TRABAJO ESPECIAL DE GRADO

DISEÑO DE UN EQUIPO PARA HACER AUTÓNOMO EL SISTEMA DE CONTROL DE ENCENDIDO Y APAGADO DE CIRCUITOS ELÉCTRICOS EN EDIFICACIONES

Presentado ante la Ilustre
Universidad Central de Venezuela
Por el Br. Kenyery S. Jonathan
Para optar al Título
De Ingeniero Electricista

TRABAJO ESPECIAL DE GRADO

DISEÑO DE UN EQUIPO PARA HACER AUTÓNOMO EL SISTEMA DE CONTROL DE ENCENDIDO Y APAGADO DE CIRCUITOS ELÉCTRICOS EN EDIFICACIONES

TUTOR ACADEMICO: Prof. José Alonso

Presentado ante la Ilustre
Universidad Central de Venezuela
Por el Br. Kenyery S. Jonathan
Para optar al Título
De Ingeniero Electricista

CONSTANCIA DE APROBACIÓN

Caracas, 01 de noviembre de 2017

Los abajo firmantes, miembros del Jurado designado por el Consejo de Escuela de Ingeniería Eléctrica, para evaluar el Trabajo Especial de Grado presentado por el Br. Jonathan Kenyery S., titulado:

"DISEÑO DE UN EQUIPO PARA HACER AUTÓNOMO EL SISTEMA DE CONTROL DE ENCENDIDO Y APAGADO DE CIRCUITOS ELÉCTRICOS EN EDIFICACIONES"

Consideran que el mismo cumple con los requisitos exigidos por el plan de estudios conducente al Título de Ingeniero Electricista en la opción de Electrónica y Control, y sin que ello signifique que se hacen solidarios con las ideas expuestas por el autor, lo declaran APROBADO.

Prof. Ebert Brea
Jurado

Prof. Jesús Sánchez
Jurado

Prof. José Alonso
Tutor Académico

A mi mamá y a mi papá

RECONOCIMIENTOS Y AGRADECIMIENTOS

A la Universidad Central de Venezuela y a la Escuela de Ingeniería Eléctrica, mi casa de estudios durante mis años de carrera.

Agradezco a mi profesor y tutor académico José Alonso, por el apoyo y por facilitar los componentes para el desarrollo de este proyecto.

A mis padres y familia. Por su apoyo incondicional a lo largo de mi vida y mi carrera, sin ellos no estaría donde estoy hoy.

A mis compañeros de estudio. Quienes pasaron a formar parte de mi vida, y con quienes compartí incontables horas resolviendo ejercicios y redactando aburridos informes de laboratorio. Gracias a Freddy, Cecilio, Carolina, Rosa y Santos por todo su apoyo durante estos años.

A todas aquellas personas que conocí durante mi vida universitaria, cuya lista de nombres es muy larga para mencionarlos a todos. Gracias por todos los momentos que pasamos juntos los cuales siempre recordare.

Kenyery S. Jonathan

DISEÑO DE UN EQUIPO PARA HACER AUTÓNOMO EL SISTEMA DE CONTROL DE ENCENDIDO Y APAGADO DE CIRCUITOS ELÉCTRICOS EN EDIFICACIONES

Tutor Académico: Prof. José Alonso.

Trabajo especial de grado. Caracas, U.C.V. Facultad de Ingeniería, Escuela de Ingeniería Eléctrica. Año 2017.

Resumen - En presente trabajo especial de grado, se desarrolló un Sistema de Control Autónomo (SCA) para activar circuitos eléctricos según unas tablas horarias. Al dispositivo se le implementó el protocolo Modbus RTU para su interconexión a un sistema SCADA con el propósito de modificar las tablas de una forma sencilla. De este modo el sistema de control horario opera independiente de la comunicación, ya que solo se emplea esta para reprogramar al equipo y no para el control en sí.

El SCA consta de un microcontrolador encargado de verificar la hora de un reloj en tiempo real y compárala con la información de las tablas horarias almacenadas en una memoria, y así determinar que circuito debe activar.

Además, se implementó el prototipo utilizando el hardware seleccionado, y se realizaron pruebas para validar el concepto. Satisfactoriamente, la remota diseñada fue capaz de cumplir los horarios programados, puede operar sin conexión a la red SCADA y la información de las tablas horarias sobrevive a una falla de alimentación.

Palabras Claves: Automatización, Microcontroladores, Control Horario, Software, SCADA, Modbus, Domótica.

INDICE

LISTA DE TABLAS

LISTA DE FIGURAS

INTRODUCCION

Gracias a la electrónica, el sector industrial fue capaz de automatizar la mayoría de sus procesos, optimizando recursos y garantizando el continuo monitoreo de sus líneas de producción. Pero la expansión de la automatización no se detuvo en el sector industrial, sino que también ganó terreno en el uso cotidiano, ya que proporciona mayor comodidad a los usuarios. Esta automatización de edificaciones no industriales es lo que se conoce como Domótica.

En la actualidad la domótica y la automatización doméstica se fundamenta en las redes de comunicación, principalmente el internet, ya que permite integrar una gran cantidad de servicios de una forma amigable y cómoda. Teniendo en cuenta que los sistemas de comunicación pueden fallar, es necesario plantear dispositivos de automatización que puedan operaren caso de que las comunicaciones no se encuentren disponibles

Por esto se plantea diseñar un dispositivo de control horario para manejar circuitos eléctricos en edificaciones de forma autónoma. Este dispositivo debe almacenar la información de una tabla horaria en su memoria, así las tendrá disponible siempre que las necesite. Además, debe disponer de algún tipo de comunicación mediante redes externas para poder reprogramar los horarios de forma remota. De este modo el dispositivo puede formar parte de un sistema de automatización mayor.

Una vez planteado la problemática, se expondrán conceptos y herramientas que son necesarias para el entendimiento del desarrollo del proyecto. Luego Se ofrecerá una descripción del diseño de tablas horarias, mecanismos de comunicación y el algoritmo de funcionamiento del dispositivo. Finalmente se realizarán pruebas del concepto planteado y se expondrán los resultado y conclusiones finales.

CAPITULO I

FUNDAMENTOS DE LA INVESTIGACION

PLANTEAMIENTO DEL PROBLEMA

En la actualidad algunas edificaciones disponen de sistemas de control centralizado encargados del manejo automático de circuitos eléctricos (como alumbrado, aires acondicionado, sistemas de seguridad, etc.). Generalmente el dispositivo encargado de realizar el control es un computador basado en un sistema SCADA conectados a internet, de modo que una sola central es capaz de supervisar y controlar una gran cantidad de edificaciones, incluso si no están físicamente en el mismo lugar. Dichos sistemas incrementan el nivel de confort de los usuarios y optimizan el uso de recursos, al establecer los horarios de operación de los circuitos.

Sin embargo, estos sistemas de control pueden presentar fallas, ya sea porque algún programa dejo de funcionar correctamente o simplemente por pérdida de conexión. Esto deja los servicios inoperativos con la salvedad de que puedan operarse de forma manual. Tomando esto en cuenta, resulta importante plantear una solución que garantice el continuo funcionamiento del sistema de control horario inclusive si la comunicación con el sistema central desapareciera.

Para solucionar el problema se propone utilizar un microcontrolador, como solución económica, para diseñar un prototipo capaz de obtener la información de las tablas horarias desde el sistema SCADA, almacenarlas en una memoria y realizar el control horario sobre los circuitos eléctricos siguiendo la información adquirida. De este modo se tendrá un dispositivo que puede ser programado a distancia y no dependerá de la conexión con el SCADA para el control en sí.

OBJETIVOS

General

Diseñar un prototipo, para hacer autónomo el sistema de control de encendido y apagado de circuitos eléctricos de edificaciones, empleando un micro controlador.

Específicos

- Definir el diagrama de bloques funcional del prototipo y selecciona los elementos requeridos para la elaboración del proyecto.
- Definir los protocolos de comunicación SCADA necesarios para establecer la comunicación con el prototipo.
- Realizar el programa del microcontrolador mediante el software seleccionado.
- Implementar comunicación con el sistema SCADA y el RTC.
- Realizar el montaje del elemento circuírteles necesarios para el funcionamiento del equipo.
- Realizar pruebas de funcionamiento.

ANTECENDENTES

El trabajo de pasantía del egresado de la Universidad Simón Bolívar, Carlos Bravo. Quien realizo un sistema de control horario para automatizar la alarma de entrada y salida de los turnos de trabajo de los empleados para la empresa ORINOQUIA. Fabrico un dispositivo que, mediante un panel frontal, permitía fijar los horarios de la alarma evitando así la necesidad de un operador. [1]

El controlador autónomo programable “SuperBrain” marca MONTROL. Es un dispositivo comercial diseñado para controlar y monitorizar sistemas eléctricos y de iluminación. Dispone de 8 salidas digitales, 8 salidas analógicas y 8 entradas universales, interfaz RS485 para comunicarse con redes externas para monitorear el estado de las salidas. Mediante el menú en la pantalla frontal pueden definirse hasta 400 programaciones horarios diferentes. [2]

De la referencia consultada no se encontró un dispositivo que cumpliera con las características necesaria para solventar la problemática. Un dispositivo de control horario, autónomo y reprogramable mediante un sistema SCADA.

METODOLOGIA

El primer paso en el desarrollo del proyecto fue definir las características que debía tener la remota y determinar cuáles elementos serían necesarios para cubriera los requerimientos planteados. Una vez hecho esto, se elaboró un diagrama de bloques funcional de la remota y se seleccionaron los componentes y herramientas necesarios para la desarrollar el prototipo.

Establecido el funcionamiento del equipo y los componentes que necesitaría, se diseñaron las tablas horarias y los registros Modbus que serían necesarios para el funcionamiento y control de la remota. Debido a la extensión de información de las tablas horarias, fue necesario diseñar una forma organizada de representarlas que permitiera el fácil acceso a estas con la menor cantidad posible de registro Modbus.

Una vez elaborado el algoritmo que debía seguir la remota para el control horario y la comunicación con el sistema de control central. Se realizó la programación del microcontrolador y se Implementaron los protocolos de comunicación Modbus e I^2C para comunicar en la remota con el reloj entiempo real, la memoria y el sistema SCADA.

Por último, se construyó una maqueta que sirvió de soporte para el hardware, sobre la cual se realizaron pruebas para validar el funcionamiento del prototipo. Se tomó registro de estas pruebas y se presentaron los resultados en un informe escrito.

CAPITULO II

MARCO TEORICO

LA AUTOMATIZACION

Un sistema automatizado es aquel capaz de realizar diversas operaciones de modo automático o por sí solo. Al comienzo los procesos automatizados consistían en equipos rudimentarios que producían acciones simples y repetitivas que facilitaran alguna tarea, como por ejemplo algún resorte o un flujo canalizado de agua/vapor. A mediados del siglo XX con la evolución de las computadoras digitales la automatización alcanzo nuevos niveles, ya que la flexibilidad de estas permitió automatizar cualquier tipo de tareas simples, repetitivas o especializadas. Hoy en día muchas industrias basan gran cantidad de sus fases de producción en distintos tipos de elementos automatizados.

Sin embargo, los avances en computación no solo han beneficiado al sector industrial en términos de automatización, esta se ha expandido a otro tipo de edificaciones como viviendas, centros empresariales, edificios administrativos, sucursales bancarias, entre otros.

El crecimiento de la automatización al sector doméstico tomó el nombre “Domótica”, palabra formada por los términos *Domus* (que en latín significa casa) y *Tica* (de automática). la Domótica consiste en proporcionar diversos servicios mediante sistemas tecnológicos intercomunicados, con el fin de cubrir las necesidades de los habitantes y aumentar el confort de las infraestructuras.

Entre las principales áreas de servicios que abarca la domótica podemos encontrar:

- **Control:** abarca el manejo automático (encendido, apagado, abrir, cerrar, regular) de aplicaciones y dispositivos domésticos, como sistemas de iluminación, climatización, cerraduras, etc.
- **Seguridad:** abarca alarmas de personas, bienes y averías, como alarmas para intrusos, cámaras de vigilancia, alarmas técnicas (incendio, humo, fuga de gas, fuga de agua).
- **Voz y data:** abarca la distribución de textos, imágenes y sonidos, como acceso a internet y servicio telefónico.
- **Audio y video:** abarca servicios de entretenimiento e información, como radio y televisión.

Con esta definición, queda claro que la domótica no es “Servicio” ni “Producto”, sino la integración y la implementación de los sistemas electrónicos en el inmueble.

LOS SISTEMAS EMBEBIDOS

Un sistema embebido es conjunto de componentes electrónicos diseñado para realizar alguna función específica en tiempo real, de modo que cubra alguna necesidad o solucione un problema. A diferencia de las computadoras, las culés tiene un propósito general, el sistema embebido solo puede realizas unas pocas tareas, lo que hace que sean de bajo costo y de poco consumo de potencia. En la actualidad muchos dispositivos cotidianos emplean sistemas embebidos en su funcionamiento, como por ejemplo el horno microondas, ascensores, radios, sistemas de automatización, entre muchos otros.

Los sistemas embebidos están conformados por una unidad de procesamiento y por componentes periféricos (memoria, fuentes de alimentación, módulos de comunicación, etc.) los cuales varían según la aplicación. La unidad de procesamiento,

generalmente un microcontrolador, representa el cerebro del equipo y aporta la capacidad de cómputo al sistema, cierta inteligencia mediante un *software* y la capacidad de acción sobre los componentes periféricos a través de una interfaz de entada/salida.

En la actualidad los sistemas embebidos están evolucionando a sistemas "inteligentes", donde la principal característica que debe tener para considerarlos inteligentes es que estos sean capaces de comunicarse con otros dispositivos o en especial a internet. Como la comunicación a adquirido gran importancia, muchos de estos sistemas ya incorporan interfaces estándar de comunicación por cable o inalámbricas, como por ejemplo RS-232, RS-485, I^2C, USB, Wi-Fi, GSM, etc.

LOS MICROCONTROLADORES

El nombre de controlador se le otorga al dispositivo que se emplea para gobernar uno o varios procesos. Aunque el concepto de controlador no ha variado a través del tiempo, su implementación física ha sufrido varias modificaciones, des los dispositivos de control mecánicos hasta los de control electrónico.

Los primeros controladores electrónicos estaban compuestos por componentes de lógica discreta, posteriormente se emplearon los microprocesadores o microcomputadoras, los cuales se rodeaban de chips de memoria y periféricos de entrada y salida, de esta forma mediante un código de programa el microprocesador podía ejecutar órdenes. En la actualidad, todos los elementos discretos que se requerían para el control, se han podido incluir en un único chip el cual recibe el nombre de microcontrolador.

El microcontrolador, es entonces, un circuito integrado programable, capaz de ejecutar las instrucciones grabadas en su memoria de forma secuencial, por lo que se puede referir al microcontrolador como una microcomputadora completa encapsulada en un circuito integrado.

Todo microcontrolador está compuesto de elementos esenciales para su funcionamiento y de elementos periféricos que varían según la aplicación. Entre los elementos esenciales se encuentran:

- La unidad de procesamiento central (CPU): compuesta por el microprocesador, y le otorga capacidad de computo al controlador.
- Unidad de memoria: se divide en dos, la memoria de programa (ROM) donde se guarda las instrucciones que debe seguir el microcontrolador, y la memoria temporal (RAM) donde se almacena información de algún calculo, esta última se pierde al reiniciar el controlador.
- Líneas de entrada y salida (E/S): son los periféricos que permite al microcontrolador comunicarse e interactuar con el resto de los componentes periféricos u otros microcontroladores.

Los elementos periféricos son circuitos, también integrados en el chip del controlador, que agregan funciones adicionales a los microcontroladores. Entre los más comunes encontramos convertidores analógico-digital y digital-analógico, osciladores, temporizadores, relojes en tiempo real, instrumentos de comunicación serial como el UART y buses de interfaz serie como el I^2C o SPI. Se ha convertido en estándar, que los elementos mencionados se encuentren integrados hasta en los microcontroladores más simples.

Interrupciones en el microcontrolador

Como el hardware ya viene integrado en un solo chip, para usar un microcontrolador se debe especificar su funcionamiento por software a través de programas que indiquen las instrucciones que el microcontrolador debe realizar, estas instrucciones se ejecutan secuencialmente y solo una a la vez.

Las interrupciones es un recurso del cual disponen los microcontroladores de hoy en día, el cual permite cambiar la continuidad de ejecución de las instrucciones para atender una necesidad. Esta técnica coloca el programa principal temporalmente en suspensión mientras el controlador ejecuta otro conjunto de instrucciones en respuesta a un evento, sea interno o externo. Al terminar la interrupción el programa principal continua donde fue interrumpido, creando así la sensación de que se ejecutan instrucciones en paralelo.

COMUNICACIÓN ENTRE EQUIPOS

La base de la automatización domestica radica en que los equipos electrónicos encargados de realizar alguna función dentro de una edificación, puedan comunicarse entre ellos e intercambiar información. Pero para lograr esto, es necesario dotar a esos equipos con dispositivos que les permitan enviar y recibir información, protocolos para que puedan entenderse entre ellos y normas que establezcan estándares. Con esto dicho, se presentan a continuación algunos de los elementos que se emplean para establecer el intercambio de datos entre dos o más dispositivos.

UART

UART, corresponde a las siglas en inglés para “Universal Asynchronous Receiver-Transmitter”, es un dispositivo de comunicación que se encarga de convertir los datos en forma paralelo a serie, y viceversa, para que puedan ser trasmitidos a través de un Bus de comunicación. La trasmisión en serie es más efectiva en términos de costo, pero más lenta en comparación a la comunicación en paralelo (a través de múltiples cables).

El UART toma un Bytes de información y envía los Bits individualmente de forma secuencial, en el destino, un segundo UART hacer el proceso inverso y reensambla el Byte. Tanto el trasmisor y el receptor deben ajustarse a la misma velocidad baudios y el tamaño de la trama debe predefinirse para lograr el intercambio correcto de información.

La trama de datos se compone de un Bit de inicio, los Bits de datos (generalmente 8) y un bit de paridad (en algunos casos se usan 2 bits de parada). La línea de comunicación se mantiene en “1” lógico para indicar que canal esta libre. El trasmisor coloca la línea en “0” lógico para indicar al receptor que una trama está por llegar, los próximos 8 Bits representan la información a transmitir, si se utiliza el Bit de paridad este se coloca al final de todos los datos. Por último, el bit de parada coloca la línea en “1” lógico de nuevo.

En los microcontroladores, el UART se encuentra dentro del integrado y es el dispositivo principal de comunicación de estos. Usualmente se utiliza otro dispositivo de interfaz (transceptor) para convertir las señales de nivel lógico del UART hacia y desde los niveles de señales externos, como por ejemplo los estándares para señalización por voltaje RS-232, RS-422 o RS-485.

I2C

I²C, del inglés Inter-Integrated Circuit (o circuito inter integrado), es un Bus de comunicación serie, el cual se utiliza principalmente para la comunicación interna entre distintos dispositivos de un circuito, como, por ejemplo, un controlador, memorias EEPROM, convertidores analógico-digital, termómetros, etc. La comunicación entre los elementos del bus está basada en el modelo maestro-esclavo.

La trasmisión de datos por el bus I²C se realiza a través de dos líneas binarias de comunicación bidireccional. Por la línea SDA (Serial Data) se trasmiten los bits de información, mientras en la línea SCL (Serial Clock) se genera una señal de reloj para sincronizar los datos entre dispositivos [3]. En la figura 1 se muestra un ejemplo de conexión entre un dispositivo maestro y varios esclavos. R_p representa una resistencia de Pull-up.

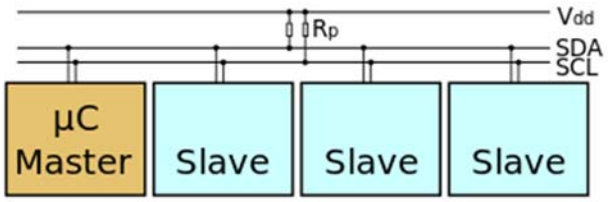


Figura 1 Esquema de conexión del bus I²C

Siempre que el Bus este libre el maestro puede realizar operaciones de escritura o lectura sobre algún esclavo. La comunicación comienza con una señal de inicia seguida, seguida de los Bytes de información y termina con una señal de parada. La figura 2 muestra el comportamiento de las señales en ambas líneas durante la transferencia de datos.

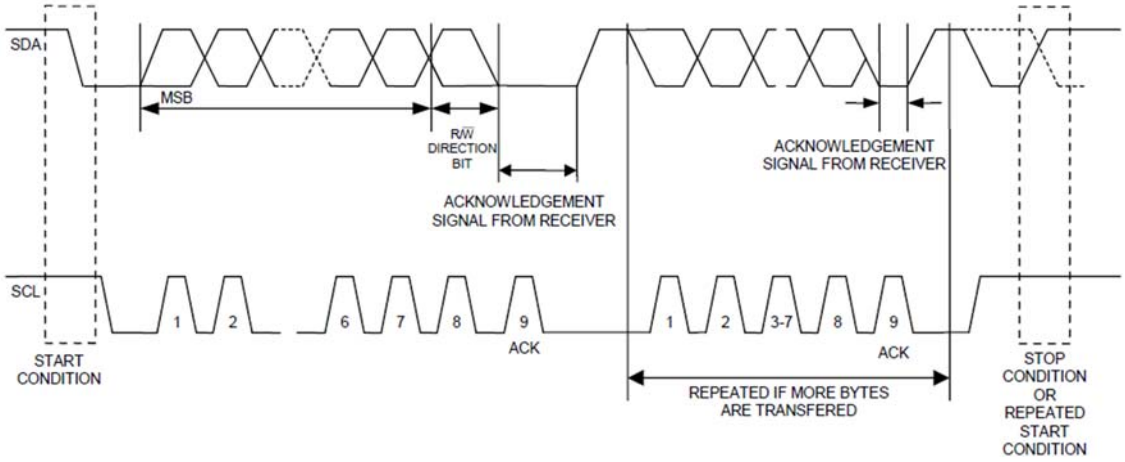


Figura 2 Señales de las líneas SDA y SCL durante la trasferencia de datos

El Bus de comunicación se encuentra libre si las ambas líneas se mantienen en el nivel lógico alto. Para iniciar la comunicación el maestro genera la señal de inicio colocando un flanco descendente en SDA mientras SCL se encuentra en nivel alto. Los Bits trasmitidos solo se consideran validos si su nivel lógico no cambia durante una fase de reloj alta [3]. Para finalizar la transición el maestro genera la señal de parada colocando un flanco ascendente en SDA mientras SCL está en nivel alto, de forma alternativa puede enviarse una señal de reinicio si se desea trasmitir otros datos inmediatamente después, esta se comporta de igual manera a la de inicio.

Durante la trasmisión de datos, el receptor debe generar un Bit de confirmación (ACK, Acknowledge) al final de cada unidad de datos (8 Bits o 1 Byte), que indica al trasmisor que los datos fueron recibidos, esta señal se genera cuando el receptor mantiene en el nivel bajo SDA durante el noveno periodo en alta del ciclo de reloj. En caso de no reconocer la información, el Bit seria de no confirmación (NACK, Noacknowlegde).

Actualmente, los microcontroladores poseen *hardware* dedicado a la comunicación por I^2C lo que facilita implantar este tipo de comunicación, ya que este se encarga de

realizar el manejo de las líneas SDA y SCL según sea necesario, mediante el uso de comandos por *software*.

El Bus I^2C emplea un protocolo de comunicación el cual establece como deben conformarse las tramas Bytes de modo que ambos dispositivos puedan entenderse. Dependiendo de si el maestro requiere leer o escribir información la trama varia varían, sin embargo, el primer byte siempre está compuesto de la siguiente manera: los primeros 7 Bits representa la dirección del esclavo al que se le envía el mensaje y el octavo Bit (R/W-Bit) es el que indica que función realizar: recibir datos (nivel lógico bajo), o enviar datos al maestro (nivel lógico alto). El segundo Byte indica cual es la información que se desea y de ultimo se tantos Bytes como sea necesarios con la información del mensaje. A continuación, se muestra cómo deben armarse las tramas de lectura y escritura.

Figura 3 Trama de escritura para el I^2C

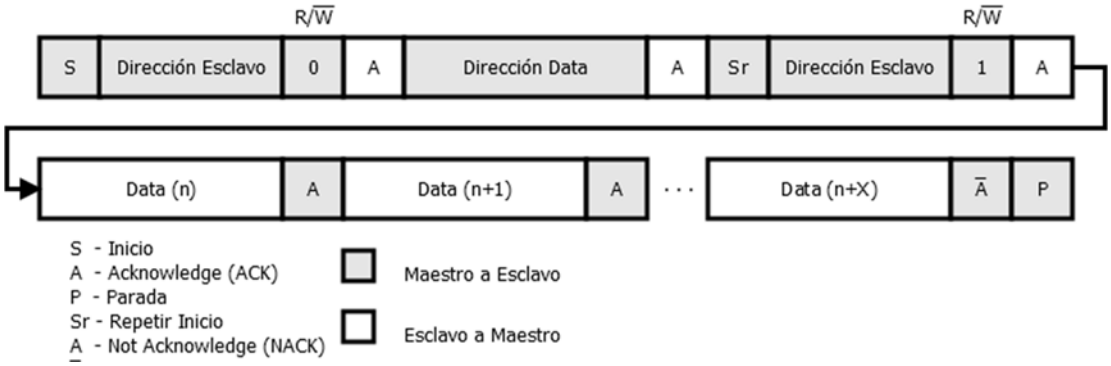


Figura 4 Trama de lectura para I²C

La figura 3 muestra la trama la trama de escritura. Siguiendo el protocolo, el maestro empieza la comunicación con la señal de inicio, inmediatamente envía el primer byte compuesto de la dirección del esclavo más el R/W-Bit en cero (nivel lógico bajo) para indicarle al esclavo que recibirá información. Espera la confirmación del esclavo y envía el segundo byte el cual corresponde a la posición de moría en el esclavo donde debe empezar a escribir la información. El maestro espera la confinación de nuevo y envía todos los Bytes que sean necesarios, con cada byte recibido el esclavo debe confirmar que recibió el Bytes para que el maestro pueda enviar el próximo. Cuando se hallan transmitido todos los datos, el maestro realiza la señal de parada y se termina la comunicación.

La figura 4 muestra la trama de lectura. Los dos primeros Bytes enviados corresponden al comando de escritura, esto se hace para que el esclavo se coloque en una posición de memoria específica, pero no se envía ningún dato, de no hacer esto y enviar solo el comando de lectura la información de respuesta corresponder a la última posición de memoria acezada. Luego de la confirmación del esclavo, el maestro realiza una seña de reinicio e inmediatamente vuelve a enviar la dirección del mismo esclavo, pero esta vez con el Bit-R/W en uno (nivel lógico alto) para indicar al esclavo que debe enviar información. El esclavo confirma y empieza a trasmitir los Bytes de información desde la posición de memoria establecida anteriormente, el

maestro enviar el Bit de confirmación tras cada byte recibido. Cuando el maestro decida que ya recibió todos los Bytes que requería, genera el Bit de no confirmado (NACK) para que el esclavo se detenga y deje el control de las líneas de comunicación al maestro para que este genere la señal de parada para y así desocupar el Bus.

Modbus

Modbus es un protocolo de comunicación que define una estructura de mensajes para ser empleada en una comunicación de tipo maestro-esclavo, donde el maestro siempre inicia la comunicación y los esclavos responde según se le ordenen. Ya que Modbus solo establece como deben ser construido los mensajes para la comunicación, el protocolo es independiente del medio físico por el cual se trasmite los datos, aunque normalmente se implementa usando los estándares RS-232, RS-422 o RS-485. [4]

Los dispositivos que se comunican utilizando este protocolo pueden ser configurados para trasmitir los datos en dos modalidades: RTU (del inglés Remote Terminal Unit) o ASCII (American Standard Code for Information Interchange). Ambas implementaciones del protocolo son serie y la estructura de las tramas son iguales, la diferencia radica en la representación numérica de los datos, *Modbus RTU* emplea una representación binaria compacta de la información, mientras que *Modbus ASCII* usa una representación legible del protocolo mediante caracteres ASCII, esta última es menos eficiente ya que no permite la misma densidad de datos que el modo RTU a la misma velocidad de trasmisión. [4] Por temas de eficiencia el modo RTU es el más empleado al implantar una comunicación con este protocolo además de ser que se utilizó para este proyecto, por lo tanto, solo se explicará detalladamente como se componen las tramas de datos para el modo RTU.

Cada trama o mensaje está compuesto por una cadena de Bytes, cada Byte contiene dos caracteres hexadecimales de cuatro bits (0 a F).

Inicio	Dirección Esclavo	Función	Data	CRC	Fin
3,5 Char time	8 Bits	8 Bits	N * 8 Bits	16 Bits	3,5 Char time

Figura 5 Trama de datos para Modbus RTU

Como se muestra en la figura 5, en modo RTU, los mensajes comienzan con un intervalo de silencio de al menos 3,5 caracteres. Esto se implementa como un múltiplo del tiempo que toma enviar un carácter (8 Bits) a la velocidad de baudios que se utiliza para la comunicación. Si un nuevo mensaje llegara comenzar antes de que trascurra el tiempo de los 3,5 caracteres, el dispositivo receptor asumirá que el nuevo dato es continuación del mensaje anterior, lo que generará un error, ya que el valor en el campo CRC final no será válido para los mensajes combinados.

Después del tiempo de inicio, el primer campo trasmitido corresponde a la dirección del esclavo a la cual va dirigida el mensaje. Seguido del campo de *Función,* el cual le indica a ese esclavo que acción debe realizar, ya sea escribir un dato, enviar información de vuelta, activar alguna salida, hacer diagnóstico, etc. La figura 6 muestra una lista de funciones Modbus, las resaltadas son las más comunes en cualquier implementación.

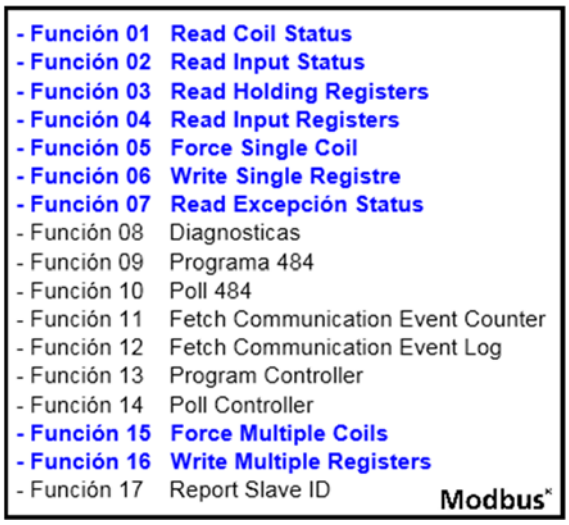


Figura 6 Lista de funciones Modbus

El campo de *Data* contiene la información necesaria para realizar la acción requerida y su longitud varía según la función, por ejemplo, si se requiere leer un dato en este campo se indica cual dato. El ultimo campo de la trama es el campo de verificación de error, este utiliza el código de verificación por redundancia cíclica o CRC (Cyclic Redundancy Check) de este modo se evalúa la integridad del mensaje.

Los esclavos conectados al Bus de comunicación monitorean la red constantemente, incluso en los intervalos de silencio, cuando el primer byte es trasmitido por el maestro (el byte de dirección) cada esclavo lo reconoce y determina si el mensaje está dirigido a él. Si el mensaje es recibido sin errores y el esclavo es capaz de procesar la información, este devuelve un mensaje al maestro el cual consiste en un eco del mensaje recibido, es decir, los capos de la trama son idénticos al mensaje original. El único caso en donde la respuesta no es exactamente el eco de la consulta son en las funciones en donde el maestro solicita información al esclavo (como por ejemplo la función 3), en estos casos en el campo de *Data* se encontrará la información solicitada y el resto de los campos de la trama permanecerán inalterados. La tabla 1 muestra un ejemplo de una trama de consulta y su respectiva respuesta para la función 3, donde

el maestro solicita información el contenido de 2 registros a un esclavo. Los registros corresponden a las posiciones de memoria en donde los dispositivos almacenas información relevante para algún proceso.

Tabla 1 Ejemplo de la trama para la función 3

Consulta		**Respuesta**	
Nombre del campo	**Ejemplo RTU (Hex)**	**Nombre del campo**	**Ejemplo RTU (Hex)**
Dirección esclavo	0x01	Dirección esclavo	0x01
Función	0x03	Función	0x03
Dirección registro HI	0x00	Cantidad de Bytes	0x04
Dirección registro LO	0x00	Data 1 HI	0x00
Número de registros HI	0x00	Data 1 LO	0x06
Número de registros LO	0x02	Data 2 HI	0x00
CRC LO	0xC4	Data 2 LO	0x05
CRC HI	0x0B	CRC LO	0xDA
		CRC HI	0x31

En la trama de consulta, El campo de *DATA* contiene el registro inicial y la cantidad de registros a leer, en este caso las direcciones de estos son de 16 Bits. Para enviar un numero de 16 Bits a través del protocolo Modbus es necesario dividirlo en dos numero de 8 Bits y enviarlos por separado, de igual manera ocurre con el código de confinación (CRC). El esclavo responde a la consulta, y en el campo de *DATA* coloca la cantidad de Bytes que contiene la información solicitada, seguido de la información de los registros.

En la tabla 2 y la tabla 3 se muestran ejemplos de las tramas para las funciones 5 y 6 respectivamente. La respuesta del esclavo siempre es un eco de la consulta realizada por el maestro. [4]

Tabla 2 Ejemplo de la trama para la función 5

Consulta		Respuesta	
Nombre del campo	**Ejemplo RTU (Hex)**	**Nombre del campo**	**Ejemplo RTU (Hex)**
Dirección esclavo	0x11	Dirección esclavo	0x11
Función	0x05	Función	0x05
Dirección de bobina HI	0x00	Dirección de bobina HI	0x00
Dirección de bobina LO	0xAC	Dirección de bobina LO	0xAC
Estado ON/OFF HI	0xFF	Estado ON/OFF HI	0xFF
Estado ON/OFF LO	0x00	Estado ON/OFF LO	0x00
CRC LO	0x4E	CRC LO	0x4E
CRC HI	0x8B	CRC HI	0x8B

Tabla 3 Ejemplo de la trama para la función 6

Consulta		Respuesta	
Nombre del campo	**Ejemplo RTU (Hex)**	**Nombre del campo**	**Ejemplo RTU (Hex)**
Dirección esclavo	0x11	Dirección esclavo	0x11
Función	0x06	Función	0x06
Dirección registro HI	0x00	Dirección registro HI	0x00
Dirección registro LO	0x01	Dirección registro LO	0x01
Data a escribir HI	0x00	Data a escribir HI	0x00
Data a escribir LO	0x03	Data a escribir LO	0x03
CRC LO	0x9A	CRC LO	0x9A
CRC HI	0x9B	CRC HI	0x9B

SCADA

SCADA, es el acrónimo en inglés para Supervisión, Control y Adquisición de Datos (Supervisory Control And Data Acquisition). Es un concepto que se emplea para crear un *software* computacional que permita controlar, supervisar y gestionar algún proceso. Este se origina con el fin de manejar procesos industriales, ya que facilitaba la retroalimentación en tiempo real con los dispositivos de campo, pero al ser parte de los sistemas automatización este se extendió a instalaciones de cualquier tipo.

Un sistema SCADA requiere de una interfaz hombre-máquina (HMI), Controladores, redes de comunicación, bases de datos y un software. Lo que diferencia a este de un

sistema de control distribuido es que el lazo de control generalmente es cerrado por el operador. Aunque, dependiendo de la complejidad y velocidad del proceso, se requiera de un sistema de control automático y se emplearía un sistema SCADA para monitorear y fijar los puntos de consigna (*setpoin*t). En la figura 7 se muestra un ejemplo de un proceso con sistema SCADA.

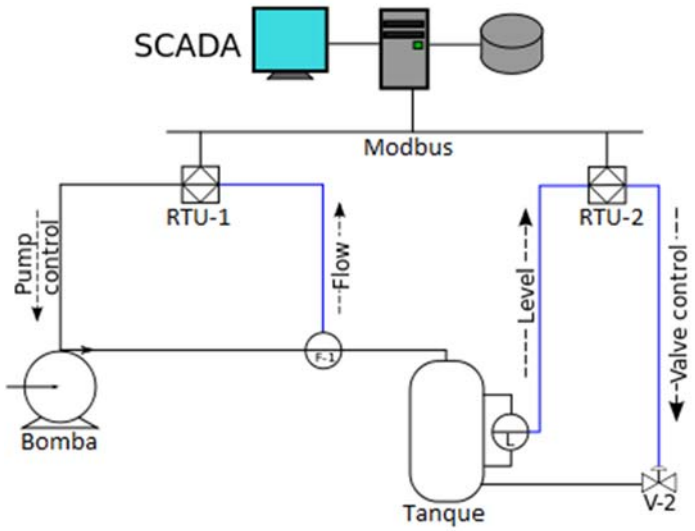


Figura 7 Proceso con sistema SCADA

En el ejemplo, el proceso es controlado de forma automática ya que la realimentación del lazo de control es cerrada a través del RTU los cuales ajustan el flujo de la bomba y el nivel del tanque. A su vez, estos se comunican mediante una red Modbus a un sistema SCADA el cual supervisa el desempeño del proceso y permitiría ajustar el proceso. El SCADA también puede mostrar gráficos históricos, tendencias, alarmas y eventos entre otras funciones, que sirvan de ayuda a los operadores del proceso.

RS - 485

Es un estándar de comunicación serie que define las características eléctricas para la interconexión de dispositivos. Está compuesto por un Bus de comunicación basado en un sistema Diferencial Half-Duplex para la trasmisión de datos.

Un Bus diferencial está compuesto de dos hilos (A y B) y los datos se representan mediante la diferencia de tensión que existe entre ellos, si "A" es más positivo que "B" se considera como un 1 (uno) lógico y el caso inverso un 0 (cero) lógico. Sin embargo, la tensión diferencial entre A y B debe ser al menos de ±0.2V para que se reconozca un valor. En la figura 8 se muestra la topología de conexión multipunto de distintos dispositivos.

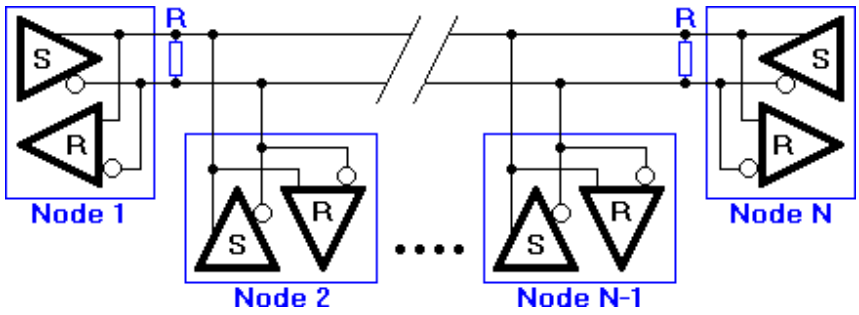


Figura 8 Topología de conexión multipunto Half-Duplex

Half-Duplex quiere decir que cada equipo conectado al par de hilos pude enviar o recibir información, pero no de forma simultánea. Por lo tanto, en esta topología de conexión se requiere de un protocolo de comunicación que permita la interacción entre dispositivos, por ejemplo, el protocolo Modbus.

El estándar RS-485 puede utilizarse en una topología Full-Dúplex (Transmisión y recepción simultanea) si se duplica la cantidad de hilos, dedicando 2 la transmisión y otros 2 a la recepción.

CAPITULO III

DESARROLLO DE PROYECTO

REQUERIMIENTOS DEL SISTEMA Y DISEÑO

El prototipo planteado para este proyecto es una unidad remota de automatización capaz de comunicarse con un sistema SCADA usando el protocolo Modbus RTU. La remota debe ser capaz de controlar el encendido y apagado de una cierta cantidad de circuitos eléctricos, siguiendo una tabla de horarios previamente definida. Además, el funcionamiento horario debe ser independiente de la comunicación con sistema de control centralizado, de modo que solo se empleará el SCADA para fijar los horarios, y una vez definidos estos, la remota debe operar inclusive si la conexión con el sistema central se pierde.

Para elaborar el proyecto fue necesario definir qué tipo de componentes se requerían para cubrieran las necesidades planteadas anteriormente y de qué forma se deben integran entre ellos para crear el sistema embebido que resuelva esta problemática:

- El componente principal para este diseño es el microcontrolador, instrumento fundamental para la automatización que conforma la unidad de lógica y cerebro de la remota.
- Para comunicar el microcontrolador al sistema SCADA, es necesario que este pueda intercambiar información utilizando uno estándares físicos de comunicación, para ello se requiere de un transceptor que convirtiera la información de los niveles de tensión del UART al estándar físico de comunicación y viceversa.
- La información de los horarios recibida desde el sistema SCADA debe ser almacena en la remota para que el control horario pueda seguir operando inclusive si la comunicación con el sistema central se pierde. Para ello se

incorporó una memoria externa de tipo EEPROM compatible con el bus de comunicación I^2C, en la cual se almacenará toda la información que deba perdurar en caso de una falla en la alimentación eléctrica de la remota.

- Para poder realizar el control horario, la remota debe estar dotada de un reloj programable que indique la hora y fecha exacta en todo momento, de esta forma podrá comparar con la información de las tablas horarias y decidir cómo debe operar. Para ello se requiere de un reloj en tiempo real (RTC) externo comunicado siempre mediante el bus I^2C. Este debe disponer de alimentación auxiliar por batería, de modo que el reloj pierda la hora en caso de alguna falla eléctrica en la alimentación de la remota.
- Por último, para que el microcontrolador pueda actuar sobre algún circuito eléctrico de mayor potencia que la de él, se debe empelar un relé por cada circuito a controlar.
- Adicionalmente se incorporará un LED indicador, el cual parpadea al iniciar la remota y mientras se estén transmitiendo datos al sistema SCADA. De esta forma, se tendrá una referencia visual del funcionamiento de la remota.

En la figura 9 se muestra un diagrama de bloques funcional de la remota, con los elementos descritos y como se interconectan entre ellos.

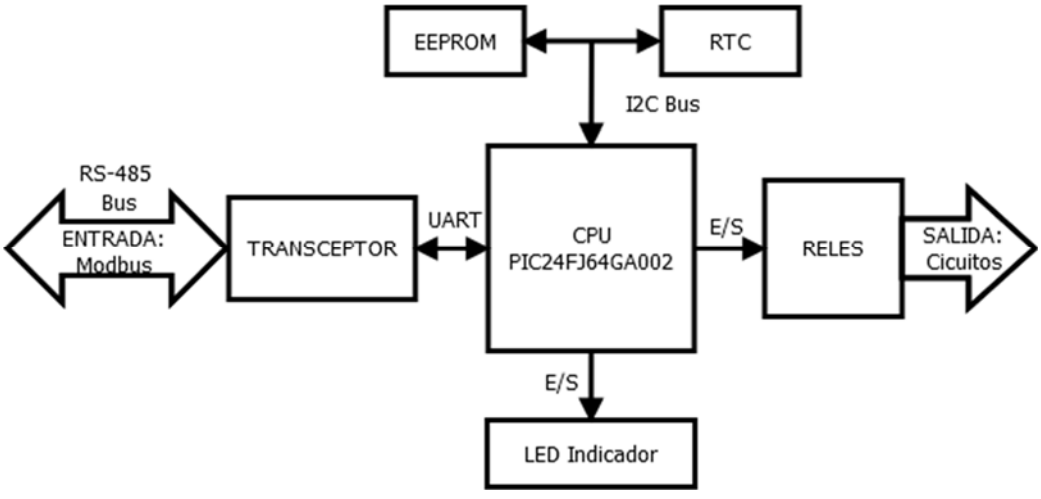


Figura 9 Diagrama de bloques funcional de la remota

Una vez definido el *hardware*, se debe establecer el funcionamiento general de la remota para crear el *software* que guiara las acciones de esta. El micro controlador deberá verificar constantemente la hora del RTC durante el día, y compararla con la tabla de horarios correspondiente a la fecha actual y así decidir cuales circuitos de salida (relés) debe activar. Adicionalmente, debe monitorear el bus de comunicación contantemente en caso de que algún maestro de la red de control centralizado decida acceder a su información, de ser así tomara la acción debida según se le ordene (enviar o almacenar información).

En caso de alguna falla eléctrica, la batería de respaldo del RTC mantendrá en funcionamiento el reloj, de modo que al restaurarse en la alimentación de la remota, se retome el retome el manejo de los horarios sin pérdida de la continuidad del tiempo.

Elementos seleccionados para el proyecto

A continuación, se presentan los componentes seleccionados para la implementación de la remota.

El microcontrolador

En el mercado existe una gran variedad de microcontroladores, lo que dificulta seleccionar el adecuado. A la hora de elegir el microcontrolador se deben evaluar las características de este y compararlas con los requerimientos del proyecto para saber si satisface las necesidades.

Para el proyecto en cuestión, se seleccionó el microcontrolador de uso general marca MCICROCHIO, modelo PIC24FJ64GA004 [5]. La razón principal para seleccionar este microcontrolador fue la disponibilidad de este *hardware* y de los equipos necesarios para su programación (gentilmente suministrados por el tutor académico). Además, el *software* de programación y el compilador para este microcontrolador se

encuentran gratuitamente en internet. Este micro controlador dispone de módulos UART para comunicación serie, módulos dedicados a la interfaz I^2C, oscilador interno, temporizadores, relojes en tiempo real y 21 puertos de Entrada/Salida. Características importantes para el desarrollo del prototipo de remota.

El microcontrolador provisto para este trabajo de grado, se encuentra soldado a una placa de circuitos junto a un regulador de voltaje y los elementos resistivos y capacitivos necesarios para su funcionamiento. Por lo tanto, solo fue necesario agregar una fuente de alimentación de corriente directa. Adicionalmente, la placa de circuito dispone de un LED conectado a uno de los puertos de E/S del microcontrolador, y un transceptor para para adaptar el UART al estándar RS-485. Respecto a los terminales de conexión, la placa dispone de 8 terminales para pines de E/S, un puerto de conexión RS-485 de dos terminales, el puerto de alimentación y 5 pines para la progresión mediante el PICkit. En la figura 10 se muestra la palca de la remota Vemetris.

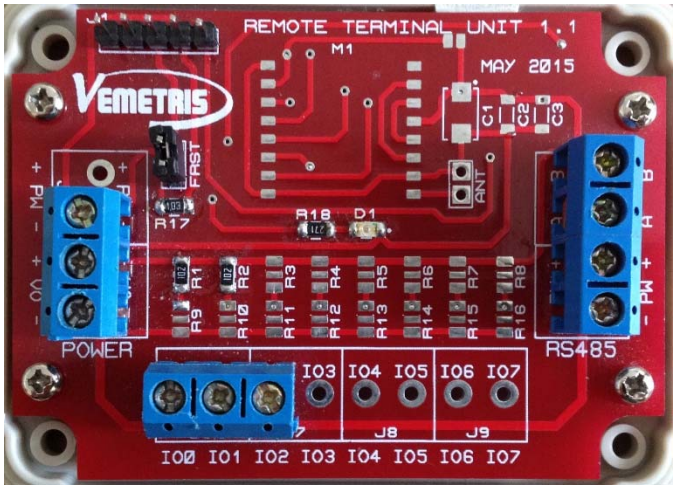


Figura 10 RTU Vemetris v1.1

El *software* utilizado para programar el microcontrolador fue el *MPLAB X IDE* versión 3.35 junto con el compilador XC16 [6], los cuales permiten desarrollar y compilar

código en leguaje C++. Para transferir el código de usuario al microcontrolador se empleó un PICkit 2, el cual es un *hardware* que toma el archivo en hexadecimal del código de usuario y lo transfiere al PIC. Todos programas requeridos se encuentran de forma gratuita en la página web de Microchip.

Reloj en tiempo real (RTC)

Para que los dispositivos de tiempo no pierdan la hora durante una falla eléctrica deben ser alimentados con baterías, de forma que su mecanismo nunca se detenga. Por esta razón se optó en utilizar un reloj externo en vez de emplear el que viene incorporado dentro del microcontrolador, ya que el consumo de corriente de los RTC es aproximadamente mil veces menor que el de un microcontrolador en funcionamiento. De este modo, utilizando una batería pequeña, la remota puede desconectarse de la alimentación por un largo tiempo y no perder la programación de la fecha y hora.

El reloj con interfaz I^2C seleccionado fue el DS3231 [7], este dispositivo mantiene la cuenta de segundos, minutos, horas, día, fecha, mes y año. Además, posee un puerto de alimentación de respaldo por batería y realiza el cambio a esta de modo automático cuando la alimentación principal es cortada. Por lo tanto, cumple con las características necesarias para el proyecto. Sin embargo, tiene una ventaja sobre otros relojes con interfaz I^2C, el DS3231 dispone de un oscilador y un cristal dentro del integrado, Lo que resulta más preciso y menos susceptible a variaciones de tiempo con respecto a los que requieren de un cristal externo.

Memoria

Como se explicará en detalle más adelante, se estimó que se requieren de aproximadamente 3 KBytes para almacenar los horarios de los 7 días de la semana y

alguna otra información de control necesaria para el funcionamiento del equipo. Como el espacio de memoria necesario es relativamente pequeño, se optó por una unidad EEPROM de 32 KBytes con interfaz de comunicación I^2C. El integrado corresponde al código 24LC256 [8].

Los 29 KBytes restantes de la memoria se emplearon para almacenar las tablas horarias de los "días especiales", estos representan días del año en los que el equipo no debe seguir la programación normal sino una programación con un horario especifico, útil para días feriados, días bancarios, cumpleaños, etc.

Transceptor

Por comodidad se utilizó el mismo transceptor que está incorporado a la placa del microcontrolador, ya que cumple con las características necesarias para establecer una comunicación mediante el estándar RS-485. El integrado de este transceptor corresponde al código MAX487. [9]

Relés

Para aislar la parte de poesía de la electrónica, se emplearon relés con acoplamiento óptico. De este modo se protege el microcontrolador y los demás componentes electrónicos en caso de un corto circuito en la alimentación de los circuitos a controlar. Se seleccionó el módulo de relés (comúnmente usado para dispositivos *Arduino*), tal como el que se muestra en la figura 11. Este permite manejar circuitos eléctricos de 125/250 VAC a 10 A.

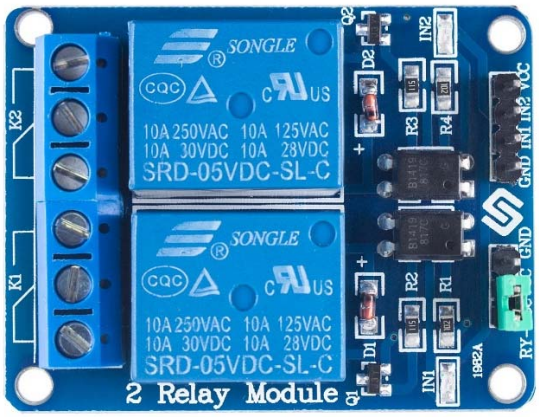


Figura 11 Modulo de 2 relés con acoplamiento óptico

La placa del microcontrolador dispone de 8 conectores que permiten el fácil acceso a 8 de los puertos de E/S de uso general, por lo tanto, se controlarán un máximo de 8 relés independientes por cada remota.

Fuente de alimentación

Ya que la placa de la remota ya incorpora un regulador de voltaje que ajusta la tensión al valor requerido para el microcontrolador, se puede alimentar el circuito con cualquier fuente de corriente directa de 4 a 11 voltios. Por esta razón se utilizó se utilizó una fuente de poder de 5 voltios DC para teléfonos móviles, lo cual permitió conectar el prototipo a un tomacorriente de 120 voltios AC a 60 Hz.

ORGANIZACIÓN DE LOS DATOS EN LA MEMORIA

La memoria EEPROM puede representarse como una serie de renglones, uno encima del otro, donde cada uno corresponde a 8 Bits (1 Byte) de información, estos renglones se enumeran desde 0 a 32767 (en hexadecimal 0000h a 7FFFh).

Los renglones de información se dividieron en 3 secciones. La primera permite ajustar los parámetros generales de la remota, por ejemplo: cambiar ID de esta en la red Modbus, almacenar la velocidad de trasmisión de tatos, indicar que circuitos están conectados o desactivados; la segunda sección conforma la información de los horarios para los días de la semana (de lunes a domingo); y la tercera sección contiene la información de los horarios de cada "día especial". En la tabla 4 se muestra la estructura general de los datos en la memoria EEPROM.

Tabla 4 Organización general de la información en la EEPROM

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	# Bytes
0000h a 0030h	Control de la remota								48 Bytes
0032h a 0942h	Tablas horarias (Rutina)								2320 Bytes
0960h a 7F26h	Tablas horarias (días especiales)								30150 Bytes
7F27h a 7FFFh	Espacio vacío								216 Bytes

Control de la remota

Esta sección de memoria fue diseñada para el control e información general de la remota. El primer Byte se emplea para almacenar la dirección de la remota en la red Modbus, el segundo Byte para almacenar la velocidad de trasmisión a la cual se realizará comunicación (predeterminada a 9600 Baudios) y se dejaron 38 Bytes para futuras aplicaciones, por ejemplo: número de remota, posición física, etc. Los últimos 8 Bytes de este grupo, se emplean para activar o desactivar los circuitos que controla la remota, al colocar 0 (0x00 en sexagesimal) el circuito se desactiva e ignora cualquier

programación previa, y al colocar 240 (F0h en sexagesimal) el circuito sigue la programación en las tablas horarias. La tabla 5 muestra cómo se organizan los Bytes en esta sección de la memoria.

Tabla 5 Organización de los Bytes de la sección "Control de la remota"

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	# Bytes
0000h	Dirección Modbus de remota								1 Byte
0001h	Baudrate								1 Byte
0002h a 0027h	Futuras aplicaciones								38 Bytes
0028h	Estado del circuito #0								1 Byte
0029h	Estado del circuito #1								1 Byte
002Ah	Estado del circuito #2								1 Byte
002Bh	Estado del circuito #3								1 Byte
002Ch	Estado del circuito #4								1 Byte
002Dh	Estado del circuito #5								1 Byte
002Eh	Estado del circuito #6								1 Byte
002Fh	Estado del circuito #7								1 Byte

Tablas Horarias

Las tablas horarias contienen la información de los eventos de cada día para cada uno de los circuitos a controlar, los eventos son los que indican como debe operar la remota. Estos están conformados por una hora de inicio y una de final, formando un intervalo de tiempo durante el cual un circuito debe ser activado. Los eventos se limitaron a un máximo de 10 por día por cada circuito a controlar.

Las tablas horarias se agruparon según su función en una sección para los horarios de "rutina semanal" y otra para los horarios de los "días especiales".

Tabla horaria de la "rutina semanal"

Esta sección contiene los eventos de la rutina principal de la remota, la cual se repite cada semana de lunes a domingo. La información de esta tabla se divide en 8

subgrupos (enumerados del 0 al 7) que representan cada uno de los circuitos a controlar. La tabla 6 muestra la organización general de esta sección de memoria.

Tabla 6 Organización global de los Bytes de la sección " Rutina semanal"

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	# Bytes	Circuito
0032h a 0153h	Eventos de los 7 días de la semana								290 Bytes	Circuito #0
	3 Bytes vacíos									
0154h a 0275h	Eventos de los 7 días de la semana								290 Bytes	Circuito #1
	3 Bytes vacíos									
0276h a 0397h	Eventos de los 7 días de la semana								290 Bytes	Circuito #2
	3 Bytes vacíos									
0398h a 04B9h	Eventos de los 7 días de la semana								290 Bytes	Circuito #3
	3 Bytes vacíos									
04BAh a 05DBh	Eventos de los 7 días de la semana								290 Bytes	Circuito #4
	3 Bytes vacíos									
05DCh a 06FDh	Eventos de los 7 días de la semana								290 Bytes	Circuito #5
	3 Bytes vacíos									
06FEh a 081Fh	Eventos de los 7 días de la semana								290 Bytes	Circuito #6
	3 Bytes vacíos									
0820h a 0941h	Eventos de los 7 días de la semana								290 Bytes	Circuito #7
	3 Bytes vacíos									

Cada subgrupo se organiza según los días de la semana, siendo el primero el día lunes y el ultimo el domingo. Adicionalmente, se agregó 3 Bytes sin información al final para espaciar horarios de cada circuito y evitar solapamiento. La tabla 7 muestra cómo se organiza la información para cada circuito, tomando como ejemplo el circuito número 0.

Tabla 7 Organización detallada de los Bytes para un Circuito

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	# Bytes	Circuito
0032h a 005Ah	Número de eventos día 1 (Lunes)								41 Bytes	Circuito #0
	Eventos del lunes									
005Bh a 0083h	Número de eventos día 2 (Martes)								41 Bytes	
	Eventos del martes									
0084h a 00ACh	Número de eventos día 3 (Miércoles)								41 Bytes	
	Eventos del miércoles									
00ADh a 00D5h	Número de eventos día 4 (Jueves)								41 Bytes	
	Eventos del jueves									
00D6h a 00FEh	Número de eventos día 5 (Viernes)								41 Bytes	
	Eventos del viernes									
00FFh a 0127h	Número de eventos día 6 (Sábado)								41 Bytes	
	Eventos sábado									
0128h a 0150h	Número de eventos día 7 (Domingo)								41 Bytes	
	Eventos domingo									
0151h a 0153h	Espacio vacío								3 Bytes	

La tabla 8 muestra cómo se organizan los eventos para un de los días de la semana. Cada día está formado por 10 eventos, y estos a su vez por 4 Bytes de información que contiene la hora de inicio y fin del intervalo de tiempo. El conjunto de eventos va encabezado por un Byte, para indícale a la remota cuantos eventos debe leer para ese día (empezando desde el primero), esto permite limitar la cantidad de eventos si así se desea.

Tabla 8 Organización detallada de los eventos para un día de la semana

<table>
<tr><th>ADDRESS</th><th>BIT 7</th><th>BIT 6</th><th>BIT 5</th><th>BIT 4</th><th>BIT 3</th><th>BIT 2</th><th>BIT 1</th><th>BIT 0</th><th># Evento</th><th># Bytes</th></tr>
<tr><td>0032h</td><td colspan="8">Número de eventos del día Lunes</td><td></td><td rowspan="24">41 Bytes</td></tr>
<tr><td>0033h</td><td colspan="8">Hora inicio</td><td rowspan="4">Evento #1</td></tr>
<tr><td>0034h</td><td colspan="8">Minutos inicio</td></tr>
<tr><td>0035h</td><td colspan="8">Hora fin</td></tr>
<tr><td>0036h</td><td colspan="8">Minutos fin</td></tr>
<tr><td>0037h</td><td colspan="8">Hora inicio</td><td rowspan="4">Evento #2</td></tr>
<tr><td>0038h</td><td colspan="8">Minutos inicio</td></tr>
<tr><td>0039h</td><td colspan="8">Hora fin</td></tr>
<tr><td>003Ah</td><td colspan="8">Minutos fin</td></tr>
<tr><td>003Bh</td><td colspan="8">Hora inicio</td><td rowspan="4">Evento #3</td></tr>
<tr><td>003Ch</td><td colspan="8">Minutos inicio</td></tr>
<tr><td>003Dh</td><td colspan="8">Hora fin</td></tr>
<tr><td>003Eh</td><td colspan="8">Minutos fin</td></tr>
<tr><td>003Fh
a
0056h</td><td colspan="8">Horas y minutos</td><td>Evento #4
al
Evento #9</td></tr>
<tr><td>0057h</td><td colspan="8">Hora inicio</td><td rowspan="4">Evento
#10</td></tr>
<tr><td>0058h</td><td colspan="8">Minutos inicio</td></tr>
<tr><td>0059h</td><td colspan="8">Hora fin</td></tr>
<tr><td>005Ah</td><td colspan="8">Minutos fin</td></tr>
</table>

41 Bytes comprendería solo los 10 eventos de un día de la semana para uno de los circuitos a controlar. Por lo tanto, para controlar 8 salida, los 7 días de la semana se requieren 2296 Bytes (2320 contabilizando los 3 Bytes de separación entre los datos de cada circuito).

Tabla horaria de los "días especiales"

La última sección de la memoria contiene los eventos de los "días especiales". A diferencia de la rutina semanal, estos ocurren solo un día al año y tienen prioridad sobre los días de la semana, es decir, si la fecha actual (Mes y fecha) coinciden con la de alguno de los días especiales, se obviará la rutina semanal y se utilizaran los datos de esta tabla.

La tabla horaria se divide en subgrupos donde cada uno representa un día especial, y va encabezada por un Byte que le indícale a la remota cuantos días especiales debe verificar (empezando desde el primero). La tabla 9 muestra la organización general de los días especiales en la memoria.

Tabla 9 Organización global de los Bytes de la sección " Días especiales"

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	# Bytes	Fecha
0960h	Número de días especiales (N)								1 Byte	
0961h a 0AAFh	Mes y Fecha día especial #0								335 Bytes	Fecha especial #1
	Eventos de los 8 circuitos									
	Espacio vacío									
00AB0h a 0BFEh	Mes y Fecha día especial #1								335 Bytes	Fecha especial #2
	Eventos de los 8 circuitos									
	Espacio vacío									
00BFFh a 0D4Dh	Mes y Fecha día especial #2								335 Bytes	Fecha especial #3
	Eventos de los 8 circuitos									
	Espacio vacío									
0D4Eh a 07DD7h	Eventos de los 8 circuitos								28810 Bytes	Fechas del #4 al #89
7DD8h a 7F26h	Mes y Fecha día especial #90								335 Bytes	Fecha especial #90
	Eventos de los 8 circuitos									
	Espacio vacío									

Cada subgrupo contiene la información de los eventos para los 8 circuitos a controlar y dos Bytes de cabecera con la información de mes y fecha en la que deben aplicarse esta tabla horaria. De igual manera que antes, se dejó un espacio vacío al final (de 5 Bytes esta vez) para separar un día especial de otro. La tabla 10 muestra cómo se organiza la información para un día especial, tomando como ejemplo el primero.

Tabla 10 Organización detallada de los Bytes para un día especial

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	# Bytes	Fecha
0961h	Mes								2 Bytes	Fecha especial #1
0962h	Fecha									
0963h a 098Bh	Número de eventos del circuito #0								41 Bytes	
	Eventos									
098Ch a 09B4h	Número de eventos del circuito #1								41 Bytes	
	Eventos									
09B5h a 09DDh	Número de eventos del circuito #2								41 Bytes	
	Eventos									
09DEh a 0A06h	Número de eventos del circuito #3								41 Bytes	
	Eventos									
0A07h a 0A31h	Número de eventos del circuito #4								41 Bytes	
	Eventos									
0A30h a 0A58h	Número de eventos del circuito #5								41 Bytes	
	Eventos									
0A59h a 0A81h	Número de eventos del circuito #6								41 Bytes	
	Eventos									
0A82h a 0AAAh	Número de eventos del circuito #7								41 Bytes	
	Eventos									
0AABh a 0AAFh	Espacio vacío								5 Bytes	

Como se muestra en la tabla 11, la organización interna de los eventos dentro de cada equipo es similar a la de la rutina semanal. La información dentro de cada circuito está formada por los 10 eventos y esto encabezados por un Byte que indica cuantos eventos deben leerse para el día especial.

Tabla 11 Organización detallada de los eventos para uno de los circuitos

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	# Evento	# Bytes
0963h	Número de eventos del circuitos #0									41 Bytes
0964h	Hora inicio								Evento #1	
0965h	Minutos inicio									
0966h	Hora fin									
0967h	Minutos fin									
0968h	Hora inicio								Evento #2	
0969h	Minutos inicio									
096Ah	Hora fin									
096Bh	Minutos fin									
096Ch	Hora inicio								Evento #3	
096Dh	Minutos inicio									
096Eh	Hora fin									
096Fh	Minutos fin									
0970h a 0987h	Horas y minutos								Evento #4 al Evento #9	
0988h	Hora inicio								Evento #10	
0989h	Minutos inicio									
098Ah	Hora fin									
098Bh	Minutos fin									

Una vez establecidas las dos primeras secciones de la memoria, quedaron disponible 30367 Bytes, estos se usaron para almacenar la tabla de los días especiales. Como cada uno de estos días requiere de 335 Bytes para almacenar la información de sus eventos, 90 es el número máximo de días extraordinarios que admitirá la remota. Con esto se obtiene un total de 30150 Bytes.

RREGISTROS MODBUS

Los esclavos en una red Modbus deben tener definido unos espacios de memoria que permita al dispositivo maestro acceder a la información, estos espacios de memoria se llaman registros y su función varía según el equipo. Para esta remota se crearon 104 registros que permiten operar la remota y rellenar de forma organizada las tablas de información expuestas anteriormente.

A la remota se le implementaron las funciones 3 y 6 del protocolo Modbus que son las instrucciones básicas de lectura y escritura respectivamente. Estas permiten a otros dispositivos acceder a la información dentro del equipo y modificarla. En la tabla 12 se muestran los registros a los que pueden acezarse con estas dos funciones.

Tabla 12 Tabla de Registros Modbus de la remota

Registros	Nombre	Uso
0	Modbus ID	Contiene el ID de la remota
1	Estado de las salidas	Indica estado de los circuitos de salidas
2	RTC segundos	Al activar el col RTC se asigna esta información al "RTC"
3	RTC minutos	
4	RTC hora	
5	RTC día	
6	RTC fecha	
7	RTC mes	
8	RTC año	
9	Estado Circuito #0	Estado de los circuitos
10	Estado Circuito #1	
11	Estado Circuito #2	
12	Estado Circuito #3	
13	Estado Circuito #4	
14	Estado Circuito #5	
15	Estado Circuito #6	
16	Estado Circuito #7	
17	Numero de Circuito a leer/escribir	Seleccionar circuito de salida
18	Día de la semana a leer/escribir	Seleccionar día de la semana
19	Número del día Especial a leer/escribir	Seleccionar día especial
20 al 60	Información de la rutina semanal para el Circuito y día seleccionado	Información de la tabla horaria seleccionada con los registros 17 y 18
61	Cantidad de días especiales	Los días especiales
62	Mes del día especial	Información de la tabla horaria seleccionada con los registros 17, 18 y 19
63	Fecha del día especial	
64 al 104	Información del día especial para el Circuito y día seleccionado	

Registro 0: Muestra la dirección Modbus de la remota, al escribir en él se asigna una nueva dirección.

Registro 1: Muestra el estado de los circuitos de salida de la remota. Representados como un numero de 8 Bits, donde cada Bit representa uno de los circuitos como se muestra en la figura 12. Si el Bit es igual a "1" la salida se encuentra activa.

```
Circuito: #7 #6 #5 #4 #3 #2 #1 #0
Bits:     MSB.................LSB
```

Figura 12 Byte que representa el estado de los circuitos de salida

Registro 2 al 8: Estos registros son empleados para fijar la fecha y hora del RTC. El reloj emplea el sistema de 24 horas, donde las 00:00:00 representa la media noche y el comienzo del día. Escribir en estos registros no asigna automáticamente la hora del reloj, para ello se debe activar el registro correspondiente a la Bobina 9.

Registro 9 al 16: Estos registros representan el Byte de estado de las salidas de la remota, se emplean para activar o desactivar los circuitos a controlar. Al colocar al colocar 0 (0x00 en sexagesimal) el circuito se desactiva e ignora cualquier programación previa, y al colocar 240 (F0h en sexagesimal) el circuito sigue la programación en las tablas horarias

Registro 17: Selecciona el circuito que se desea acceder de la tabla horaria (ver registros "20 al 60" o "64 al 105"). Los circuitos se enumeran desde el 0 hasta el 7.

Registro 18: Selecciona el día que se desea acceder de la tabla (ver registros "20 al 60"). Los días se enumeran del 1 al 7 siendo lunes el primero y domingo el ultimo.

Registro 19: Selecciona el día especial que se desea acceder de la tabla horaria (ver registros "64 al 105"). Los días especiales se enumeran desde el 0 hasta el 89.

Registro 20 al 60: Estos 41 registros permiten leer y escribir los eventos en las tablas horarias de la rutina semanal, cada uno de ellos representa directamente los bytes de la tabla horaria como se mostraron en la tabla 8. La información de estos registros corresponde al *día* y *circuito* seleccionado mediante los registros 17 y 18. Por ejemplo, si se desea asignar algún evento a la salida 5 para el día "JUEVES" primero se debe escribir en el registro N°17 en valor de "5" y en el registro N°18 el valor de "4" para luego llenar los eventos con la información deseada.

Los 41 registros se organizan como muestra la tabla 13. Donde el Registro 20 es la cantidad de eventos del día, (si se coloca 0 en este registro ningún evento ocurrirá ese día). El resto de los registros son usados para asignar la hora y minutos de cada evento.

Tabla 13 Organización detallada de los registros del 20 al 60

Registro	Nombre	# Evento
20	Número de eventos	
21	Hora inicio	Evento #1
22	Minutos inicio	
23	Hora fin	
24	Minutos fin	
25	Hora inicio	Evento #2
26	Minutos inicio	
27	Hora fin	
28	Minutos fin	
29 a 56	Horas y minutos	Evento #3 al Evento #9
57	Hora inicio	Evento #10
58	Minutos inicio	
59	Hora fin	
60	Minutos fin	

Registro 61: Este registro contiene la cantidad de días especiales a usar. Si se coloca 0, ningún día especial será tomado en cuenta y solo se ejecutará la rutina semanal.

Registros 62 y 63: Según el valor que se escriba en el registro 19, los registros 62 y 63 muestran el mes y fecha del día especial seleccionado. De esta forma se determina que día debe ocurrir la rutina especial.

Registros del 64 al 104: Similares en funcionamiento a los registros "20 a 60", estos permiten escribir y leer los eventos en la tabla horaria de los días especiales. La información en ellos corresponde al *circuito* y *día especial* seleccionado mediante los registros 17 y 19. Los 41 registros se encuentran organizados como muestra en la tabla 14, Donde el Registro 64 es la cantidad de eventos del día especial (si se coloca 0 en este registro ningún evento se usará ese día). El resto de los registros son usados para asignar la hora y minutos de cada evento.

Tabla 14 Organización detalla de los registros del 64 al 104

Registro	Nombre	# Evento
64	Número de eventos del día para el circuito selecciona	
65	Hora inicio	Evento #1
66	Minutos inicio	
67	Hora fin	
68	Minutos fin	
69	Hora inicio	Evento #2
70	Minutos inicio	
71	Hora fin	
72	Minutos fin	
73 a 100	Horas y minutos	Evento #3 al Evento #9
101	Hora inicio	Evento #10
102	Minutos inicio	
103	Hora fin	
104	Minutos fin	

Adicionalmente se crearon 10 registros especiales que representan salidas virtuales dentro del microcontrolador. Para acceder a estos registros se requiere de la función 5 del protocolo Modbus, función que se emplea para fijar el estado de ON/OFF a una salida digital. En la tabla 15 se muestran estos 10 registros

Tabla 15 Tabla de registros para las puertos de salida virtuales de la remota

Registro	Nombre	Uso
0	Forzar Circuito #0	Permite forzar los circuitos de salida a ON/OFF
1	Forzar Circuito #1	
2	Forzar Circuito #2	
3	Forzar Circuito #3	
4	Forzar Circuito #4	
5	Forzar Circuito #5	
6	Forzar Circuito #6	
7	Forzar Circuito #7	
8	Reiniciar Relés	Los relés forzados retoman el funcionamiento automático
9	RTC set	Fija la hora del RTC según los registros

Bobina 0 al 7: Estos permiten forzar los circuitos de salida a ON u OFF, de modo que obviaran cualquier tabla horaria. El estado de la salida permanecerá como el asignado hasta las 12 de la noche donde la remota vuelve a su funcionamiento automático. Si se desea apagar un circuito de salida permanentemente se debe usar los registros “9 a 16” anteriormente explicados.

Bobina 8: Al enviar una señal de ON a esta bobina digital, se reinician todos los relés forzados y vuelven a su funcionamiento automático.

Bobina 9: Al enviar una señal de ON a esta bobina virtual, se asignan los valores de los registros “2 a 8” al RTC. De esta forma se programa el reloj de la remota.

EL ALGORITMO

El funcionamiento de la remota es el siguiente:

A las 12 de la noche o al reiniciar el equipo, la remota verifica la fecha y determina si hay algún día especial programado para la fecha actual, de ser así toma los eventos de la tabla de los días especiales, de lo contrario usa los de la rutina semanal para el día que corresponda. Una vez establecidos los eventos que se usaran para las próximas 24 horas, la remota verifica constantemente la hora y la compara con las tablas horarias seleccionadas, de este modo determinara cuando deben estar activos los circuitos de salida.

Al ser un dispositivo esclavo en la red Modbus, la remota monitorea siempre la red en caso de que algún mensaje este dirigida a ella. Si algún mensaje es detectado, determina si la información es para ella. De ser así la remota evaluará la información, tomará la acción correspondiente y devolverá el mensaje de respuesta al maestro de la red Modbus.

Definido el funcionamiento del sistema, se decidió tratar el algoritmo dividiéndolo en dos partes y trabajar cada una de ellas por separado. La primera parte consta del control horario, esta se encarga de leer las tablas de la memoria, operar el reloj y manejar las salidas. La segunda parte consiste en la comunicación Modbus, en esta se implementó el protocolo y contiene las funciones para evaluar la información y decidir qué acciones tomar según la trama recibida.

Control horario

El manejo de las tablas horarias es independiente de la comunicación Modbus, por lo tanto, la remota puede realizar el control de circuitos de salida siempre y cuando estén predefinidos los horarios en su memoria.

El primer paso para realizar el control horario fue implementar las funciones de comunicación para el Bus I^2C. Como el PIC24FJ64GA004 ya dispone de un hardware dedicado a la comunicación por I^2C, solo fue necesario emplear los registros de moría correspondientes para trasmitir datos con este protocolo. De este modo el microcontrolador tiene acceso al reloj en tiempo real y a la información de las tablas horarias contenida en la EEPROM.

Una vez hecho esto el siguiente paso consistió en crear el resto de funciones que verificaran contantemente la hora y la compararan con la tabla horaria

El algoritmo de la primera parte es el siguiente: al iniciar la remota se establecen los parámetros para el funcionamiento del microcontrolador (iniciar temporizadores, configurar el UART, etc.). Hecho esto, se toma la fecha del RTC y se compara con cada día especial guardado en la memoria, de coincidir con alguno se tomarán los eventos de cada circuito de salida para ese día especial, de lo contrario se usarán los eventos del día de la semana que corresponda. Establecido los eventos del que deben ocurrir el día actual, se consulta la hora en el RTC cada 500 ms y se compara con la información de loes eventos seleccionados para determinar cuándo deben estar activas las salidas. A media noche el sistema reinicia las salidas que fueron forzadas a ON/OFF, y repite el procedimiento de verificar la fecha con los días especiales para determinar que tabla horaria corresponde al próximo día. La figura 13 muestra el diagrama de flujo para el control horario.

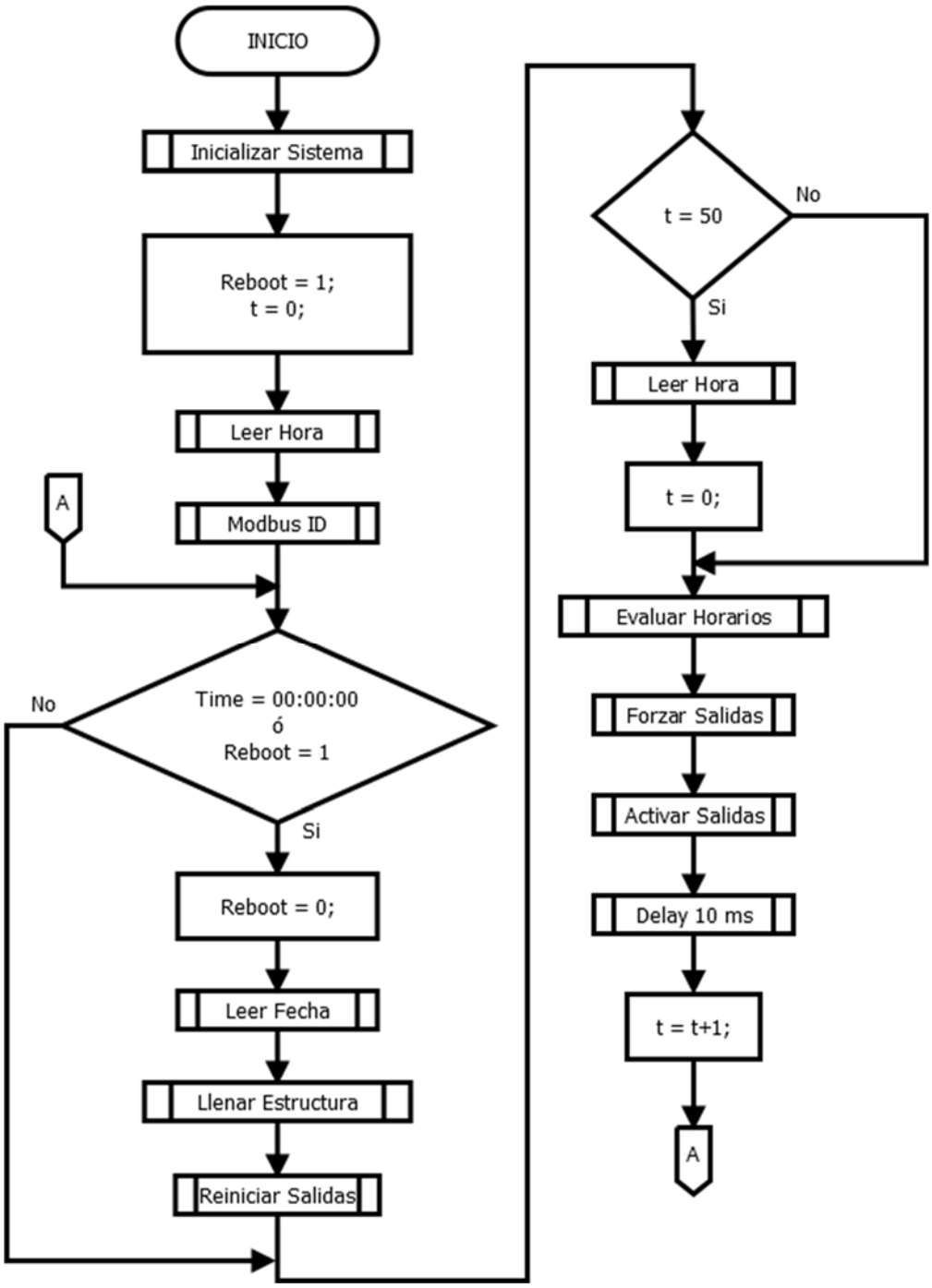


Figura 13 Diagrama de flujo para el control horario

A continuación, se explica cada una de las funciones del diagrama anterior:

- **Inicializar Sistema**: Esta función fija los parámetros que requiere el microcontrolador para su funcionamiento, como por ejemplo frecuencia del cristal interno, configuración de los pines de salida/entrada, ajuste del UART, ajuste de los temporizadores, etc.
- **Leer Hora**: Esta función lee la hora, minutos y segundos del RTC, y entrega al programa principal estos tres valores.
- **Modbus ID**: Se encarga de leer la posición de la memoria EEPROM designada para almacenar la dirección Modbus de la remota (0000h) y la asigna a una variable, la cual se emplea en las funciones relacionadas al protocolo Modbus.
- **Leer Fecha**: Esta función lee el mes, fecha y día de la semana del RTC, y entrega al programa principal estos tres valores.
- **Llenar Estructura**: Esta función toma los valores del mes, fecha y día de la semana proveniente de la función "Leer Fecha", y compara estos valores con los días especiales para determinar de qué tabla horaria (rutina semanal o días especiales) debe tomar los eventos. Una vez seleccionados estos, la información se organiza en una estructura de datos. Esta estructura se muestra en la figura 14, donde "n" es el número del evento y "m" el número del circuito de salida.

```
typedef struct {
    uint8_t Hora;
    uint8_t Mins;
}Hour;

struct {
    uint8_t Byte_Control;
    uint8_t n_Eventos;
    Hour Activar[n];
    Hour Desactivar[n];
}Circuito[m];
```

Figura 14 Estructura de datos para las tablas horarias (C++)

- **Evaluar Horarios**: Esta función Toma los valores de la hora, minutos y segundos provenientes de la función “Leer Hora” y compara esos valores con los eventos almacenados en la estructura de datos para decidir si debe o no activar las salidas. Esta función devuelve un valor de 8 Bits donde cada Bit representa uno de los circuitos a controlar, de igual forma como se mostró en la figura 12.
- **Forzar Salidas**: Lee los valores de los registros Modbus correspondientes a las “Bobinas 0 a 7” y de ser necesario modifica los 8 Bits provenientes de la función “Evaluar Horarios”, para activar o desactivar los circuitos de salida deseados.
- **Activar salidas**: Esta función toma los 8 Bits de los circuitos y activa las salidas del microcontrolador correspondientes. Por ejemplo, si los 8 Bits tiene el valor de 10001010b en binario quiere decir que se activaran los circuitos 7, 3 y 1.
- **Reiniciar Salidas**: esta función reinicia los valores de los registros Modbus correspondientes a las “Bobinas 0 a 7”, por lo tanto, las salidas de los circuitos volverán a su funcionamiento automático según las tablas horarias.
- **Delay 10 ms**: esta función detiene el programa principal por 10 ms para frenar al microcontrolador y reducir la cantidad de lecturas del RTC a una cada 500 ms, de este modo se evita la congestión Bus I^2C.

Comunicación Modbus

La segunda parte del algoritmo consiste en la transmisión y recepción de datos de la red Modbus. Para implementar el protocolo de comunicación se utilizaron las interrupciones del micro controlador, de este modo la remota siempre está pendiente de Bus de comunicación sin dejar a un lado el control horario. Las interrupciones utilizadas fueron las de transmisión del UART, recepción del UART y la del temporizador 2.

Al iniciar la remota el transceptor se coloca en modo de recepción para que cualquier mensaje que pase por el Bus pueda llegar al UART. La función de interrupción de recepción se activa cuando el primer Byte de una trama es recibido, seguidamente se tomará cada uno de los Byte y se almacena en un arreglo llamado “Buffer_Rx”. A medida que se reciben los Bytes, esta función reinicia el temporizador 2 para indicar que no se ha llegado los 3,5 caracteres de silencio que marcan el final del mensaje Modbus. La figura 15 muestra el diagrama de flujo de la función de interrupción.

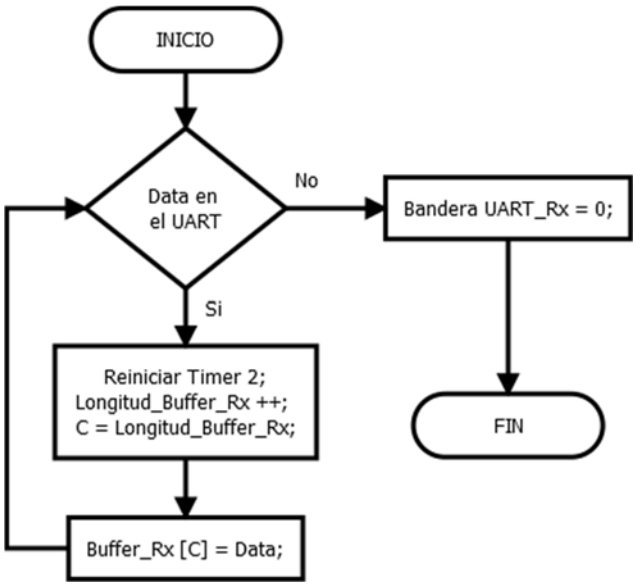


Figura 15 Diagrama de flujo de la rutina de interrupción para la Recepción

Para medir los 3,5 caracteres en silencio que debe haber entre cada trama, primero es necesario determinar cuánto equivalen estos en segundos. A 9600 baudios, 1 carácter (8 Bits) tarda 833,33 µs en trasmitirse, por el tanto 3,5 caracteres equivaldrán a 2910 µs. Sabiendo esto, la interrupción de temporizador 2 se fijó para que ocurriera cada 3000 µs. Tiempo que se mide una vez deje de llegar información por el Bus, ya que la función de recepción impide que el temporizador avance mientras estén llegando datos al UART.

Una vez ocurrida la interrupción del temporizador se determina si hay datos en el “Buffer_Rx”. De ser así, se deshabilita la recepción para evitar que llegue otra trama mientras ce procesan los datos. Se toma el primer Bytes del “Buffer_Rx” (correspondiente al ID del dispositivo) y se determina si el mensaje va dirigido a esta remota. De resultar positivo, se evalúa el resto de la información, se construye el mensaje de respuesta en el “Buffer_Tx” y se envía el mensaje forzando la interrupción de trasmisión de UART. Enviado el mensaje, se limpia el “Buffer_Rx” y se habilita de nuevo la recepción de datos en espera de la próxima trama. La figura 16 muestra el diagrama de flujo de la función de interrupción del temporizador.

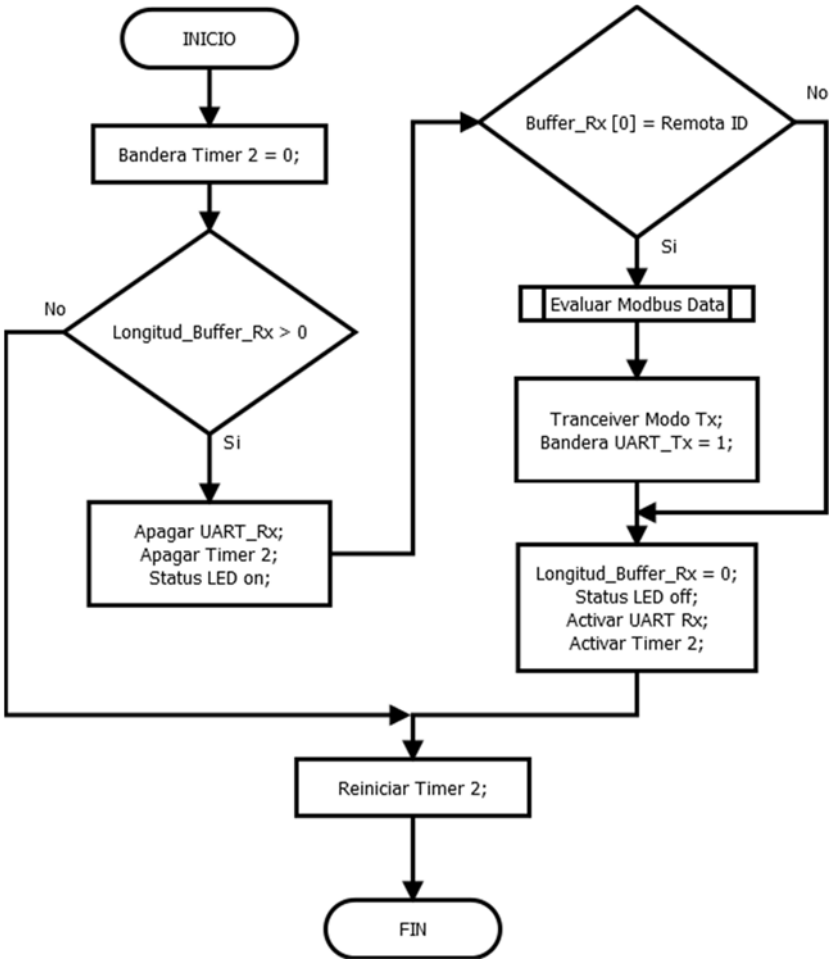


Figura 16 Diagrama de flujo de la rutina de interrupción del temporizador

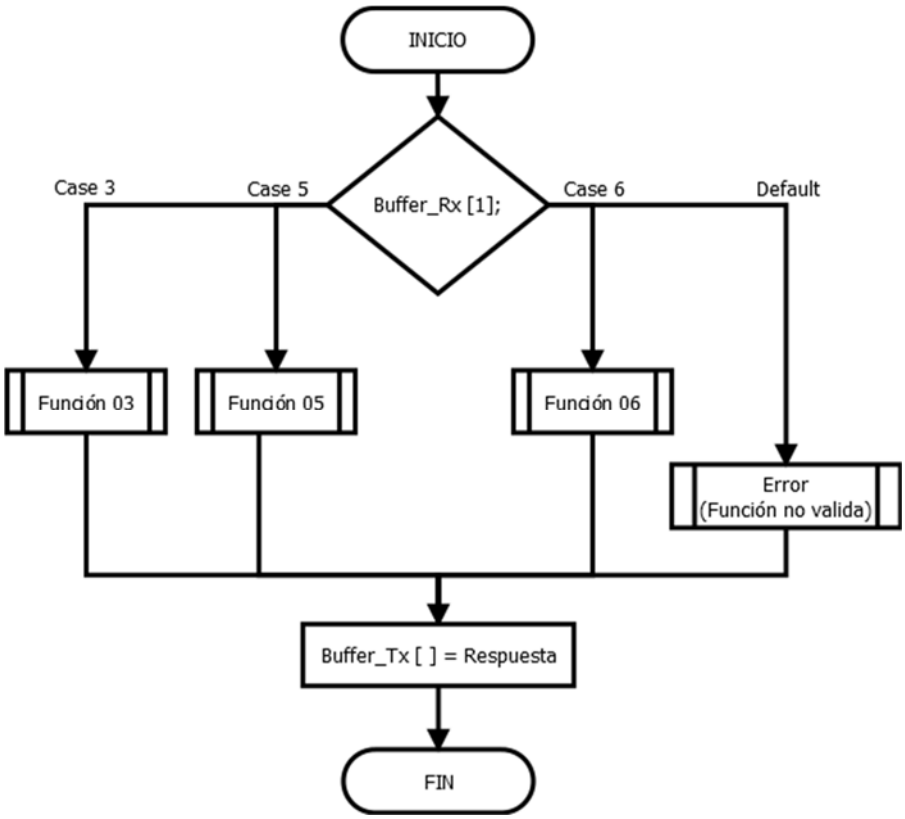


Figura 17 Diagrama de flujo de la rutina "Evaluar Modbus Data"

La función "Evaluar Modbus Data" toma el segundo Byte del "Buffer_Rx" (correspondiente al número de la función Modbus), determina si la función solicitada corresponde a las implementadas en la remota y de ser así entrega el resto de los bytes del "Buffer_Rx" a la función correspondiente:

- **Función 03**: Lee en la memoria EEPROM la información de los registros solicitados y construye el arreglo de respuesta en el "Buffer_Tx" con la información solicitada.
- **Función 05**: Fuerza la salida del microcontrolador indicada y construye el arreglo de respuesta en el "Buffer_Tx" donde confirma al maestro que se produjo la acción solicitada.

- **Función 06**: Escribe en la memoria EEPROM la información del registro indicado y construye la después en el “Buffer_Tx” que indica que la data fue copiada con éxito

De solicitarse una función Modbus no implementada, se ejecuta la función “Error”, esta construye una respuesta en el “Buffer_Tx” indicando que la función no es válida. En la figura 17 se muestra el diagrama de flujo de la función “Evaluar Modbus Data”.

Una vez construida la respuesta en el arreglo “Buffer_Tx, fuerza” se utiliza la interrupción de transmisión del UART para enviar los datos. Esta interrupción ocurre cada vez que se tramite un Byte y se repite hasta que es transmitida toda la trama de respuesta. Una vez tramitado el transceptor se coloca de nuevo en modo recepción y se limpia el buffer de trasmisión. La figura 18 muestra el diagrama de flujo para la interrupción de trasmisión del UART.

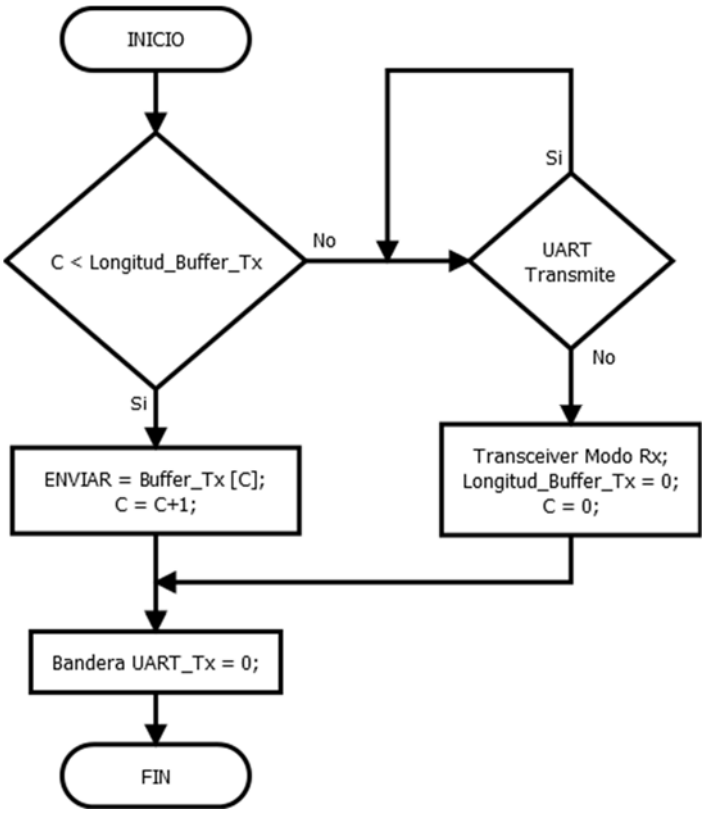


Figura 18 Diagrama de flujo de la rutina de interrupción para la Transmisión

EL HARDWARE

La figura 19 muestra el diagrama de conexiones de los pines del microcontrolador con los demás elementos. Los pines del I_00 al I_07 corresponde a las conexiones de los circuitos a controlar. SDA y SCL son los pines designados para el bus I^2C. UART_Tx y UART_Rx los pines seleccionados para la trasmisión y recepción respectivamente, *Toggle* se emplea para alternar el funcionamiento del transceptor entre transmitir o recibir.

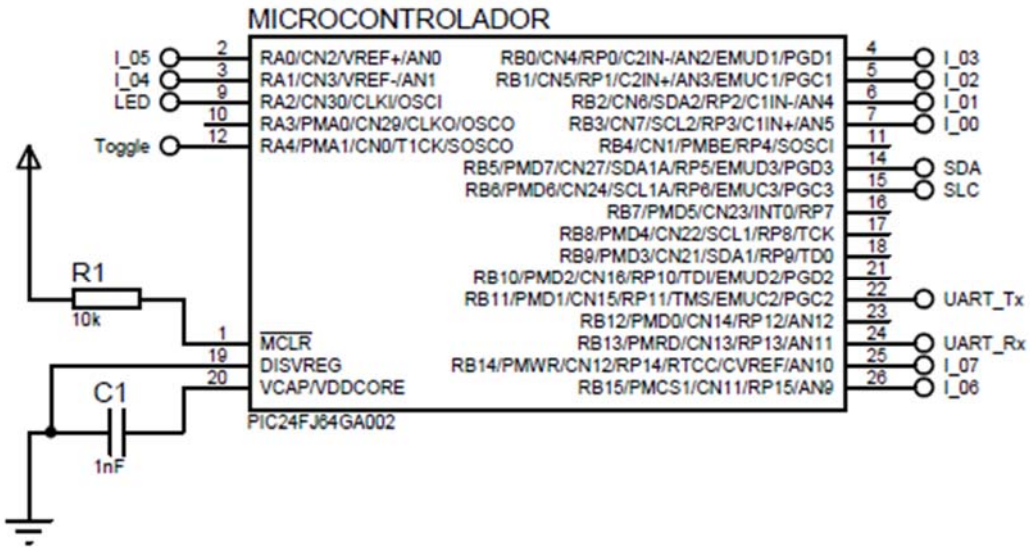


Figura 19 Diagrama de conexión del microcontrolador

La figura 20 muestra la EEPROM y el RTC conectados al bus I^2C, además de las conexiones para el transceptor.

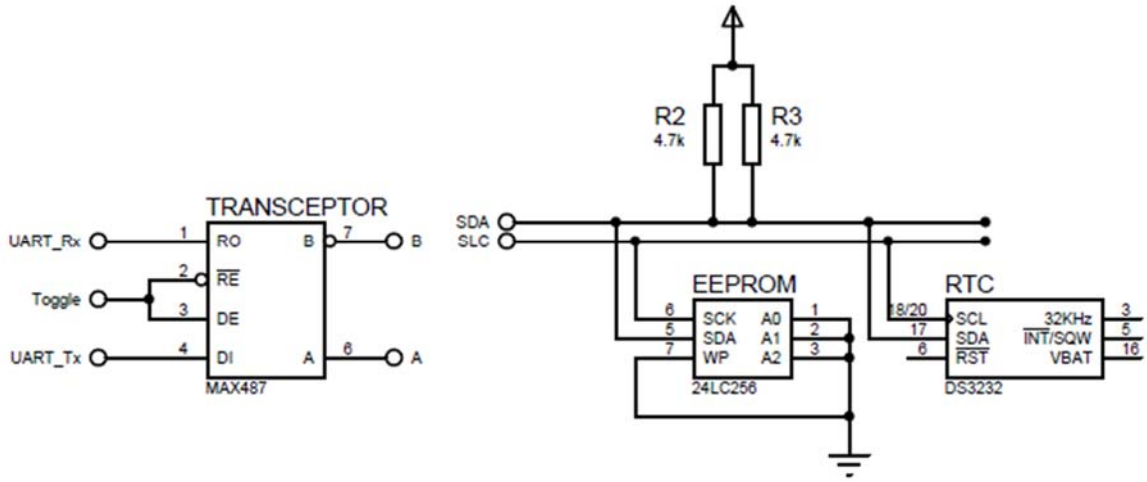


Figura 20 Diagrama de conexión para el RTC, la EEPROM y el Transceptor

La figura 21 muestra el diagrama del circuito para uno de los relés con acoplamiento óptico. El terminal *J1* corresponde al a salida del relé y dispone de dos modalidades normalmente cerrado serrado o normalmente abierto. El terminal *Jumper* permite alimentar el relé con una fuente externa si se desea. El puerto I_0X representa cualquiera de los pines de salida del microcontrolador destinados al control de circuitos.

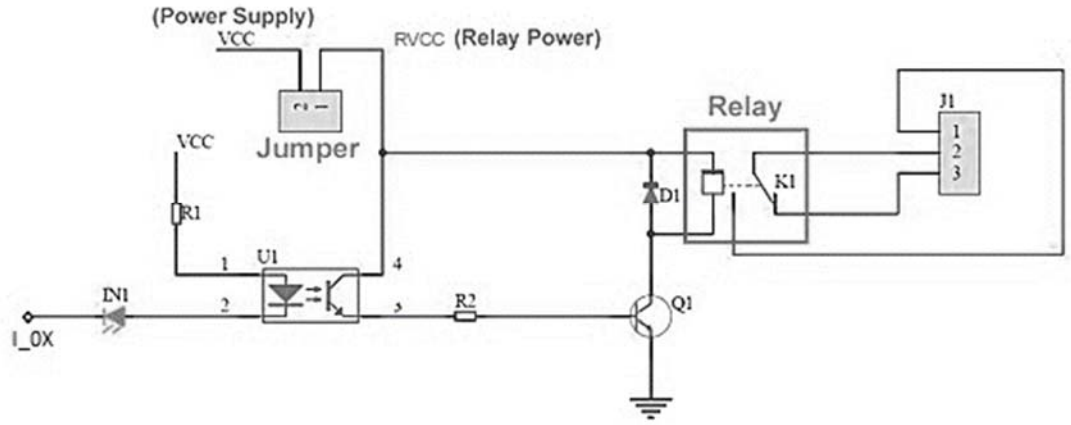


Figura 21 Diagrama del circuito de los relés con acoplamiento óptico

CAPITULO IV

RESULTADOS OBTENIDOS

LA REMOTA

Se construyó una maqueta con una pieza rectangular de MDF que sirvió de soporte al *hardware* del prototipo (el microcontrolador, un *proto-board*, el RTC y la memoria). Para poder visualizar el estado de las salidas durante las pruebas de concepto, se sustituyeron los relés por un juego de ocho LEDs, de esta forma fue posible evaluar el comportamiento de la remota, usando solo la observación. La figura 22 muestra la maqueta sobre la cual se instaló el prototipo final de la remota.

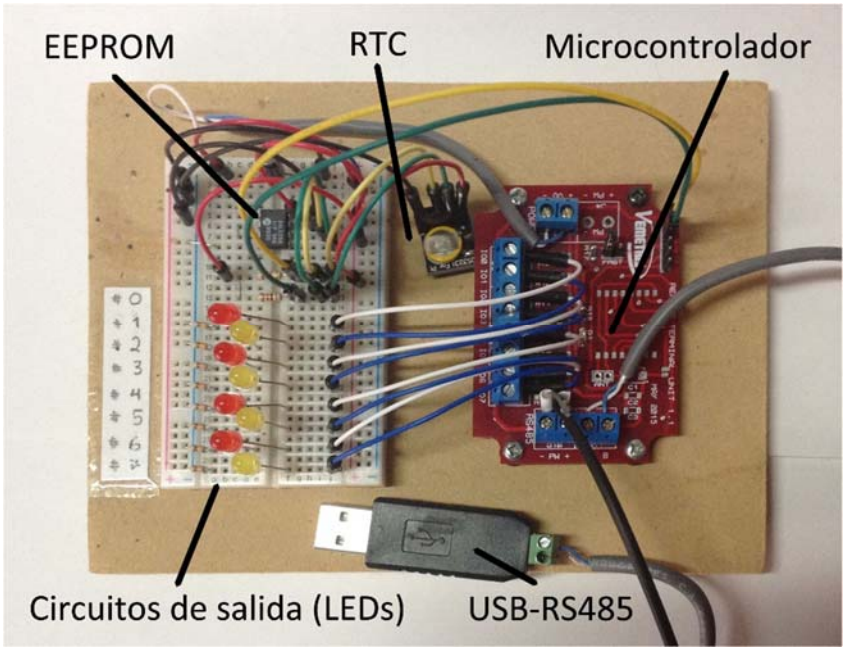


Figura 22 Fotografía de la remota prototipo

VALIDACION DEL PROTOTIPO

Como el dispositivo es un actuador programable, la forma de validar el funcionamiento consistió en enviarles intrusiones a través del puerto RS-485 usando el protocolo Modbus y comprobar si la remota opera de la forma como se espera

Para ello, se empleó el *software* "Modbus Poll" junto a un adaptador USB-RS485 (código de pieza CH340DS). Estas dos herramientas permitieron crear una red Modbus para trasmitir las tramas del protocolo a través del puerto USB de la computadora, utilizando el estándar RS-485. La figura 23 muestra el espacio de trabajo creado en *Modbus Poll,* para el control de la remota.

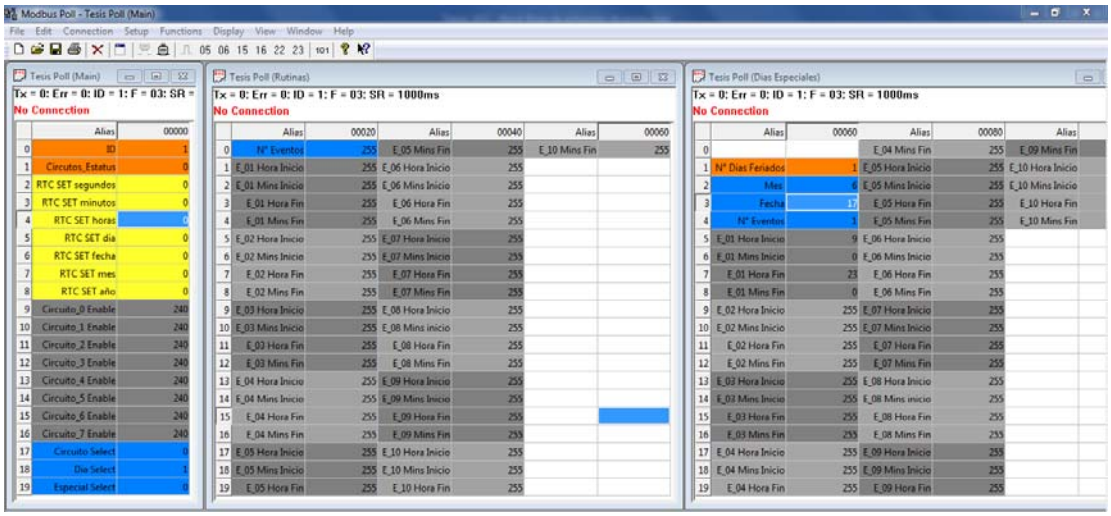


Figura 23 Espacio de trabajo en Modbus Poll

El espacio de trabajo está formado por tres ventanas, donde cada una corresponde a un grupo de registros Modbus. La primera venta son los registros de control de la remota (Registros del 0 al 19). La segunda, los registros para los eventos de la rutina semanal (Registros del 20 al 60). Y la última, para los eventos y fechas de los días especiales (Registros del 61 al 104).

Se creó una tabla horaria de prueba, para la rutina semanal, de los circuitos de salida 0, 1, 2 y 3. Esta se trasfirió a la remota escribiendo en los registros pertinentes del espacio de trabajo creado en *Modbus Poll*. La tabla 16 muestra los horarios de prueba.

Tabla 16 Tabla horaria de prueba

Circuito 0				**Circuito 1**			
	Evento #1	**Evento #2**	**Evento #3**		**Evento #1**	**Evento #2**	**Evento #3**
LU				**LU**	10:20 a 12:00	15:00 a 18:00	19:00 a 20:15
MA	8:30 a 12:30	16:15 a 16:45		**MA**	10:20 a 12:00	15:00 a 18:00	19:00 a 20:15
MI				**MI**	10:20 a 12:00	15:00 a 18:00	19:00 a 20:15
JU	8:30 a 12:30	16:15 a 16:45		**JU**			
VI				**VI**			
SA	9:00 a 17:00	20:30 a 24:00		**SA**			
DO	9:00 a 17:00			**DO**	11:00 a 15:00	17:30 a 20:15	
Circuito 2				**Circuito 3**			
	Evento #1	**Evento #2**	**Evento #3**		**Evento #1**	**Evento #2**	**Evento #3**
LU	8:00 a 9:00	13:00 a 14:00	17:00 a 18:00	**LU**	9:30 a 10:15	13:15 a 14:30	
MA	8:00 a 9:00	13:00 a 14:00	17:00 a 18:00	**MA**	9:30 a 10:15	13:15 a 14:30	16:00 a 16:30
MI	8:00 a 9:00	13:00 a 14:00	17:00 a 18:00	**MI**	9:30 a 10:15	13:15 a 14:30	16:00 a 16:30
JU	8:00 a 9:00	13:00 a 14:00	17:00 a 18:00	**JU**	9:30 a 10:15	13:15 a 14:30	16:00 a 16:30
VI	8:00 a 9:00	13:00 a 14:00	17:00 a 18:00	**VI**	9:30 a 10:15	13:15 a 14:30	
SA	9:00 a 12:00	14:00 a 16:00		**SA**	8:40 a 10:40	16:15 a 21:00	
DO				**DO**	15:00 a 17:00		

Para verificar que toda la información es almacenada correctamente dentro del dispositivo y esta puede sobrevivir a una falla de energía, se cortó deliberadamente la alimentación de la remota por unos segundos. Tras reconectar la energía, se observaron los registros que contenían la tabla horaria de prueba, y la información de los eventos era exactamente la misma que antes de la falla.

A modo de ejemplo, La figura 24 y la figura 25 muestran los registros correspondientes al día lunes para el circuito 2, antes y después de la falla eléctrica respectivamente. Como puede observarse, los eventos permanecieron intactos, por lo tanto, la remota es capaz de retomar su funcionamiento al recuperar la energía eléctrica.

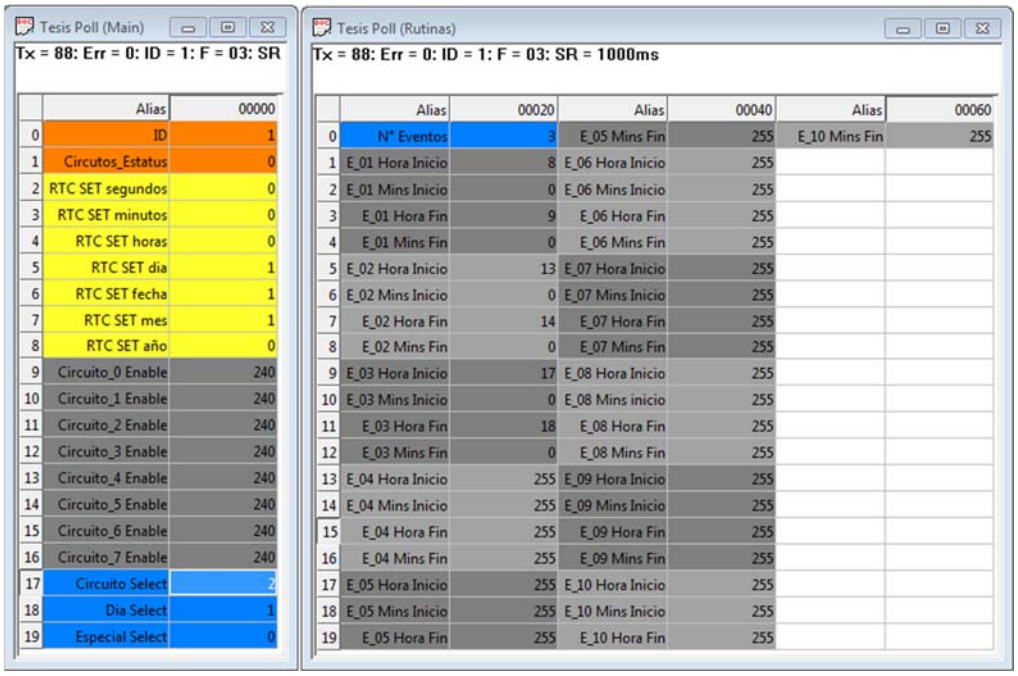
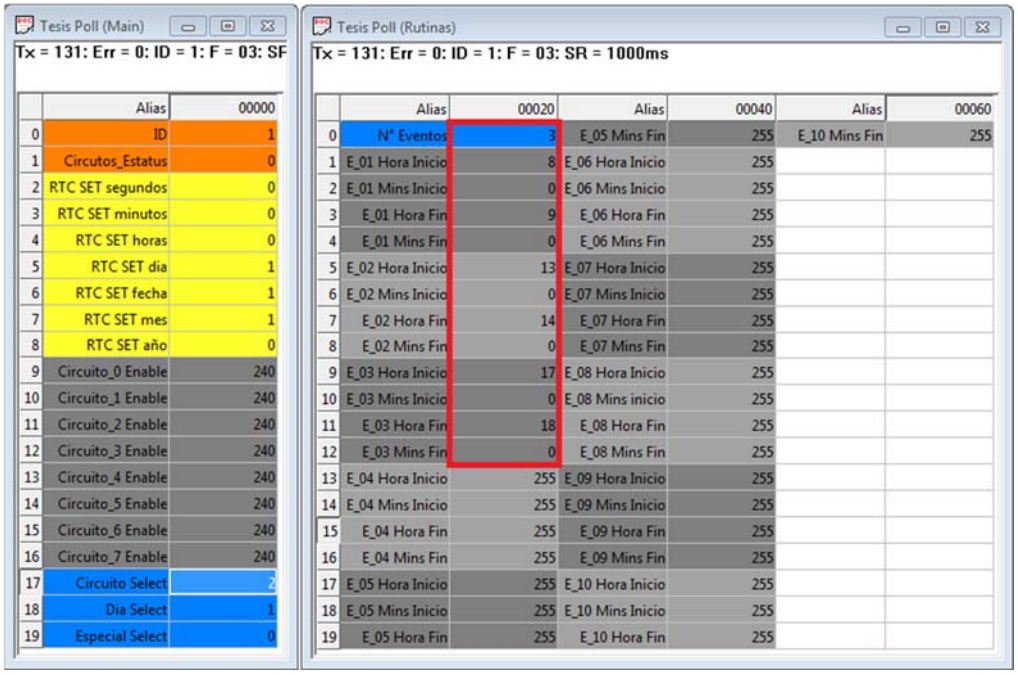


Figura 24 Registros Modbus del Lunes-circuito 2 antes del Reinicio



Tesis Poll (Main)

Tx = 131: Err = 0: ID = 1: F = 03: SF

	Alias	00000
0	ID	1
1	Circutos_Estatus	0
2	RTC SET segundos	0
3	RTC SET minutos	0
4	RTC SET horas	0
5	RTC SET dia	1
6	RTC SET fecha	1
7	RTC SET mes	1
8	RTC SET año	0
9	Circuito_0 Enable	240
10	Circuito_1 Enable	240
11	Circuito_2 Enable	240
12	Circuito_3 Enable	240
13	Circuito_4 Enable	240
14	Circuito_5 Enable	240
15	Circuito_6 Enable	240
16	Circuito_7 Enable	240
17	Circuito Select	2
18	Dia Select	1
19	Especial Select	0

Tesis Poll (Rutinas)

Tx = 131: Err = 0: ID = 1: F = 03: SR = 1000ms

	Alias	00020	Alias	00040	Alias	00060
0	N° Eventos	3	E_05 Mins Fin	255	E_10 Mins Fin	255
1	E_01 Hora Inicio	8	E_06 Hora Inicio	255		
2	E_01 Mins Inicio	0	E_06 Mins Inicio	255		
3	E_01 Hora Fin	9	E_06 Hora Fin	255		
4	E_01 Mins Fin	0	E_06 Mins Fin	255		
5	E_02 Hora Inicio	13	E_07 Hora Inicio	255		
6	E_02 Mins Inicio	0	E_07 Mins Inicio	255		
7	E_02 Hora Fin	14	E_07 Hora Fin	255		
8	E_02 Mins Fin	0	E_07 Mins Fin	255		
9	E_03 Hora Inicio	17	E_08 Hora Inicio	255		
10	E_03 Mins Inicio	0	E_08 Mins inicio	255		
11	E_03 Hora Fin	18	E_08 Hora Fin	255		
12	E_03 Mins Fin	0	E_08 Mins Fin	255		
13	E_04 Hora Inicio	255	E_09 Hora Inicio	255		
14	E_04 Mins Inicio	255	E_09 Mins Inicio	255		
15	E_04 Hora Fin	255	E_09 Hora Fin	255		
16	E_04 Mins Fin	255	E_09 Mins Fin	255		
17	E_05 Hora Inicio	255	E_10 Hora Inicio	255		
18	E_05 Mins Inicio	255	E_10 Mins Inicio	255		
19	E_05 Hora Fin	255	E_10 Hora Fin	255		

Figura 25 Registros Modbus del Lunes-circuito 2 después del Reinicio

Se comparó el estado de las salidas con la tabla horaria de prueba durante siete días para comprobar el funcionamiento de la rutina semanal. Tras la semana de observación, el horario se cumplió exactamente como el establecido confirmando el correcto funcionamiento del equipo. La figura 26 muestra un ejemplo del control horaria para el día martes y la figura 27 para el día sábado, ambas pueden corroborarse comparándolas con la tabla horaria de prueba.

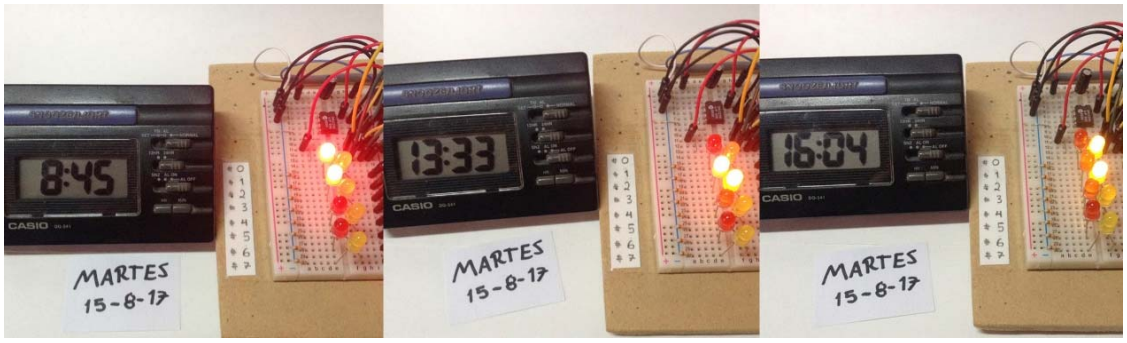


Figura 26 Ejemplos del control horario durante el día Martes

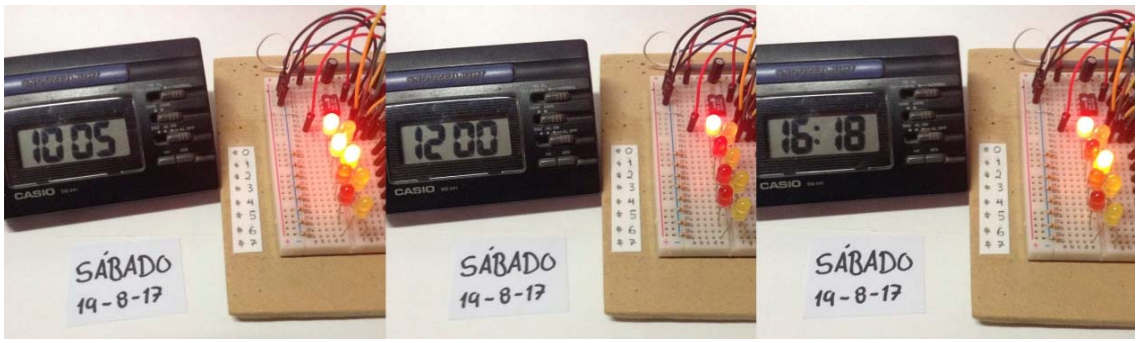


Figura 27 Ejemplos del control horario durante el día Sábado

Después de verificar la "rutina semanal", se agregó un "día especial" a la remota fijado para el 21 de agosto. Para este día se programó un solo evento por salida como se muestra en tabla 17.

Tabla 17 Tabla horaria para el 21 de Agosto

Día especial 21 de Agosto	
Salida	**Evento Único**
Circuito 0	00:00 a 12:00
Circuito 1	00:00 a 12:00
Circuito 2	00:00 a 12:00
Circuito 3	00:00 a 12:00
Circuito 4	12:00 a 24:00
Circuito 5	12:00 a 24:00
Circuito 6	12:00 a 24:00
Circuito 7	12:00 a 24:00

El 21 de agosto correspondía a un lunes, y como se esperaba, la remota ignoro la rutina semanal de ese día y utilizo los eventos programados para el día especial. La figura 28 muestra el un ejemplo control horario durante el día especial.

Figura 28 Ejemplos del control horario durante el día especial

Con el control horario funcionando correctamente, solo restaba probar la función 5 de Modbus para forzar las salidas de la remota. Escribiendo en los registros para las bobinas virtuales, se le ordeno a la remota que active todas las salidas, tal como se muestra en la figura 29. Independientemente de la programación horaria, las salidas permanecieron activas hasta la media noche (hora 00:00:00) donde el cambia horaria reseteo los circuitos de salida.

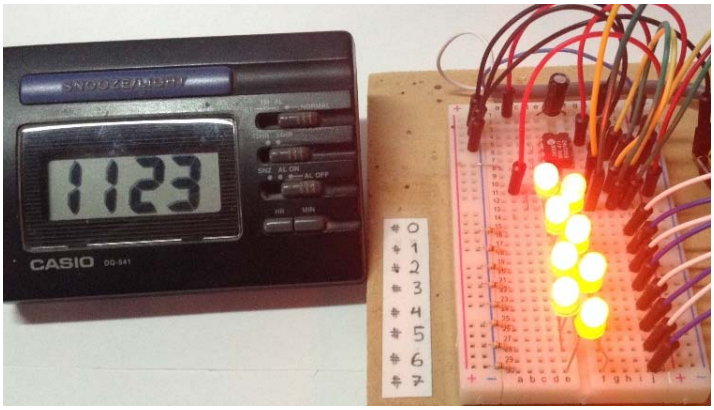


Figura 29 Todos los circuitos activos mediante la función 5 de Modbus

Totas las pruebas anteriores se realizaron con la remota desconectada de la red Modbus, solo fue necesario emplear el *Modbus Poll* para introducir los horarios o al forzar las salidas. Por lo tanto, el control horario es independiente del maestro del sistema de control centralizado. La figura 30 muestra la remota en funcionamiento estando desconectada de la red Modbus.

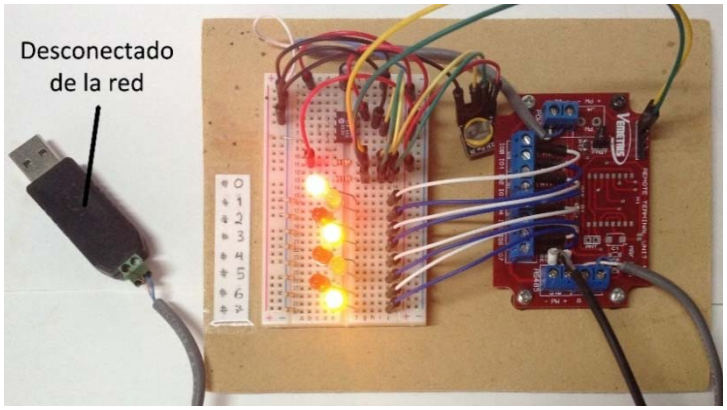


Figura 30 Remota en funcionamiento sin conexión

CONCLUCIONES Y RECOMEDACIONES

En función de los resultados obtenidos, basado en el desarrollo del presente trabajo y los objetivos planeados, se pueden presentar las siguientes conclusiones:

- El prototipo desarrollado en este trabajo permitió validar el concepto del tipo de dispositivo que se deseaba: un actuador remoto reprogramable, a través de una red externa, capaz de realizar un control (ON/OFF) de forma autónoma, basándose en la información de unas tablas horarias.
- Aunque los sistemas de control horario existan ya desde hace tiempo, lo novedoso de este dispositivo es que puede ser reprogramado a distancia a través de una red Modbus, por lo tanto, no se requiere de un operador en campo cada vez que dese agregar o editar algún horario.
- La función de días especiales (o festivos) le da un agregado que ningún otro dispositivo en el mercado posee, la posibilidad de tener eventos extraordinarios que ocurran si tener que modificar los horarios programados para los días de la semana. Además, la capacidad de forzar salidas lo convierte en un dispositivo ideal para la domótica.
- La aplicación principal del dispositivo es para implementaciones de domótica en las que se desee un manejo automático de los servicios. Como ejemplo podemos mencionar una edificación con aulas de clase, en donde se programaría el dispositivo con los horarios de clases para activar las luces de los salones. Otro ejemplo factible podría ser en un hotel, donde se fijen horarios para el encendido de luces externas durante la noche, calefacción en piscinas en las horas del día más frías, etc. Como estas las aplicaciones son bastante extensas.

En base a la experiencia adquirida durante el desarrollo del proyecto, se proponen las siguientes recomendaciones con el fin de mejorar el funcionamiento del equipo en futuras aplicaciones:

- Pueden sustituirse los relés electromecánicos por relés de estado sólido. Las ventajas de estos sobre estos sobre otro tipo de relés es que su integrado es de menor tamaño, su accionamiento casi instantáneo, no son susceptibles al desgaste mecánico y no producen ondas electromagnéticas que puedan interferir con otros equipos.
- Emplear una memoria EEPROM de al menos 128 kBytes. Esto permitiría almacenar los eventos de 365 días especiales, de este modo se podría asignar una programación horaria diferente para cada día del año, si así se desea.
- Implementar un registro Modbus para modificar la velocidad de trasmisión de datos con la red externa, de este modo la remota podría adaptarse distintas redes.
- Implementar un registro Modbus para resetear (colocar de fábrica) la remota: borrar la información de los horarios de la memoria y colocar los valores predeterminados.

REFERENCIAS BIBLIOGRAFICAS

[1] J. C. Bravo, "DISEÑO Y FABRICACION DE SISTEMA EMBEBIDO PARA LA AUTOMATIZACION DEL TIMBRE DE TERRAZA Y PLANTA DE LA INDUSTRIA ELECTRONICA ORINOQUIA" [online]. Caracas: Universidad Simón Bolívar, 2015. Disponible: http://www.bib.usb.ve/tesis/000171162.pdf

[2] MONTROL, Super Brain página web [online] Disponible: http://www.montrol-e.com/es/home/38-superbrain.html

[3] Manual de usuario bus I2C: I2C-bus specification and user manual, NXP/Philips Semiconductors, 2014 [online] Disponible: https://www.nxp.com/docs/en/user-guide/UM10204.pdf

[4] Manual de referencia Modbus Poll: Modbus Serie Protocol description, Witte Software, 2017 [online] Disponible: http://www.modbustools.com/modbus.html

[5] Manual de referencia PIC24FJ64GB004: 28/44-Pin General Purpose 16-Bit Flash Microcontrollers, Microchip Technology Inc., 2013 [online] Disponible: http://www.microchip.com/wwwproducts/en/PIC24FJ64GB004

[6] MPLAB® X Integrated Development Environment (IDE), Microchip Technology Inc., 2017 [online] Disponible: http://www.microchip.com/mplab/

[7] Manual de referencia DS3231: Extremely Accurate I2C-Integrated RTC/TCXO/Crystal, Maxim Integrated Products Inc., 2015 [online] Disponible: https://datasheets.maximintegrated.com/en/ds/DS3232.pdf

[8] Manual de referencia 24LC256: 256K I2C™CMOS Serial EEPROM, Microchip Tecnología Inc., 1998 [online] Disponible: http://ww1.microchip.com/downloads/en/DeviceDoc/21203M.pdf

[9] Manual de referencia MAX487: Low-Power Slew-Rate-Limited RS-485/RS-422 Transceivers, Maxim Integrated Products, Inc., 2014 [online] Disponible: https://datasheets.maximintegrated.com/en/ds/MAX1487-MAX491.pdf

GLOSARIO

ACK: Acuse de recibo (*acknowledgement*).

Arduino: Compañía de hardware libre que diseña y manufactura placas de desarrollo de hardware, compuestas por Microcontroladores.

ASCII: Código Estándar Estadounidense para el Intercambio de Información (*American Standard Code for Information Interchange*).

Baudios: Número de símbolos por segundo en un medio de transmisión digital.

Bit: Acrónimo para Digito Binario (***Bi****nary digi****t***).

Byte: Equivale a 8 Bits.

CLK: Señal de reloj (*Clock*).

CRC: Verificación por redundancia cíclica (*Cyclic Redundancy Check*)

EEPROM: Eléctricamente programable y borrable memoria de sólo lectura (*Electrically Erasable Programmable Read-Only Memory*).

GSM: Sistema global para las comunicaciones móviles (*inglés Global System for Mobile communications*).

Hardware: Partes físicas tangibles de un sistema informático

HMI: Interfaz de usuario (*Human Machine Interface*).

NACK: Acuse de recibo negativo (*Negative Acknowledgement*).

RAM: Memoria de acceso aleatorio (*Random Access Memory*).

ROM: memoria de sólo lectura (*Read-Only Memory*).

RTC: Reloj en tiempo real (*Real Time Clock*).

RTU: (*Remote Terminal Unit*). En Modbus se utiliza para identificar la version binaria del protocolo

SDA: Señal de datos serie (*Serial Data*).

Software: soporte lógico de un sistema informático.

SPI: Interfaz periférica serial (*Serial Peripheral Interface*).

USB: Bus Universal en Serie (*Universal Serial Bus*).

Wi-Fi: ecanismo de conexión de dispositivos electrónicos de forma inalámbrica.