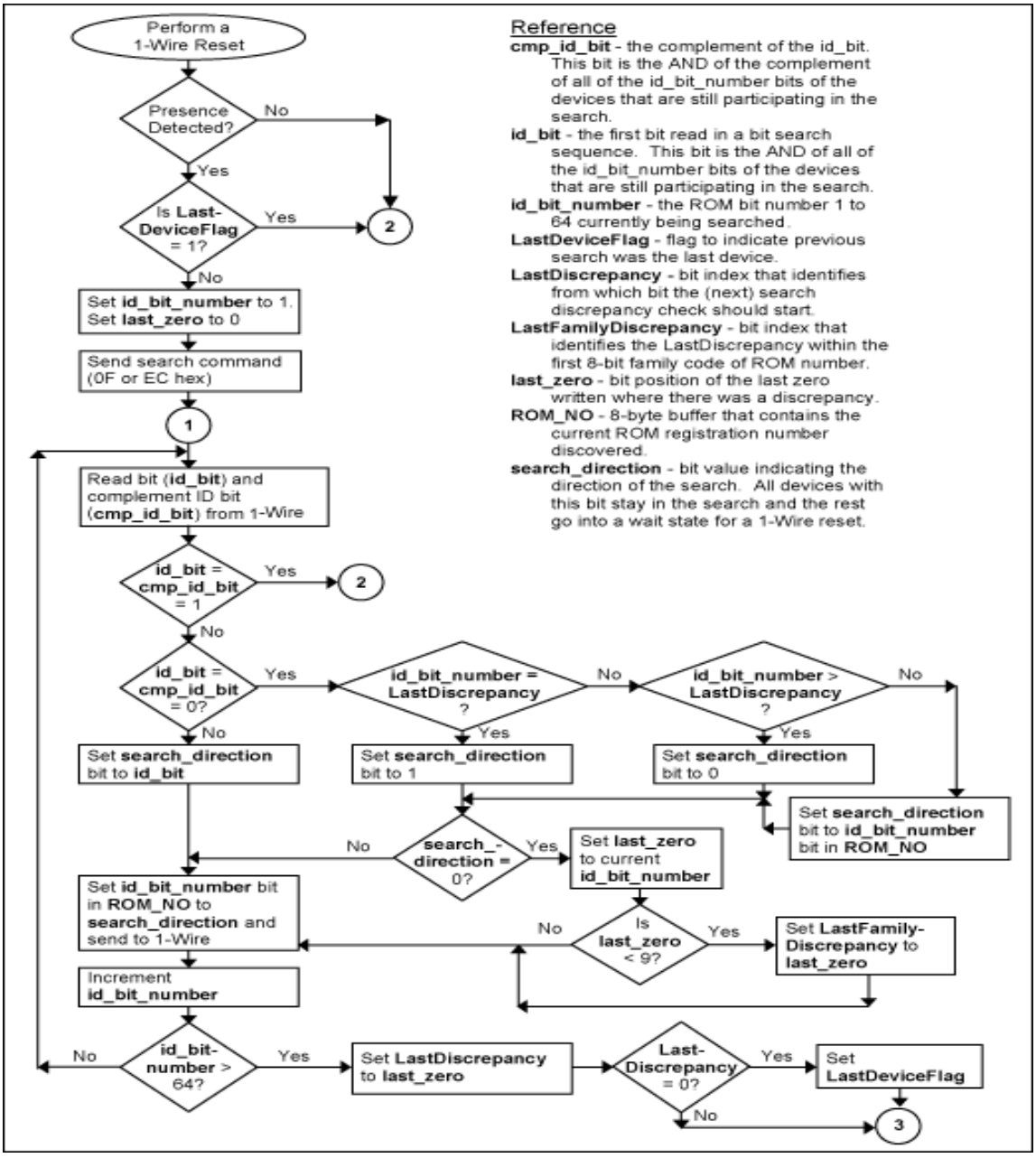
El principio general del proceso de búsqueda es la desactivación de un dispositivo después de cada posición en el que esté un bit de conflicto. Al final de cada secuencia de búsqueda ROM, el maestro ha aprendido otro código ROM.

El siguiente paso es el mismo que el anterior hasta el punto de la última decisión. En este punto el maestro va en la dirección opuesta y continúa. Si no se encuentra otro conflicto se escribe 0, y así sucesivamente. Después de que ambas maneras en los bits de conflictos más significativos son seguidos hasta el final, el maestro va de la misma manera que antes, pero decidiendo opuestamente en los bits menos significativos, así sucesivamente, hasta que el código ROM es completamente leído.

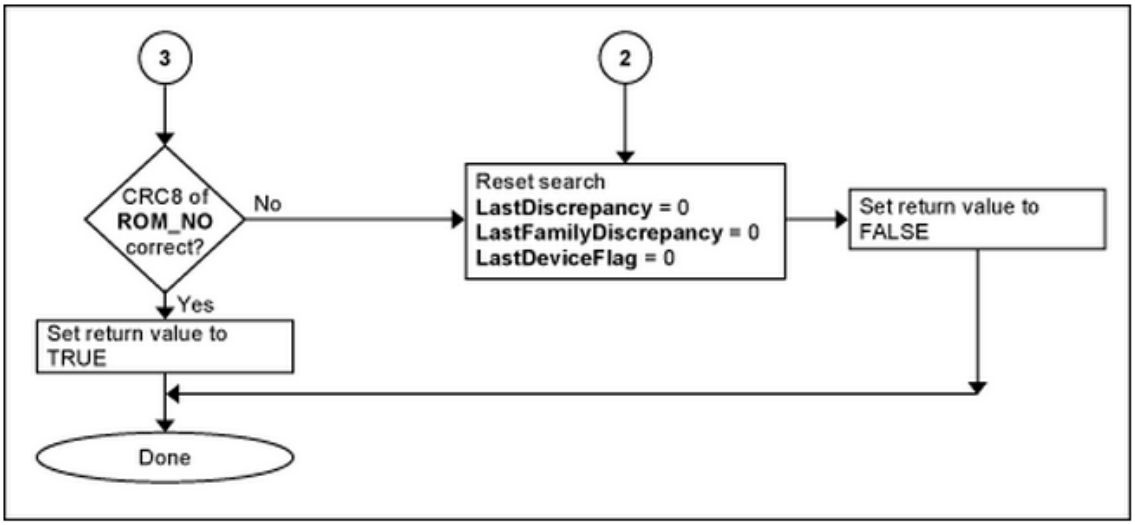
Un diagrama de flujo optimizado del algoritmo SEARCH ROM es mostrado en la figura 21. Este dibujo explica cómo realizar una búsqueda general del código ROM. Para el propósito de este diagrama de flujo, el dato del código ROM es acumulado en un arreglo de bits, con los bits numerados del 1 al 64. Se debe de inicializar este arreglo antes de iniciar el sistema 1-Wire. Un llamado al FIRST, restablece la búsqueda al principio y se identifica el primer código ROM, y una llama a NEXT, identifica códigos ROM sucesivos. Un valor falso devuelto indica que no hay más códigos ROM que encontrar.

El tiempo requerido para aprender un código ROM, es de 960 µs + (8+3*64)*61µs =13.16 ms. Por lo tanto es posible identificar hasta 75 dispositivos por segundo.

Figura 21. **Diagrama de flujo de búsqueda de código ROM**



Continuación de la figura 21.



Fuente: http://www.maximintegrated.com/en/app-notes/index.mvp/id/187. Consulta: 19 de octubre de 2014.

5. DISEÑO DE PROTOTIPO DE DISPOSITIVO PARA LA MEDICIÓN DE TEMPERATURA Y CONTROL DE ACCESO EN CUARTOS REFRIGERADOS

El prototipo plantea la resolución de los problemas más comunes en el transporte y almacenaje de productores refrigerados o perecederos: la medición de temperatura en cada uno de los ambientes, la verificación de puertas abiertas o cerradas y la identificación del personal que tiene acceso a los diferentes ambientes.

En los capítulos uno al cuatro de este documento, se expone los conceptos sobre los cuales se fundamenta el diseño presentado en este capítulo. Se describe el funcionamiento y especificaciones del dispositivo, se presenta el diagrama electrónico, el diseño de PCB, el diagrama de flujo del programa del microntrolador y el presupuesto para realizar dicho prototipo.

5.1. Funcionamiento y especificaciones

El diseño presentado tiene la capacidad de medir la temperatura de cuatro ambientes distintos, gracias a que puede comunicarse con cuatro sensores DS18B20 al mismo tiempo. Para el correcto funcionamiento de la LAN 1-Wire se debe conectar el pin GND de todos los sensores de la red al borne GND del dispositivo. El pin DQ de todos los sensores debe ir conectado al dispositivo al borne ROW, y el pin VDD de cada uno de los sensores debe ir conectado al borne +5V del dispositivo. Ver figura 23.

El prototipo es capaz de identificar usuarios y decidir qué usuario está autorizado y cual no lo está. Esto se hace mediante el dispositivo conocido como iButton. Este trae un código único y mediante el protocolo 1-Wire se comunica con el dispositivo. En el diseño presentado, solamente plantea que cualquier iButton que es conectado al dispositivo sea reconocido, pero solo uno es el que toma como válido, este número ROM de 64 bits que toma como válido, está guardado, en este diseño, en la memoria ROM del microcontrolador en las direcciones del 0x00 al 0x07. Este código puede ser modificado en el código para reconocer un iButton que el usuario quisiera como válido. Para leer el iButton se debe conectar el borne GND del iButton al borne GND del dispositivo y el borne DQ del iButton al borne o terminal KEY del dispositivo. Ver figura 23.

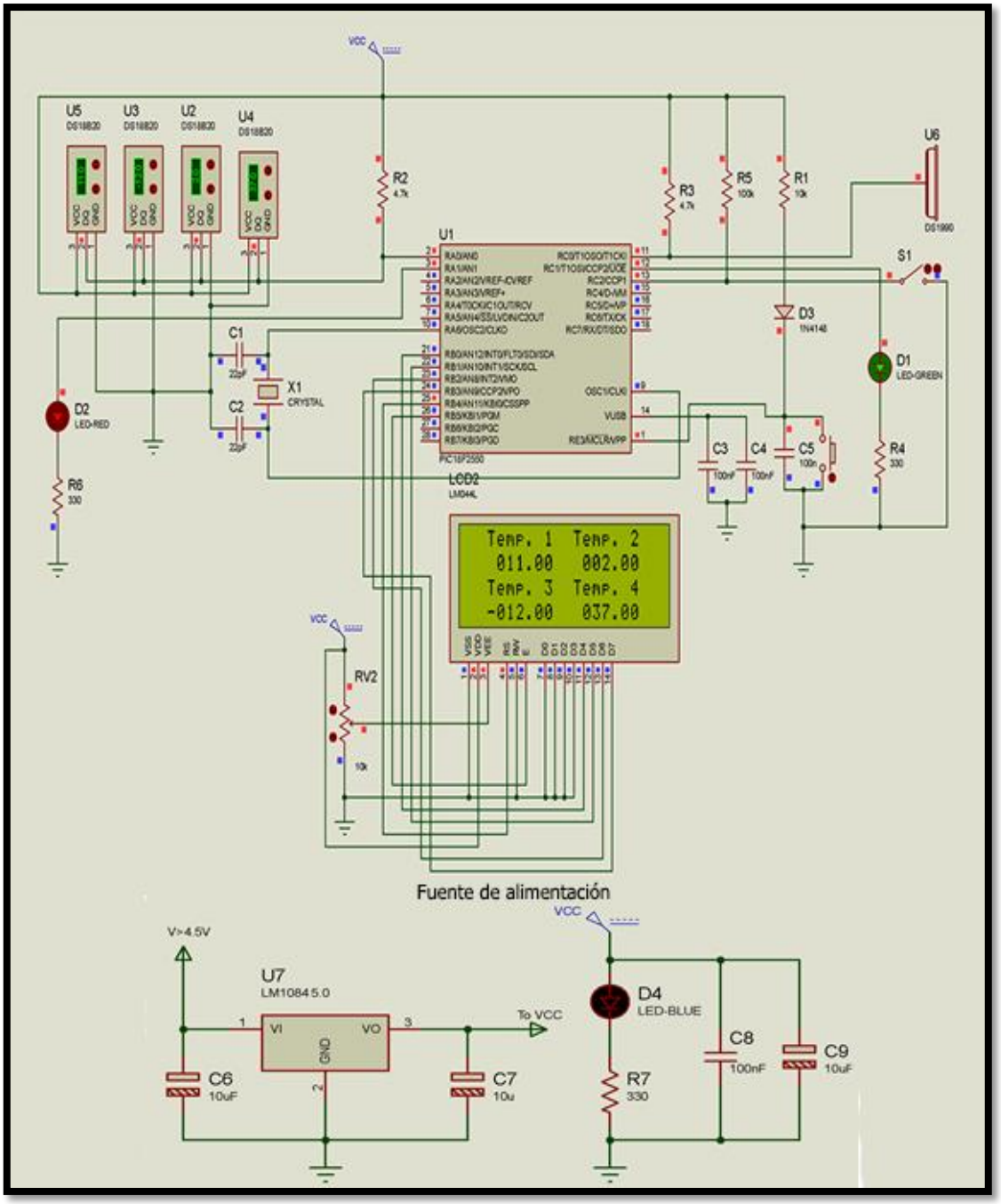
Por último, este diseño plantea verificar el estado de una puerta; abierta o cerrada. Para ello se debe usar un sensor adecuado, por ejemplo: finales de carrera o sensor magnético para puertas. El sensor que se elija debe ir conectado a la entrada IN1. Ver figura 23. Para que detecte que la puerta está abierta, el sensor debe mandar un uno lógico o alta impedancia a la entrada IN1. Para que detecte que la puerta está cerrada, el sensor elegido, debe mandar un 0 lógico o la señal GND a la entrada IN1.

A continuación se describen las especificaciones del prototipo.

- Voltaje de entrada: 4.5 V a 25 V
- Corriente máxima en terminal +5V: 4A
- Cantidad de dispositivos DS18B20 soportados : 4
- Cantidad de llaves iButton validas: 1
- Indicador de puerta abierta: Si
- Indicador de usuario autorizado: Si
- Tiempo máximo en la medición de 4 sensores de temperatura: 3s

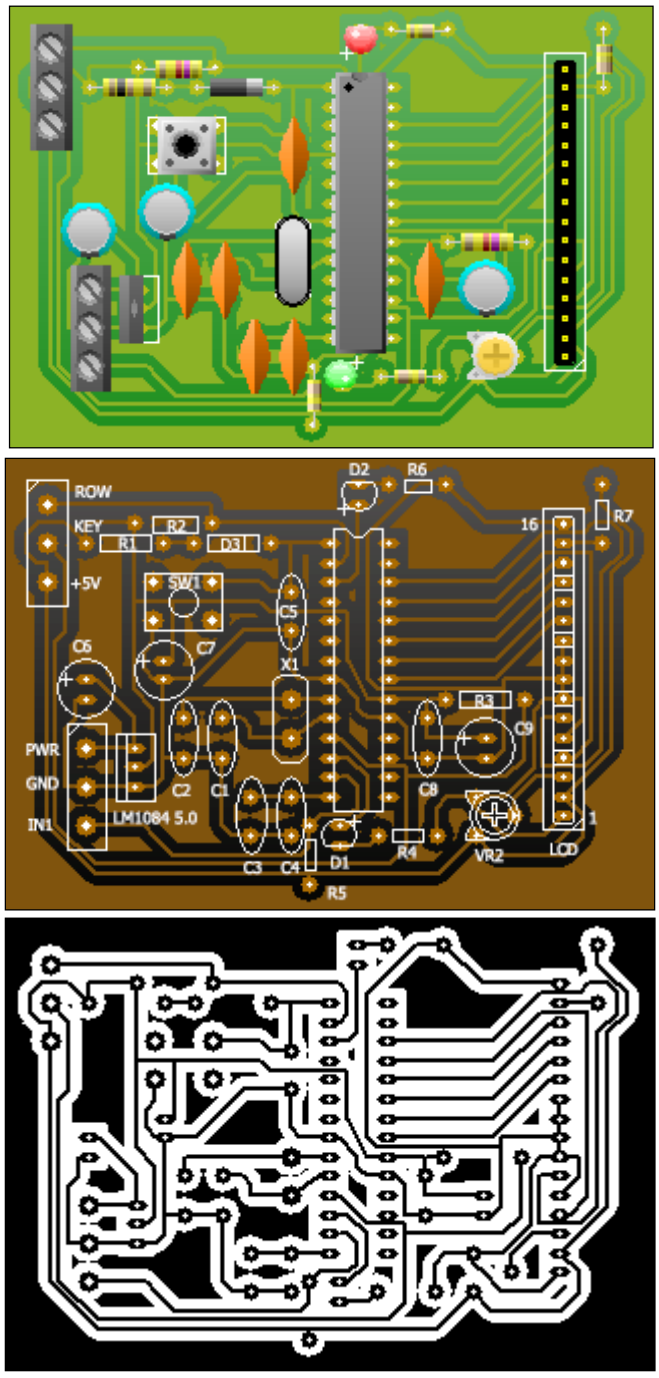
En la figura 22 se muestra el diagrama electrónico del dispositivo, asimismo, en la 23 se presenta el diseño de PCB y en la 24 se describe el diagrama de flujo del programa del microcontrolador PIC18F2550.

Figura 22. **Diagrama electrónico**



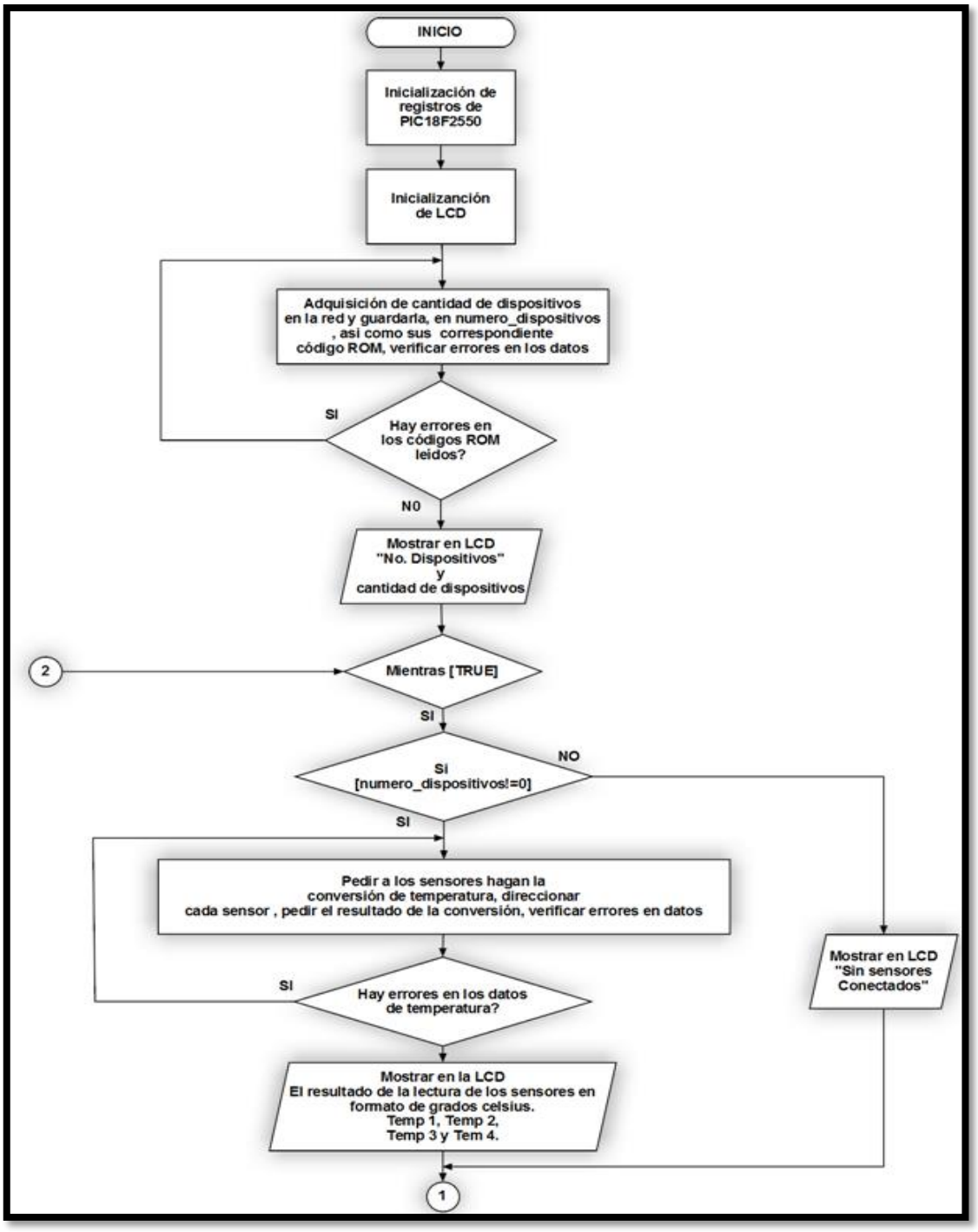
Fuente: elaboración propia, con programa de NI Multisim 11.

Figura 23. **Diseño de PCB**

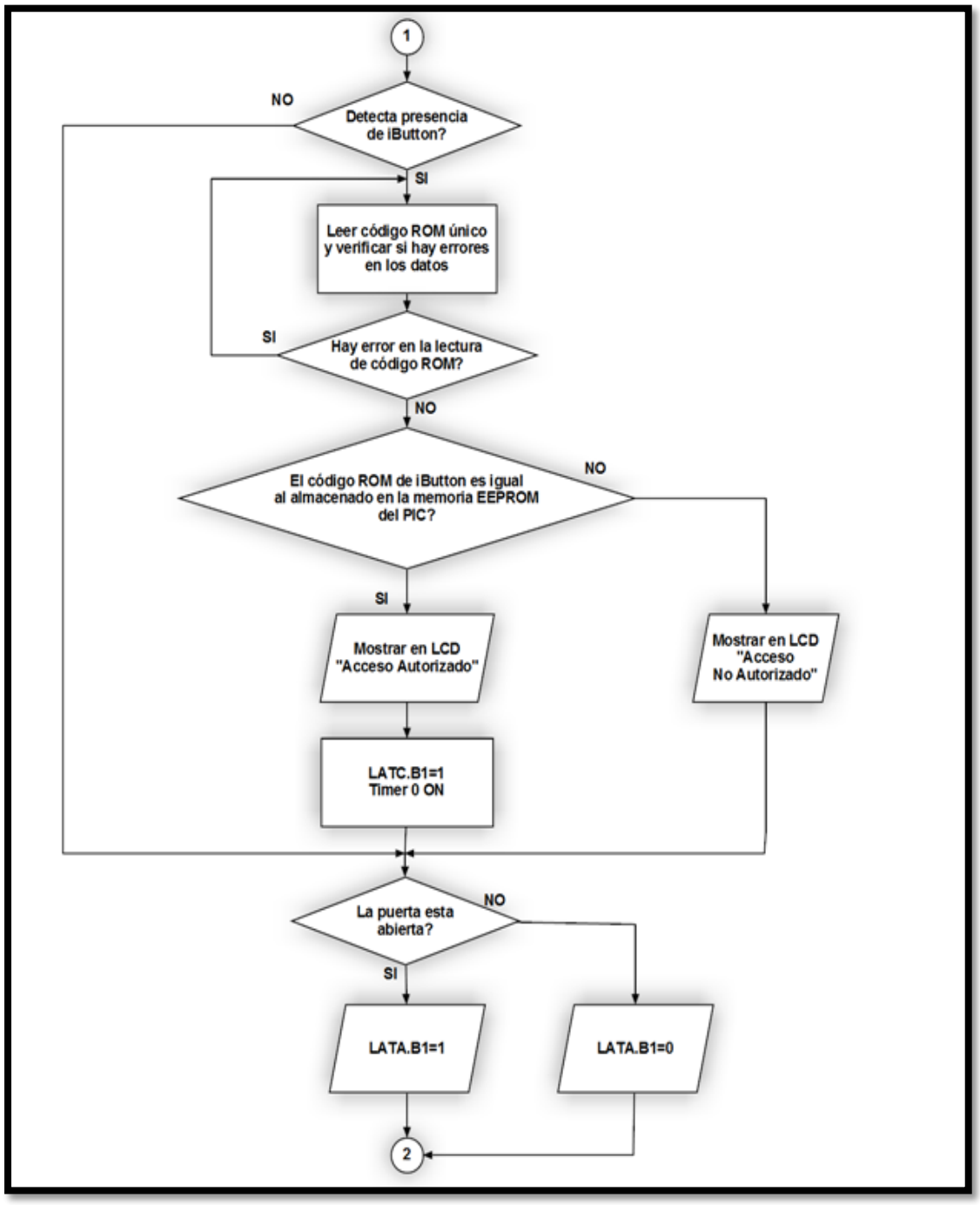


Fuente: elaboración propia, con programa de PCB Wizard 3.

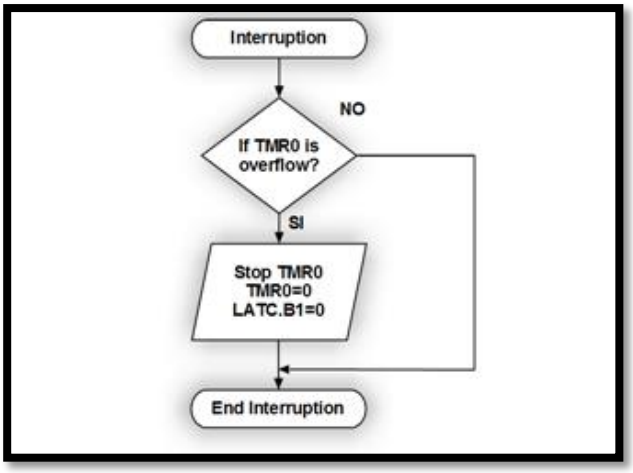
Figura 24. **Diagrama de flujo**



Continuación de la figura 24.



Continuación de la figura 24.



Fuente: elaboración propia.

En la tabla IV se muestra el presupuesto para realizar el prototipo. Se detallan los componentes electrónicos a usar, así como los elementos utilizados para su presentación.

Tabla IV. **Presupuesto**

Componente	Cantidad de dispositivos	Precio c/u en Q	Precio Total en Q
Sensor DS18B20	4	15	60
R2, R3: Resistencias de 4.7KΩ	2	1	2
R5: Resistencia de 100KΩ	1	1	1
R1: Resistencia de 10KΩ	1	1	1
R4, R6, R7: Resistencia de 330Ω	3	1	3
X1: Cristal de Cuarzo de 8Mhz	1	5	5
C1, C2: Capacitor de 22pF	2	1	2
C3,C4,C5, C8: Capacitor de 100nF	4	1	4
C6,C7,C9: Capacitor electrolítico de 10µF	3	1	3
D3: Diodo 1N4148	1	1	1
Microcontrolador 18F2550	1	75	75
Pantalla LCD 4X20	1	135	135
LM1084 IT-5.0	1	15	15
VR2: Potenciómetro 5KΩ	1	3	3
D1: LED Verde	1	2	2
D2: LED Rojo	1	2	2
SW1: Push Botton	1	2	2
iButton DS1990A	1	25	25
Lector para iButton	1	10	10
Caja plástica	1	20	20
Plug de alimentación 5.5mm x 2.1mm	1	5	5
Jack de alimentación 5.5mm x 2.1mm	1	5	5
Jack RJ-11 simple para placa	1	10	10
Total	35	337	**391**

Fuente: elaboración propia.

CONCLUSIONES

1. Un protocolo de comunicaciones es un conjunto de reglas y normas que permiten que dos o más entidades de un sistema de comunicación se comuniquen entre ellos, para transmitir información por medio de cualquier tipo de variación de una magnitud física.

2. 1-Wire es un protocolo bidireccional, Half-Duplex, que transporta datos y alimenta a los dispositivos que se conectan al bus por medio de una única línea o cable y tierra. Acepta solamente un maestro en el bus y uno o más esclavos.

3. El protocolo de comunicación 1-Wire tiene dos velocidades de transmisión de datos: estándar y overdrive.

4. Toda comunicación con protocolo 1-Wire comienza con una secuencia de inicialización que consiste en un pulso de reset desde el maestro seguido por un pulso de presencia desde el esclavo.

5. Cada dispositivo esclavo 1-Wire tiene un código único de 64 bits guardado en su memoria ROM. Este código contiene información de la función que tiene el esclavo, un número serial de 48 bits y un CRC de 8 bits para la detección de errores en la transmisión del código.

6. El esquema de detección más eficiente para localizar errores en un flujo de datos en serie, con la mínima cantidad de hardware es la comprobación de redundancia cíclica.

7. Todos los dispositivos 1-Wire, son diseñados para operar en un entorno de red, esto amplia el campo de aplicación a una mayor capacidad de almacenamiento de datos distribuidos usando solo una línea de datos común para el maestro.

8. En relación al modelo de Interconexión de Sistemas Abiertos (OSI), el protocolo de red 1-Wire está formado por cinco capas, las cuales son: presentación, transporte, red, enlace y física.

9. La implementación del protocolo de red 1-Wire en lenguaje C, es factible y su estudio a profundidad evita el uso integrados controladores de bus que aumentan de forma significativa el costo de su implementación en un proyecto en donde se utilice dicho protocolo.

10. Las ventaja en el campo del protocolo 1-Wire son: la alta inmunidad al ruido eléctrico, la poca probabilidad de tomar un dato erróneo como correcto debido al método de comprobación de redundancia cíclica, y por último, la posibilidad de controlar varios dispositivos por medio de un solo cable.

RECOMENDACIONES

1. En la implementación del protocolo 1-Wire tomar en cuenta la cantidad de corriente consumida por cada uno de los componentes que conforman la LAN 1-Wire.

2. Continuar con la investigación acerca del protocolo de comunicación 1-Wire, pues es un protocolo relativamente joven con poca información disponible y que está cobrando importancia en la industria electrónica.

3. Tomar en cuenta que cuando la cantidad de dispositivos en una LAN 1-Wire es variable, debería ejecutarse el algoritmo SEARCH ROM cada vez que se necesite conocer algún dato de cualquier dispositivo esclavo conectado a la red.

4. Tomar en cuenta que cuando la cantidad de dispositivos conectados a la LAN 1-Wire es estática, debería ejecutarse el algoritmo SEARCH ROM solamente al inicio de la rutina principal del programa.

5. Es importante que el maestro de la LAN 1-Wire verifique cada dato recibido de parte de los dispositivos esclavos, por medio del método de comprobación de redundancia cíclica, para garantizar que el dato leído sea el correcto.

6. En la implementación de una LAN 1-Wire, hay que tomar en cuenta que la cantidad de dispositivos conectados a la red sea menor a 100 y la distancia de los dispositivos esclavos al maestro sea menor a 200 metros.

BIBLIOGRAFÍA

1. Cisco Networking Academy. Acceso a la red. [en línea] <http://www.ie.itcr.ac.cr/acotoc/CISCO/R&S%20CCNA1/R&S_CCNA1_ITN_Chapter4_Acceso%20a%20la%20red.pdf> [Consulta: 16 de agosto de 2014].

2. Conexión de una pantalla LCD a un microcontrolador. [en línea]. <server-die.alc.upv.es/asignaturas/lsed/2002-03/Pantallas_LCD/LCD.pdf> [Consulta: 7 de agosto de 2014].

3. Maxim Integrated. Application note 27. [en línea] < http://www.maximintegrated.com/en/app-notes/index.mvp/id/27> [Consulta: 23 de septiembre de 2014].

4. _____. DS18B20 [en línea]. <http://www.maximintegrated.com/en/products/analog/sensors-and-sensor-interface/DS18B20.html> [Consulta: 17 de agosto de 2014].

5. _____. iButton TM Overview. [en línea]. < web.archive.org/web/20090127003835/http://www.maxim-ic.com/products/ibutton/ibuttons/standard.pdf > [Consulta: 30 de agosto de 2014].

6. _____. note 187. [en línea]. <http://www.maximintegrated.com/en/app-notes/index.mvp/id/187> [Consulta: 19 de octubre de 2014].

7. Microchip. PIC18F4550 [en línea]. < http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010300> [Consulta: 16 de agosto de 2014].

8. MIKROELECTRONIKA. *Programación de los microcontroladores – Microcontroladores PIC – Programación en C con ejemplos* [en línea] < http://www.mikroe.com/chapters/view/80/capitulo-2-programacion-de-los-microcontroladores> [Consulta: 10 de octubre de 2014].

APÉNDICE

Código de programa del microcontrolador PIC18F2550.[1]

```
/*****************************************************************************************************************************************/
/*Proyecto: 1-Wire aplicado a la medición de temperatura y control de acceso                                                             */
/*Autor: Mynor Geovanny Mendoza Ordoñez                                                                                                  */
/*Fecha: Guatemala 23 de septiembre de 2014                                                                                              */
/*Microcontrolador: PIC18F2550                                                                                                           */
/*Reloj: 8MHZ                                                                                                                            */
/*Distribución de pines:                                                                                                                 */
/* LCD:                                                                                                                                  */
/*    RS: RB4                                                                                                                            */
/*    EN: RB5                                                                                                                            */
/*    D4: RB0                                                                                                                            */
/*    D5: RB1                                                                                                                            */
/*    D6: RB2                                                                                                                            */
/*    D7: RB3                                                                                                                            */
/* Pines 1-Wire                                                                                                                          */
/*    RA0 (maestro red DS18B20)                                                                                                          */
/*    RC0 (maestro iButton)                                                                                                              */
/* Protocolo 1-Wire                                                                                                                      */
/*    Velocidad Standar                                                                                                                  */
/*    Alimentacion red DS18B20: Standar                                                                                                  */
/*    Alimentacion iButton: Parasite Power Mode                                                                                          */
/* Direccion de Memoria EEPROM utilizada para guardar el codigo ROM del iButton                                                          */
/* 0x00-0x07                                                                                                                             */
/*****************************************************************************************************************************************/

// LCD module connections
sbit LCD_RS at LATB4_bit;
sbit LCD_EN at LATB5_bit;
sbit LCD_D4 at LATB0_bit;
sbit LCD_D5 at LATB1_bit;
sbit LCD_D6 at LATB2_bit;
```

[1] Debido a políticas de privacidad, solamente se muestra el código de la función principal. Las demás subrutinas del programa aparecen, pero sin mostrar su código.

Continuación del apéndice.

```
sbit LCD_D7 at LATB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
// End LCD module connections

//Definiciones

#define FALSE 0
#define TRUE  1

// global search state
unsigned char ROM_NO[8];              //Codigo ROM
unsigned char ROM_CODE[5][8];         //Guarda el valor de los 5 dispositivos 1_wire
unsigned char scratchpad[5][9];       //Guarda el valor del Scratchpad
char Codigo_Rom_Unico[8];             //Almacena el valor del codigo ROM iButton
char TEMPS[]=" 000.00";               //Guarda el valor de la temperatura
char TEMPS_1[]=" 000.00";
char TEMPS_2[]=" 000.00";
char TEMPS_3[]=" 000.00";
char TEMPS_4[]=" 000.00";

int LastDiscrepancy;
int LastFamilyDiscrepancy;
int LastDeviceFlag;
unsigned char crc8;


char txt[3];
const unsigned short TEMP_RESOLUTION = 12;


void Port_Init(void){ …
}

void Interrupt(){…
}
```

Continuación del apéndice.

```
void Display_Temperature(unsigned int temp2write) {…
}


unsigned short ROW_RESET(unsigned short *port){…
}

void ROW_Write_bit(unsigned short *port, unsigned short pulso){…
}

void ROW_Write_Byte(unsigned short *port, unsigned short dato){..
}

unsigned short ROW_Read_bit(unsigned short *port){…
}

char ROW_Read_Byte(unsigned short *port){…
}

static unsigned char dscrc_table[] = {…
}

unsigned char ROW_crc_scratchpad(unsigned char codigo[9]){…
}

unsigned char docrc8(unsigned char value){…
}

unsigned short ROW_Search_ROM_Read(unsigned short *port){…
}

unsigned short ROW_First(unsigned short *port){…
}

unsigned short ROW_Next(unsigned short *port){…
}

unsigned short ROW_Codigo_ROM(unsigned short *port){…
}

void main() {
```

Continuación del apéndice.

```
    unsigned short i=0;
    unsigned short j=0;
    unsigned numero_dispositivos=0;
    unsigned temp;                      //Variables del sensor de temperatura
    char memoria_trabajo[9];

    ADCON1=0b00001111;          //Todos los puertos son digitales


    T0CON=0b00000111;           //Counter OFF; Prescaler 1:256
    TMR0H=0x0B;
    TMR0L=0xDC;
    GIE_bit=1;
    TMR0IE_bit=1;
    T0CON.TMR0ON=0;             //Stop Timer 0;
    TMR0IF_bit=0;               //Timer 0 interrup bajo

    Port_Init();                // Incializacion de los puertos
    Lcd_Init();                 //Inicializar la LCD
    Lcd_Cmd(_LCD_CLEAR);        // Clear display
    Lcd_Cmd(_LCD_CURSOR_OFF);   // Cursor off


    Lcd_Out(2,6,"BIENVENIDO");
    Lcd_Out(3,5,"TEMP G2M-1.0");
    Delay_ms(2000);
    Lcd_Cmd(_LCD_CLEAR);        // Clear display
    numero_dispositivos=ROW_Codigo_ROM(&PORTA); //Adquiere los codigos ROM unicos, y la cantidad  de
dispositivos en la Red 1-Wire.
    ByteToHex(numero_dispositivos, txt);
    Lcd_Out(2,3,"NO. DISPOSITIVOS");
    Lcd_Out(3,10,txt);
    Delay_ms(2000);
    Lcd_Cmd(_LCD_CLEAR);        // Clear display

    while(1){

        asm{
            CLRWDT                      //Resetear el watchdog timer
        }
```

Continuación del apéndice.

```
        if(numero_dispositivos!=0){ //Si hay dispositivos en la red
            //Lectura de Temperatura
            for(i=0;i<numero_dispositivos;i++){

                Lcd_Out(1,3,"Temp. 1");
                Lcd_Out(2,3,TEMPS_1);

                Lcd_Out(1,12,"Temp. 2");
                Lcd_Out(2,12,TEMPS_2);

                Lcd_Out(3,3,"Temp. 3");
                Lcd_Out(4,3,TEMPS_3);

                Lcd_Out(3,12,"Temp. 4");
                Lcd_Out(4,12,TEMPS_4);

                ROW_RESET(&PORTA);                      //Reset
                Delay_ms(1);
                ROW_Write_Byte(&PORTA,0x55);            //Comando Match ROM

                for(j=0;j<8;j++){

                    ROW_Write_Byte(&PORTA,ROM_CODE[i][j]);      //Codigo ROM

                }

                ROW_Write_Byte(&PORTA,0x44);            //Comando CONVERT_T
                Delay_ms(750);                          //Espera a que se realize la conversion de temperatura dentro
del DS18B20
                ROW_RESET(&PORTA);                      //Reset

                do{
                    ROW_Write_Byte(&PORTA,0x55);            //Comando March ROM
                    for(j=0;j<8;j++){
                    ROW_Write_Byte(&PORTA,ROM_CODE[i][j]);      //Codigo ROM
                    }

                    ROW_Write_Byte(&PORTA,0xBE);            //Comando Scratchpad
                    for(j=0;j<9;j=j+1){
                        scratchpad[i][j]=ROW_Read_Byte(&PORTA);
                    }
```

Continuación del apéndice.

```
ROW_RESET(&PORTA);                    //Reset

            for(j=0;j<9;j=j+1){
                memoria_trabajo[j]=scratchpad[i][j];
            }

        }while(ROW_crc_scratchpad(memoria_trabajo)!=0);

        Display_Temperature((scratchpad[i][1]<<8)+scratchpad[i][0]);

        switch(i){
            case 0:
                    for(j=0;j<8;j++){
                        TEMPS_1[j]=TEMPS[j];
                    }; break;
            case 1:

                    for(j=0;j<8;j++){
                        TEMPS_2[j]=TEMPS[j];
                    }; break;


            case 2:
                    for(j=0;j<8;j++){
                        TEMPS_3[j]=TEMPS[j];
                    }; break;
            case 3:
                    for(j=0;j<8;j++){
                        TEMPS_4[j]=TEMPS[j];
                    }; break;
            default: break;
        }

    }//End For; Final de Lectura de temperatura
}
else{

    Lcd_Cmd(_LCD_CLEAR);
    Lcd_Out(2,4,"Sin Sensores");
    Lcd_Out(3,5,"Conectados");
```

Continuación del apéndice.

```
        Delay_ms(1000);
        Lcd_Cmd(_LCD_CLEAR);

    }
    //Lectura de iButton
    if(ROW_RESET(&PORTC)==0){                    //Reset; Si hay un iButton intentando identificarse

        do{
            ROW_Write_Byte(&PORTC,0x33);              //Comando Read ROM

            for(j=0;j<8;j++){

                Codigo_Rom_Unico[j]=ROW_Read_Byte(&PORTC);

            }

            ROW_RESET(&PORTC);

            for(j=0;j<8;j++){
                docrc8(Codigo_Rom_Unico[j]);
            }

        }while(crc8 != 0); //Hasta que el codigo ROM sea el correcto


if((EEPROM_Read(0x00)==Codigo_Rom_Unico[0])&&(EEPROM_Read(0x01)==Codigo_Rom_Unico[1])&&(EEP
ROM_Read(0x02)==Codigo_Rom_Unico[2])&&(EEPROM_Read(0x03)==Codigo_Rom_Unico[3])&&(EEPROM_
Read(0x04)==Codigo_Rom_Unico[4])&&(EEPROM_Read(0x05)==Codigo_Rom_Unico[5])&&(EEPROM_Read(
0x06)==Codigo_Rom_Unico[6])&&(EEPROM_Read(0x07)==Codigo_Rom_Unico[7])){

            LATC.B1=1;

            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(2,7,"Acceso");
            Lcd_Out(3,5,"Autorizado");
            Delay_ms(1000);
            Lcd_Cmd(_LCD_CLEAR);

            T0CON.TMR0ON=1;          //Stop Timer 0;
        }
        else{
```

Continuación del apéndice.

```
                    Lcd_Cmd(_LCD_CLEAR);
                    Lcd_Out(2,7,"Acceso");
                    Lcd_Out(3,3,"No Autorizado");
                    Delay_ms(1000);
                    Lcd_Cmd(_LCD_CLEAR);
                  }
                }
                if(PORTC.B2==0){  //Si esta abierta la puerta
                    LATA.B1=0;
                }
                else{
                    LATA.B1=1;
                }
              }
}    //END
```

Fuente: elaboración propia.