SECRETARIA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ

INGENIERÍA ELECTRÓNICA

INFORME TÉCNICO DE RESIDENCIA PROFESIONAL

Nombre del proyecto:
"REDISEÑO Y CONSTRUCCIÓN DEL RELOJ DEL EDIFICIO PRINCIPAL DE LA ESCUELA SECUNDARIA DEL ESTADO"

Asesor:
M.C. JOSÉ ÁNGEL ZEPEDA HERNÁNDEZ

Presentan:
HERMOSILLA MOHA CRUZ DANIEL
LÓPEZ LÓPEZ GLENDER GUADALUPE

Periodo de realización:
AGOSTO- DICIEMBRE 2015

TUXTLA GUTIÉRREZ CHIAPAS, DICIEMBRE 2015

CONTENIDO

CAPITULO I
INTRODUCCIÓN

Antecedentes


FIGURA 1. FUNDACIÓN ESE

El ICACH fue fundado el 15 de mayo de 1944, por el Dr. Rafael Pascasio Gamboa, Siendo en aquel entonces una escuela mixta y además de preparatoria del Estado. En el año de 1945, por decreto del Gobernador Juan M. Esponda, se establece como el Instituto de Ciencias y Artes de Chiapas (ICACH), que reúne a las escuelas secundaria, preparatoria y normal. Su nombre era "Escuela Secundaria, Preparatoria y Normal del Instituto de Ciencias y Artes de Chiapas", ya que la misma instalación albergaba en su interior los tres niveles educativos, lo que la hacía la única escuela que ofrecía estudios del nivel Medio superior y superior, denominándosele así hasta 1965; en ese año se divide el área de preparatoria y normal trasladándolas a otro lugar, quedando sólo la secundaria.
Surgen tres nuevas instituciones educativas: Escuela Secundaria del Estado del ICACH, Preparatoria del Estado y Normal del Estado del ICACH.
La Escuela Secundaria del Estado funcionaba en sus turnos matutino y nocturno bajo el mando de un solo director, y es en el año de 1996 cuando a través de un concurso escalafonario la Secretaría de Educación determinó nombrar a un Director para cada turno, con su respectiva plantilla de personal, designando como Directora del turno vespertino la Profesora Josefa Eloína Ocampo Santiago, quien a la fecha sigue al frente de la misma.

En el año de 1944, es decir en la fundación de la escuela, se colocó un reloj en el edificio principal de esta institución, convirtiéndose de inmediato en un símbolo de representativo. Así pues el emblemático reloj emitía melodías provenientes de unas campanas que eran escuchadas por todos los vecinos cercanos al lugar. El funcionamiento del reloj era mecánico es decir, que era de cuerda.

Actualmente el reloj de la secundaria no funciona desde hace aproximadamente 50 años, y el tiempo ha mermado sobre él y su estructura, dejando solo una imagen de un reloj viejo.

En base a la reseña anterior, se decidió hacer el rediseño del reloj de la escuela secundaria del estado como proyecto de residencia para que el reloj tenga un nuevo funcionamiento, más eficiente al anterior.


FIGURA 2. RELOJ DESABILITADO

El nuevo diseño consistirá en quitar el sistema de cuerdas con el que trabajaba anteriormente, y colocar un motor en el eje del reloj que se encargará de mover las mancillas, el motor será activado cada minuto. Debido a que tenemos que llevar un control de tiempos se implementará un RTC (real time clock o reloj en tiempo real), que se encargara de llevar un conteo de las horas y minutos transcurridos. Para poder saber si el motor ya avanzó el equivalente a un minuto se va a hacer uso de un encoder, el cual medirá los grados avanzados de un motor y los cuales servirán para poder apagar el motor una vez posicionado.

Para poder saber si el RTC tiene la hora y minutos exactos, es decir, el tiempo de ese instante, se diseñará una página de internet a nivel (LAN), para que el usuario ajuste de forma automática la hora del reloj en caso de desajuste. También dicha página contará con un apartado para el administrador, donde se podrán hacer configuraciones del RTC, así como monitorear el estado de los equipos.

Con una uSD (tarjeta micro SD) se almacenaran tonos y canciones representativas de Chiapas que se reproducirán por medio de una shield Mp3 hacia un amplificador dándole así más realce al reloj y por ende a la Escuela Secundaria del Estado.

En el año 2004, el Instituto Tecnológico de Tuxtla Gutiérrez, fue parte del proyecto encargado de diseñar un nuevo sistema electrónico capaz de grabar y reproducir canciones digitalizadas provenientes de una computadora para el reloj de la Catedral de San Marcos.

JUSTIFICACIÓN

El reloj que se encuentra ubicado en la Escuela Secundaria del Estado, se rediseñará para que funcione con un sistema Mecatrónico el cual es una mejor opción que un sistema de cuerda.

El Rediseño del Reloj tendrá las siguientes características:

Contará con un RTC (real time clock), que será el encargado de llevar el conteo del tiempo con el fin de ir posicionando las manecillas en el lugar que les corresponde en función a las horas y minutos.

En dado caso de haber ausencia de corriente eléctrica, el RTC será capaz de seguir llevando el conteo de las horas, minutos y segundos con el fin de que cuando la corriente se restablezca podamos llevar las manecillas del reloj (tiempo anterior) a la posición tiempo actual que tiene el real time clock (RTC).

Tiene una página de internet a nivel (LAN) para que desde ahí se haga el reajuste de tiempo. Se extraerá la hora y los minutos de la computadora para poder ajustar el tiempo del RTC en dado caso que se atrase o adelante el conteo del tiempo.

La página de internet cuenta con una opción de administrador donde se monitoreara el estado de los equipos, así como también el estado del motor que se encargara de mover las manecillas. Lo anterior se usara con el fin de poder revisar el estado del reloj y evitar así posibles daños. Dicha opción usa una contraseña.

Cuenta con una contraseña para acceder a la página de usuario del reloj, para tener un control de seguridad evitando que cualquier persona diferente a la directora haga configuraciones en el sistema del reloj.

Se instalarán bocinas tipo trompeta para reproducir cada hora tonos de campanas y canciones representativas de Chiapas.

OBJETIVOS

OBJETIVO GENERAL

Rediseñar el Sistema de Control del Reloj del edificio principal de la escuela secundaria del estado

OBJETIVOS ESPECÍFICOS

- Diseñar el sistema de control del reloj para el movimiento de las manecillas
- Realizar ajuste de la hora y los minutos del RTC mediante la página de internet.
- Crear una página web para el control del reloj. Una a nivel de usuario y una a nivel Administrador.
- Diseñar el sistema de control para reproducir tonos de campana de acuerdo a la hora y canciones representativas de Chiapas.
- Diseñar un Sistema de seguridad para modo usuario y administrador, con contraseñas independientes.
- Diseñar un Sistema de Monitoreo del estado de los equipos (Fuente de voltaje DC, nivel de voltaje AC, Estado del motor) por medio de una página de internet en el modo administrador.

CARACTERIZACIÓN DEL ÁREA EN QUE SE PARTICIPÓ

En la Escuela Secundaria del Estado se encuentra el reloj a restaurar, esta institución está ubicada en la dirección Segunda Sur Oriente 620, col. Centro, 29000 Tuxtla Gutiérrez, Chis.

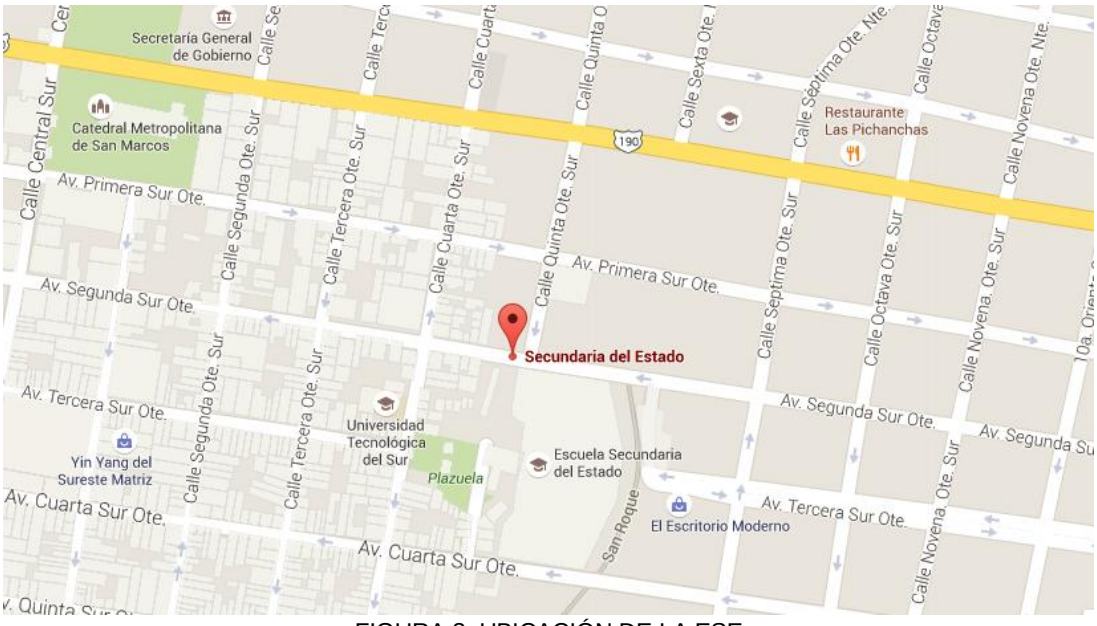


FIGURA 3. UBICACIÓN DE LA ESE

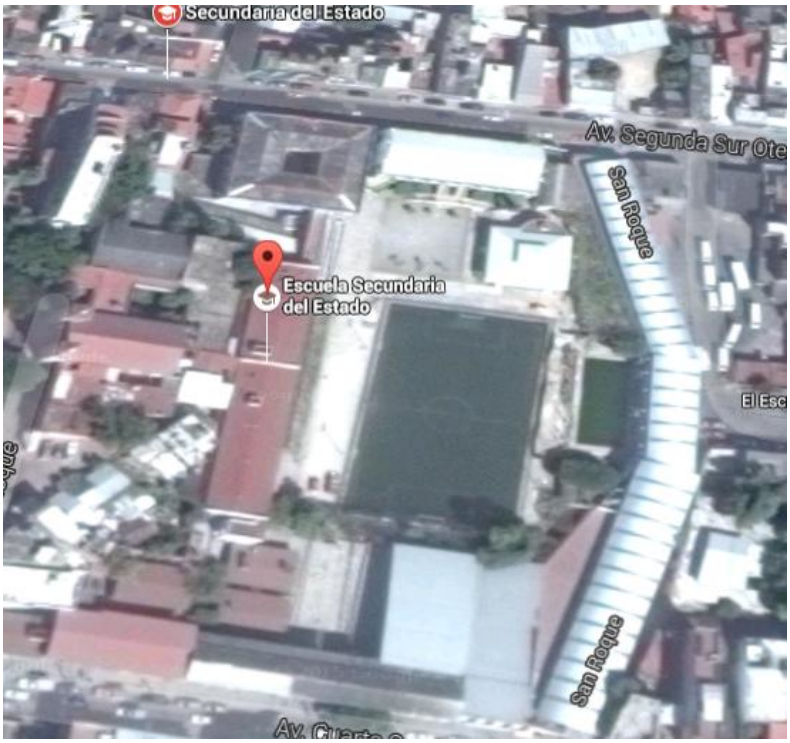


FIGURA 4. UBICACION DE LA ESE

Como se puede observar en la siguiente imagen, el reloj esta ubicado en el edificio principal de la institución, en la parte superor de éste

FIGURA 5. EDIFICIO DONDE SE ENCUENTRA EL RELOJ

PLANTEAMIENTO DEL PROBLEMA

El Sistema de Control del Reloj de la Escuela Secundaria del Estado basaba su funcionamiento en un sistema de cuerdas. Esto quiere decir que todo el reloj era completamente mecánico. Teniendo como desventajas.

Si el reloj sufría un desajuste tiempo, (atraso o adelanto de horas y minutos), había que subir hasta donde esta ubicado el reloj y mover una palanca que realizaba la calibración del tiempo. Cabe señalar que el acceso al reloj no es muy seguro poniendo en peligro la integridad del individuo que subía a realizar la calibración del reloj.

Se contaba con un sistema mecánico que realizaba el toque de las campanas, no habiendo variedad en los tonos.

El reloj estaba a la intemperie provocando la oxidación de todas las piezas y a falta de mantenimiento éste quedo fuera de servicio.

Debido a que el reloj no contaba con luminaria, por las noches no se podía visualizar la hora.

Al fallar alguna pieza del sistema mecánico del reloj, el reemplazo de esta pieza resultaba algo difícil y tedioso además de ser costoso.

CAPITULO II
FUNDAMENTO TEÓRICO

PLACA ARDUINO MEGA 2560

La tarjeta Arduino Mega 2560 es una placa electrónica basada en el Atmega2560. Cuenta con 54 pines digitales de entrada / salida (de los cuales 15 se pueden utilizar como salidas PWM), 16 entradas analógicas, 4 UARTs (hardware puertos serie), un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP, y un botón de reinicio. Contiene todo lo necesario para apoyar el micro controlador; simplemente se conecta a un ordenador con un cable USB o con un adaptador de CA o la batería a CC para empezar. La tarjeta Mega 2560 es compatible con la mayoría de las shields diseñadas para el arduino Uno y los antiguos tableros Duemilanove o Diecimila.

Programación

La tarjeta Mega se puede programar con el software de Arduino (IDE). Los Atmega2560 en la Mega 2560 vienen pre-programado con un gestor de arranque que le permite cargar nuevo código a él sin el uso de un programador de hardware externo. Se comunica mediante el protocolo original STK500 (referencia, archivos de cabecera C).
También puede pasar por alto el gestor de arranque y programar el microcontrolador a través del ICSP (In-Circuit Serial Programming) utilizando ISP del arduino o similar.

Memoria

El Atmega2560 tiene 256 KB de memoria flash para almacenar el código (de los cuales 8 KB se utiliza para el cargador de arranque), 8 KB de SRAM y 4 KB de EEPROM (que puede ser leído y escrito con la biblioteca EEPROM).

Comunicación

Una biblioteca SoftwareSerial permite la comunicación en serie en cualquiera de los pines digitales del Mega 2560. El Mega 2560 también soporta la comunicación TWI y SPI.

SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). Estos pines soportan la comunicación SPI utilizando la biblioteca de SPI. Los pines SPI también se desglosan en la cabecera ICSP, que es físicamente compatible con el Arduino Genuino Uno y los viejos tableros Duemilanove y Diecimila Arduino.

FICHA TÉCNICA

Microcontroladores	Atmega 2560
Tensión de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (limite)	6-20V
Digital pines I/O	54 (de las cuales 15 proporcionan salida PWM)
Pines de entrada analógica	16
Corriente DC por E/S Pin	20 Ma
Corriente DC de 3.3V Pin	50 Ma
Memoria Flash	256 KB de los cuales 8 KB utilizado por el gestor de arranque
SRAM	8 KB
EEPROM	4 KB
Velocidad de reloj	16 MHz
Largo	101.52 mm
Anchura	53.3 mm
Peso	37 g

TABLA 1. FICHA TÉCNICA DEL ARDUINO MEGA

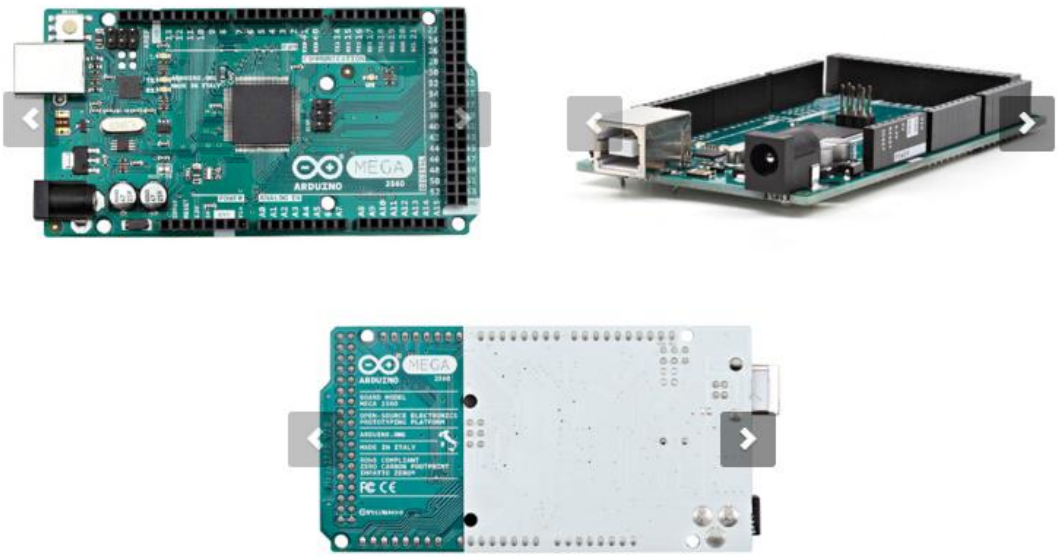


FIGURA 6. PLACA ARDUINO

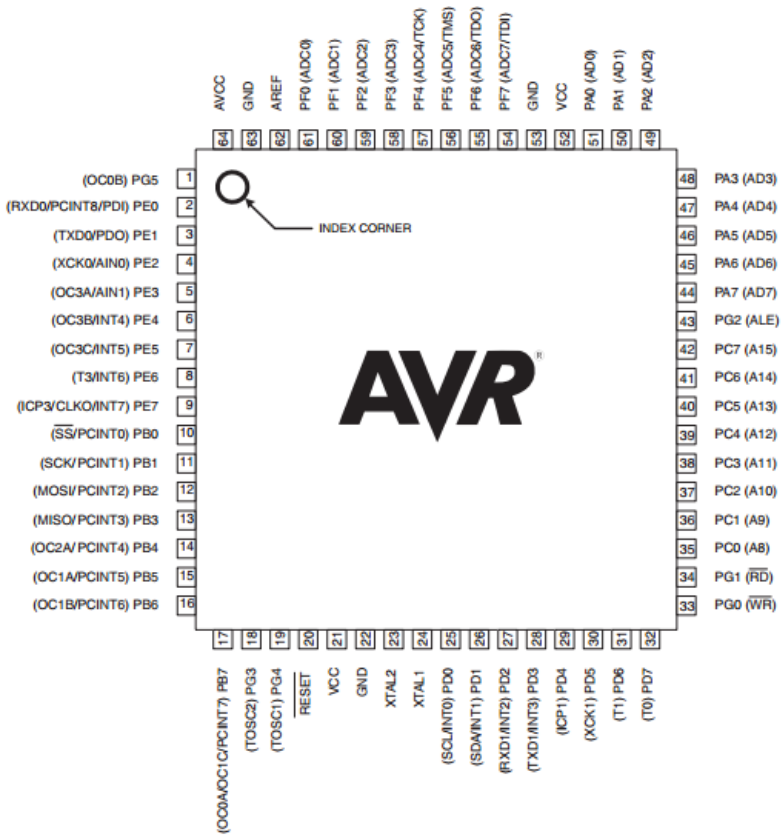


FIGURA 7. MICROCONTROLADOR ATMEGA 2560

SPARKFUN MP3 PLAYER SHIELD

La SparkFun MP3 Player Shield es un decodificador MP3 impresionante con las capacidades de almacenamiento de archivos de música en una tarjeta microSD run-of-the-mill, lo que da la capacidad toadd músic o efectos de sonido a cualquier proyecto. Con esta tarjeta se pueden arrastrar archivos MP3 desde una tarjeta microSD y reproducirlo con sólo una shield, convirtiendo cualquier Arduino en un reproductor de MP3 independiente totalmente funcional. La MP3 Shield utiliza el VS1053B decodificador de audio MP3 IC para decodificar los archivos de audio. El VS1053 también es capaz de decodificar Ogg Vorbis / MP3 / AAC / WMA / audio MIDI y codificación IMA ADPCM.

La pieza central de la shield MP3 Player es un VS1053B Audio Codec IC. El VS1053B es un pequeño chip polifacético.

El VS1053 tiene un zócalo para la tarjeta µSD, que se puede utilizar para almacenar archivos MP3. El uso de la Arduino biblioteca SD es fácil, lee archivos de música de una tarjeta SD, y transmitir a la VS1053B. Hay circuitos adicionales de acuerdo a las señales de cambio de nivel hasta el máximo permitido por 3.3V tarjetas SD.

El VS1053B recibe su flujo de bits de entrada a través de un bus de entrada en serie (SPI). Después que la corriente ha sido decodificada por el CI, el audio se envía a la vez un conector para auriculares estéreo de 3,5 mm, así como un header de 2 pines 0,1". VS1053b de VLSI es un solo chip Ogg Vorbis / MP3 / AAC / WMA / MIDI decodificador de audio y un ADPCM IMA y Ogg Vorbis codificador-usuario cargable. Contiene un alto rendimiento, bajo consumo de energía patentada núcleo del procesador DSP VS DSP4, trabajando la memoria de datos, 16 instrucción KB RAM and 0.5+ RAM de datos KiB para aplicaciones de usuario que se ejecutan simultáneamente con cualquier decodificador incorporado, interfaces de control de serie y datos de entrada, hasta 8 de propósito general pines I / O, una UART, así como un ADC de alta calidad estéreo tasa variable muestra (micrófono, línea, línea + micrófono o 2 × línea) y DAC estéreo, seguido de un amplificador de auriculares y un voltaje común amortiguador. VS1053b recibe su flujo de bits de entrada a través de un bus de entrada de serie, que se escucha como un esclavo del sistema. El flujo de entrada se decodifica y se pasa a través de un control de volumen digital a una de 18 bits sobre-muestreo, de varios bits, sigma-delta DAC. La decodificación se controla a través de un bus de control serial.

He aquí un rápido resumen visual de los conectores importantes y otros componentes en la shield MP3 player:

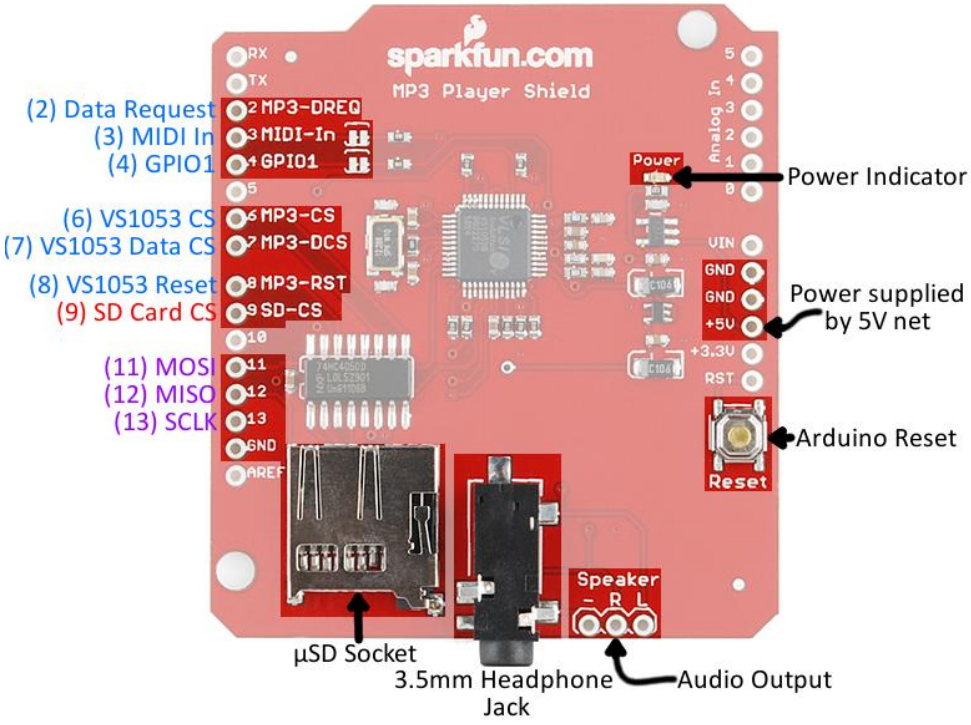


FIGURA 8. SHIELD MP3 PLAYER

En la imagen anterior, las etiquetas azules son pines utilizados por el VS1053 MP3 Codec IC, las etiquetas rojas se utilizan para la comunicación con la tarjeta µSD, y las etiquetas de color púrpura son utilizadas para la comunicación SPI.

La Mp3 Player Shield requiere el uso exclusivo de un grupo de pines. Estos pines pueden ser usados para la interfaz con otros dispositivos

- **D2** está conectado a la **solicitud de datos** (**data request)** de salida de la VS1053B. Este pin es una interrupción, que cuenta al Arduino que el CI necesita más datos de música.
- **D6** está conectado a la **selección de chip (chip select)** de entrada del VS1053B. Esto activa a bajo el pin dice al chip cuando los datos se envían a la misma.
- **D7** se conecta a la **selección de chip de datos (data chip select)** de entrada de la VS1053B, que narra al chip cuando se está enviando datos de música.
- **D8** está conectado a la **re inicialización (reset)** de entrada de la VS1053B.
- **D9** está conectado a la **selección de chip** (**chip select)** entrada de la tarjeta µSD.

Son tres pines de SPI para arduino correspondientes a datos y reloj, D11, D12, D13. Se pueden utilizar para interactuar con otros componentes de SPI. No pueden, sin embargo, ser utilizados para fines distintos a la SPI.

La shield utiliza hasta un buen número de pines, los pines disponibles para conectarse a otros componentes son:

- Los pines UART hardware - **RX** y **TX** - en los pines 0 y 1
- **D5** y **D10** (pines PWM!)
- Todos los pines analógicos **(A0 a** A5).

Características:

- 3.5mm Jack de salida de audio
- 0.1 " espaciados de headers de salida de altavoz
- ranura para tarjetas microSD

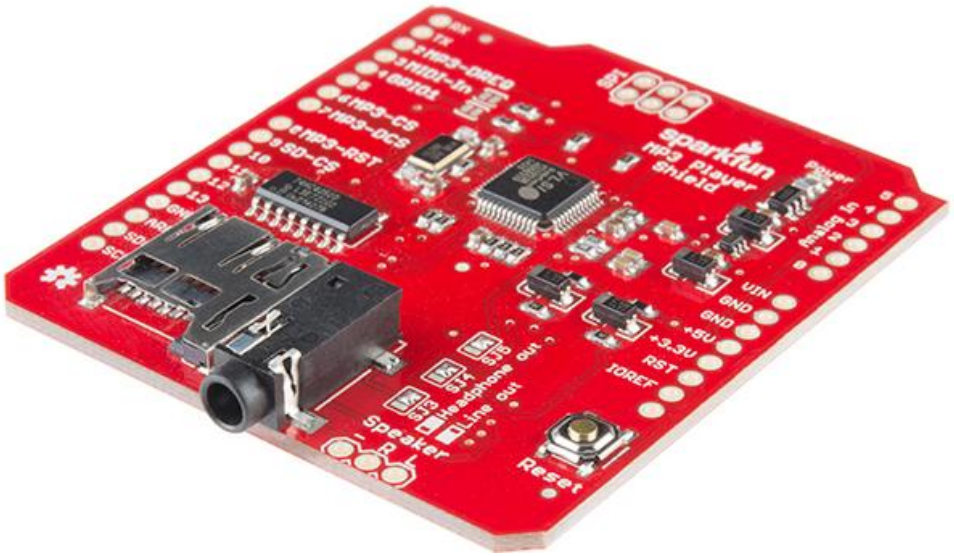


FIGURA 9. SHIELD MP3 PLAYER

ARDUINO ETHERNET SHIELD 2

Descripción general

La Arduino Ethernet shield 2 conecta el Arduino a Internet en cuestión de minutos. Sólo se tiene que conectar este módulo en la placa Arduino, conectarlo a una red con un cable RJ45 y seguir unos pasos sencillos para empezar a controlar el mundo a través de internet.

Descripción

La Arduino Ethernet shield 2 permite a una placa Arduino conectarse a internet. Se basa en el chip Wiznet W5500 Ethernet. El Wiznet W5500 ofrece una red (IP) capaz de usar TCP y UDP. Soporta hasta ocho conexiones de socket simultáneas. Utiliza la biblioteca de Ethernet para escribir sketches que se conectan a Internet a través de la shield Ethernet. La shield Ethernet 2 se conecta a una placa Arduino usando heades wire-wrap que se extienden por toda ella. Esto mantiene la disposición de las clavijas intacto y permite que otras shields se apilen en la parte superior de la misma.

FIGURA 10. CHIP W5500

Hay una ranura para tarjetas micro-SD a bordo, que se puede utilizar para almacenar archivos y leerlos a través de la red. Es compatible con el Arduino Uno y Mega (utilizando la librería Ethernet). El lector de tarjetas micro-SD de a bordo es accesible a través de la Biblioteca SD. Cuando se trabaja con esta biblioteca, SS es el Pin 4. La shield también incluye un controlador de reajuste, para asegurar que el módulo Ethernet W5500 se restablece correctamente en el encendido.

Arduino se comunica tanto con el W5500 y la tarjeta SD usando el bus SPI (a través de la cabecera ICSP). Esto es en los pines digitales 10, 11, 12, y 13 en el Uno y los pines 50, 51 y 52 en el Mega. En ambas tarjetas, el pin 10 se utiliza para seleccionar el W5500 y el pin 4 de la tarjeta SD. Estos pines no se pueden utilizar para general I / O. En los Mega, el pin SS hardware, es el 53, no se utiliza para seleccionar el W5500 o la tarjeta SD, pero debe mantenerse como la interfaz SPI o no funcionará de salida.

Debido a que la tarjeta comunica a ambos el W5500 y la SD, sólo una a la vez puede estar activo. Si se utiliza ambos periféricos en el programa, debe ser atendido por las bibliotecas correspondientes. Si no utiliza uno de los periféricos en su programa, sin embargo, tendrá que anular la selección explícitamente. Para hacer esto con la tarjeta SD, ajuste el pin 4 como salida y escribir un alto a la misma. Para el W5500, establezca pin digital 10 como una salida de alta.

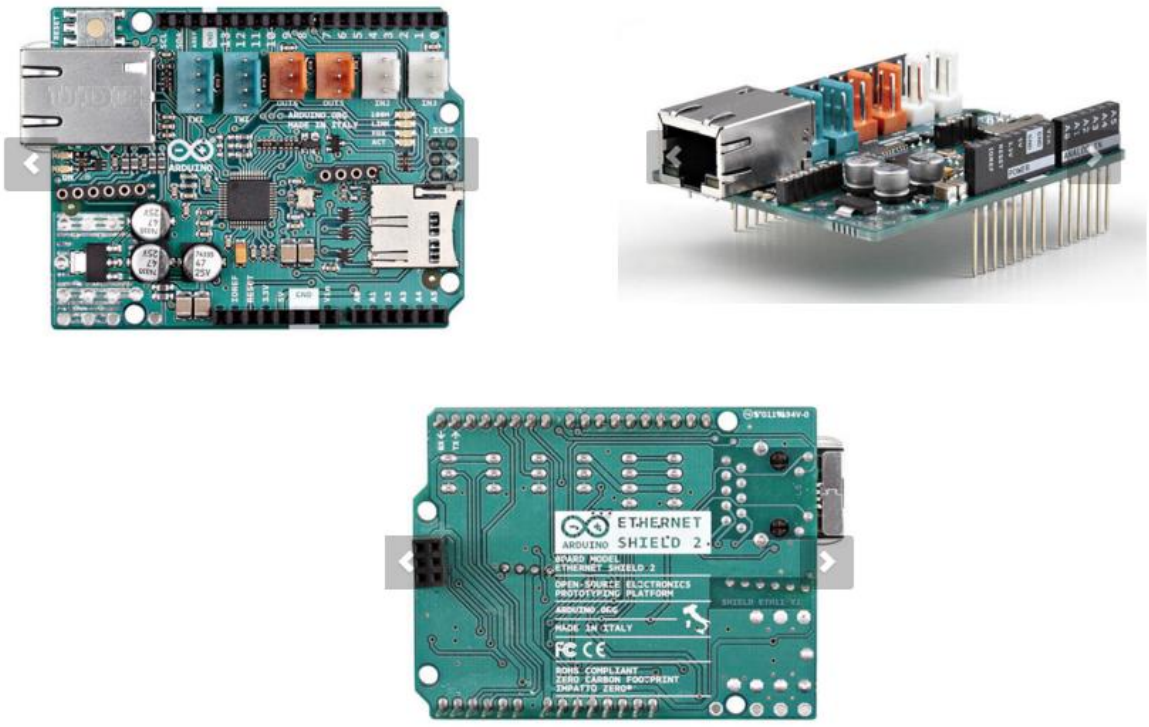


FIGURA 11. ARDUINO ETHERNET SHIELD

RTC DS1307

El RTC (Real Time Clock) Reloj en Tiempo Real, es un circuito electrónico especializado cuya función es mantener la hora y fecha actual en un sistema informático (ya sea con microcontrolador u otro tipo de CPU). Se caracteriza por tener un bajo consumo de energía y también normalmente su propia fuente de alimentación auxiliar. Normalmente al recurrir a este tipo de circuitos integrados obtenemos una mejor precisión del tiempo. Un ejemplo de dispositivos que incluyen relojes en tiempo real son las computadoras personales (PC).

Bus I2C, Es el protocolo de comunicación físico mediante el cual se comunican el Arduino y el módulo RTC DS1307. El bus cuenta con dos líneas: de datos y de reloj, ambas del tipo colector abierto (o drenador abierto). Por lo que se requieren resistencias pull-up, para generar un estado lógico alto. La figura siguiente muestra una conexión básica del DS1307 con un microcontrolador a través del bus I2C.

El DS1307, es un reloj de tiempo real (RTC) de baja potencia, utiliza decimal codificado en binario completo (BCD) reloj / calendario más 56 bytes de SRAM NV. La dirección y datos se transfieren en serie a través de un I2C, bus bidireccional. El reloj / calendario proporciona segundos, minutos, horas, día, fecha, mes, año y la información. El final de la fecha de mes se ajusta automáticamente para el mes con menos de 31 días, incluyendo las correcciones de salto de año. El reloj funciona en el formato de 24 horas o de 12 horas con indicador AM / PM. El DS1307 tiene un integrado en el circuito de potencia que detecta fallas de energía y cambia automáticamente a la alimentación de reserva. La operación de hora normal continúa mientras que la parte opera desde el suministro de copia de seguridad.

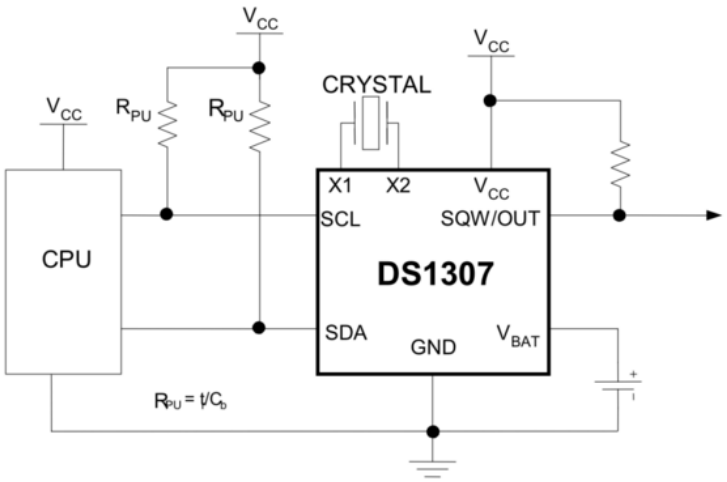


FIGURA 12. CIRCUITO TÍPICO DE OPERACIÓN

Decimal codificado en binario (BCD), es un estándar de datos en el cual los dígitos decimales se representan mediante secuencias de 4 bits. Esto es así porque con 4 bits se pueden representar todos los dígitos decimales del 0 al 9. En un byte en memoria se pueden almacenar números decimales del 0 al 99. En el DS1307 los datos de fecha y hora están codificados en BCD. A continuación se muestran los dígitos decimales y sus equivalentes en BCD.

Decimal:	0	1	2	3	4	5	6	7	8	9
BCD:	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

FIGURA 13. DÍGITOS DECIMALES Y SUS EQUIVALENTES

Funcionamiento interno y registros del DS1307

- El DS1307 requiere un cristal de 32.768 KHz, este valor viene dado por el hecho de que 2^{15} = 32,768. Esto quiere decir que la frecuencia es divisible binariamente para generar un segundo exacto. El cristal ya se incluye si compramos algún módulo con el DS1307.

- El DS1307 requiere dos fuentes de alimentación: Por una parte, requiere alimentación de 5 volts que opera mientras el circuito esta encendido y funcionando y otra fuente de poder que proviene de una batería de litio (tipo reloj) que mantiene funcionando el reloj/calendario mientras la alimentación principal NO está disponible. El cambio entre ambas fuentes de alimentación es gestionado por el DS1307 de manera automática.

- Disponemos de un pin de salida que puede ser configurado para que el RTC entregue una onda cuadrada con una frecuencia configurable, las frecuencias disponibles se muestran en la siguiente tabla (Tabla 2) y se configuran mediante los bits RS1, RS0 y SQWE de registro de control, si se usa este pin hay que agregar una resistencia pull-up, ya que es del tipo “Open drain”.

RS1	RS0	SQW/OUT OUTPUT	SQWE	OUT
0	0	1 Hz	1	x
0	1	4.096 kHz	1	x
1	0	8.192 kHz	1	x
1	1	32.768 kHz	1	x
X	X	0	0	0
X	X	1	0	1

TABLA 2. FRECUENCIAS DISPONIBLES

Los registros del DS1307 almacenan la fecha y la hora en formato BCD. La dirección de cada registro y la información almacenada en cada uno se muestra en la siguiente imagen:

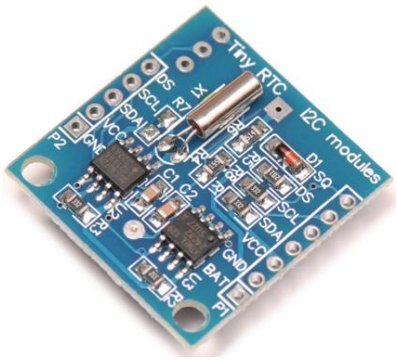
ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/ AM							
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

FIGURA 14. DIRECCIÓN DE CADA REGISTRO E INFORMACIÓN ALMACENADA EN CADA UNO

El funcionamiento del chip se controla mediante el bit 7 del registro del segundero (0x00) y el registro de control (0x07): El bit CH del segundero detiene el reloj cuando esta en alto (así se entregan los módulos de fábrica) y en este modo NO se lleva el conteo de tiempo porque el oscilador está detenido. ES MUY IMPORTANTE PONER A 0 ESTE BIT PARA QUE EL RTC FUNCIONE. El registro de control maneja la funcionalidad del pin de salida de onda cuadrada.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OUT	0	0	SQWE	0	0	RS1	RS0

TABLA 4. REGISTRO DE BITS



A partir de la dirección 0x08 disponemos de memoria RAM no volátil (NVRAM) cuyo contenido no se borra con la pérdida de energía, este segmento de memoria funciona de manera similar a una memoria EEPROM.

FIGURA 15. MODULO RTC

MICRO SD

Una tarjeta Micro SD (Secure Digital) es una tarjeta de memoria para almacenar contenidos en dispositivos portátiles, como teléfonos móviles, cámaras digitales, tablets o navegadores GPS. Las tarjetas SD son uno de los sistemas más populares de almacenamiento de gran cantidad de información en pequeño tamaño.

Una micro SD el formato más habitual en los dispositivos pequeños como teléfonos móviles o tablets. Su tamaño es de 15 mm de alto x 11 mm de ancho x 10 mm de grosor.

Los aparatos donde el usuario estará quitando y poniendo la tarjeta como las cámaras digitales o consolas de videojuegos, suelen usar tarjetas SD de tamaño estándar. Los dispositivos donde el tamaño se reduce como los teléfonos móviles o tablets, suelen usar tarjetas micro SD sin adaptador.

Tipos y capacidad de almacenamiento

En lo que respecta a la capacidad de almacenamiento de datos, las tarjetas se clasifican también en 3 tipos que encontrarás especificados en la propia tarjeta:

- **SD SC** (Standard Capacity) o simplemente SD: con capacidad para almacenar hasta 2GB de datos.
- **SD HC** (High Capacity): permiten guardar hasta 32GB.
- **SD XC** (extended Capacity): pueden almacenar hasta 2TB (2.000GB)

Hoy en día una tarjeta de 16GB suele ser suficiente para la mayoría de usos, aunque mucha gente opta por comprar tarjetas de 32GB o 64GB para ampliar la memoria de sus dispositivos.

Velocidad y clases

CLASE

La velocidad a la que una tarjeta SD guarda los datos viene definida por su clase. Para dispositivos como cámaras digitales sencillas realmente no importa la clase de la tarjeta, tan sólo que sea compatible. Guardar ficheros de 1 o 2MB no es difícil. Sin embargo, si quieres guardar ficheros mayores, vídeos de alta definición o secuencias rápidas de fotos, la tarjeta SD debe ser lo suficientemente rápida almacenando datos.

- **Clase 2**: graba 2MB por segundo, lo que sería una foto normal.

- **Clase 4**: capaz de almacenar 4MB por segundo, el tamaño de un archivo MP3 con una canción.
- **Clase 6**: graba 6MB por segundo (hasta aquí fácil, ¿verdad?).
- **Clase 10**: graba a 10MB por segundo **o más rápido** (algunas pueden llegar a 90MB/segundo aunque muy pocos dispositivos necesitan esa velocidad).

La clase de una tarjeta indica la velocidad mínima a la que graba, no la velocidad real. Por ello, una buena tarjeta de clase 2 puede funcionar más rápido que una mala tarjeta de clase 6.

Velocidad de bus
Desafortunadamente no sólo la clase (la velocidad a la que graba) define la velocidad de la tarjeta SD. Existe otro factor que es la velocidad a la que la información se envía entre el dispositivo y la tarjeta SD. Ésa es la Velocidad de Bus.

Todas las tarjetas de clase 6 o menos utilizan un Bus Estándar, pero a partir de la clase 10 podemos encontrar el Bus de Alta Velocidad y los Buses de Ultra Alta Velocidad (Ultra High Speed) UHS-I y UHS-II.

Las tarjetas de clase 10 con Buses de Ultra Alta Velocidad pueden ser de clase U1 o clase U3. Este último garantiza velocidades de escritura mínimas de 30MB por segundo, destinado principalmente a la grabación de vídeos con resolución 4K.

CLASE	VELOCIDAD MÍNIMA	APLICACIONES
Clase 2	2 MB/s	Hacer fotos y grabar vídeos estándar
Clase 4	4 MB/s	Grabar vídeo de alta definición (HD) incluyendo Full HD (de 720p hasta 1080p/1080i)
Clase 6	6 MB/s	Grabar vídeo de alta definición (HD) incluyendo Full HD (de 720p hasta 1080p/1080i)
Clase 10	10 MB/s	Grabar video Full HD (1080p) y tomar fotos HD (Bus de Alta Velocidad)
UHS Clase 1 (U1)	10 MB/s	Grabación en tiempo real y vídeos largos de alta definición (Bus de Ultra Alta Velocidad)
UHS Clase 3 (U3)	30 MB/s	Archivos de vídeo de resolución 4K (Bus de Ultra Alta Velocidad)

FIGURA 16. CLASES DE LAS SD

PROTOCOLO I2C

El protocolo I2C, define las reglas de cómo podemos conectar diferentes dispositivos entre sí, desarrollado por Philips en la década de los 80/90, convirtiéndose en la actualidad como un estándar. I2C crea un bus de comunicación entre los diferentes dispositivos en serie, esto nos permite conectar hasta 1000 dispositivos uno detrás de otro. La comunicación siempre se realizara entre dos dispositivos, uno actuara de maestro, este es el que trasmitirá la señal para sincronizar la transferencia de datos, y el otro de esclavo. El que hace de maestro, no tiene por qué hacer esta función siempre, puede ir pasándose de uno a otro, aunque no todos los periféricos tienen esta funcionalidad, están diseñados para ser siempre esclavos.

Para identificar que un periférico a enviado la información, cada uno de ellos tiene una dirección única, igual que una red de ordenadores donde cada ordenador tiene su propia dirección.

El bus I2C consta de 3 líneas, SDA (datos), SCL (reloj) y GND (masa)

SDA: Es la línea por donde circulan los datos, formateada de la siguiente forma:

| start | A7 A6 A5 A4 A3 A2 A1 R/W | ACK | … DATA … | ACK | stop | idle |

SCL: Por esta línea va la señal para sincronizar las transferencias de datos.

GND o masa: Se utiliza como referencia de voltaje para el cálculo de las otras líneas.

Arduino cuenta la posibilidad de poder manejar comunicaciones I2C. Según el modelo de Arduino que tengamos, los pines de SDA y SCL, varían, En el caso del Arduino UNO los pines son: para SDA es el pin A4, y el SCL es el pin A5, para el LEONARD estos vienen marcados como tales, y en caso de MEGA el pin para el SDA es el pin digital 20 y para SCL es el pin digital 21.

Funcionamiento

La condición inicial, de bus libre, es cuando ambas señales están en estado lógico alto. En este estado cualquier dispositivo maestro puede ocuparlo, estableciendo la condición de inicio (start). Esta condición se presenta cuando un dispositivo maestro pone en estado bajo la línea de datos (SDA), pero dejando en alto la línea de reloj (SCL).

El primer byte que se transmite luego de la condición de inicio contiene siete bits que componen la dirección del dispositivo que se desea seleccionar, y un octavo bit que corresponde a la operación que se quiere realizar con él (lectura o escritura).

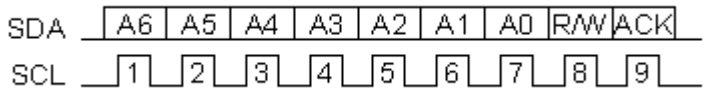


FIGURA 17. DIAGRAMA DE TIEMPO DE LA COMUNICACIÓN I2C

Si el dispositivo cuya dirección corresponde a la que se indica en los siete bits (A0-A6) está presente en el bus, éste contesta con un bit en bajo, ubicado inmediatamente luego del octavo bit que ha enviado el dispositivo maestro. Este bit de reconocimiento (ACK) en bajo le indica al dispositivo maestro que el esclavo reconoce la solicitud y está en condiciones de comunicarse. Aquí la comunicación se establece en firme y comienza el intercambio de información entre los dispositivos.

Si el bit de lectura/escritura (R/W) fue puesto en esta comunicación a nivel lógico bajo (escritura), el dispositivo maestro envía datos al dispositivo esclavo. Esto se mantiene mientras continúe recibiendo señales de reconocimiento, y el contacto concluye cuando se hayan transmitido todos los datos.

En el caso contrario, cuando el bit de lectura/escritura estaba a nivel lógico alto (lectura), el dispositivo maestro genera pulsos de reloj para que el dispositivo esclavo pueda enviar los datos. Luego de cada byte recibido el dispositivo maestro (quien está recibiendo los datos) genera un pulso de reconocimiento.

El dispositivo maestro puede dejar libre el bus generando una condición de parada (o detención; stop en inglés).

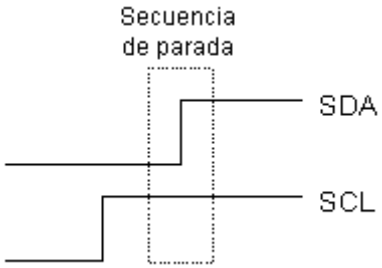


FIGURA 18. SECUENCIA DE PARADA

PROTOCOLO SPI

Serial Peripheral Interface (SPI) es un protocolo de datos en serie síncrono utilizado por los microcontroladores para comunicarse con uno o más dispositivos periféricos rápidamente en distancias cortas. También se puede utilizar para la comunicación entre dos microcontroladores.

Con una conexión SPI siempre hay un dispositivo de master (por lo general un microcontrolador) que controla los dispositivos periféricos. Normalmente hay tres líneas comunes a todos los dispositivos:

MISO (Master En Slave Out): La línea de esclavo para el envío de datos al maestro.

MOSI (Master Sale Slave In): La línea de Maestro para el envío de datos a los periféricos,

SCK (Reloj serie): Los impulsos de reloj que sincronizan la transmisión de datos generada por el maestro

Y una línea específica para cada dispositivo:

SS (Slave Select): el pasador en cada dispositivo que el maestro puede utilizar para activar y desactivar dispositivos específicos.

Cuando de un dispositivo Slave Select pin esta en bajo, se comunica con el maestro. Cuando es alta, no tiene en cuenta el maestro. Esto le permite tener múltiples dispositivos SPI que comparten la misma MISO, MOSI y líneas CLK.

Conexión.

TIPO ARDUINO	MOSI	MISO	SCK	SS (esclavo)	SS (maestro)
ARDUINO MEGA	51 o ICSP-4	50 o ICSP-1	52 o ICSP-3	53	

TABLA 6. CONEXIONADO

Tenga en cuenta que MISO, MOSI, SCK y están disponibles en una ubicación física consistente en la cabecera ICSP; Esto es útil, por ejemplo, en el diseño de una tarjeta que funciona en cada tabla.

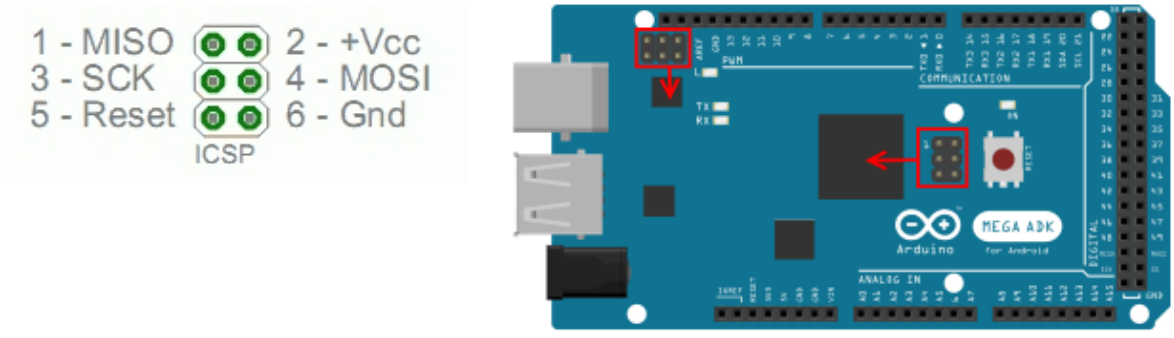


FIGURA 19. PINES SPI EN EL ARDUINO MEGA

COMUNICACIÓN SERIAL

UART, son las siglas en inglés de Universal Asynchronous Receiver-Transmitter, en español: Transmisor-Receptor Asíncrono Universal, es el dispositivo que controla los puertos y dispositivos serie.

El controlador del UART es el componente clave del subsistema de comunicaciones series de una computadora. El UART toma bytes de datos y transmite los bits individuales de forma secuencial. En el destino, un segundo UART re ensambla los bits en bytes completos. La transmisión serie de la información digital (bits) a través de un cable único u otros medios es mucho más efectiva en cuanto a costo que la transmisión en paralelo a través de múltiples cables. Se utiliza un UART para convertir la información transmitida entre su forma secuencial y paralela en cada terminal de enlace. Cada UART contiene un registro de desplazamiento que es el método fundamental de conversión entre las forma secuencial y paralela.

El UART normalmente no genera directamente o recibe las señales externas entre los diferentes módulos del equipo. Usualmente se usan dispositivos de interfaz separados para convertir las señales de nivel lógico del UART hacia y desde los niveles de señalización externos.

Las señales externas pueden ser de variada índole. Ejemplos de estándares para señalización por voltaje son RS-232, RS-422 y RS-485 de la EIA. Históricamente se usó la presencia o ausencia de corriente en circuitos telegráficos.

Algunos esquemas de señalización no usan cables eléctricos; ejemplo de esto son la fibra óptica, infrarrojo (inalámbrico) y Bluetooth (inalámbrico). Algunos esquemas de señalización emplean una modulación de señal portadora (con o sin cables); por ejemplo, la modulación de señales de audio con módems de línea telefónica, la modulación en radio frecuencia (RF) en radios de datos y la DC-LIN para la comunicación de línea eléctrica.

SERVIDOR WEB

Un servidor, como la misma palabra indica, es un ordenador o máquina informática que está al “servicio” de otras máquinas, ordenadores o personas llamadas clientes y que le suministran a estos, todo tipo de información.

El funcionamiento de una estructura cliente-servidor se presente a continuación en la imagen

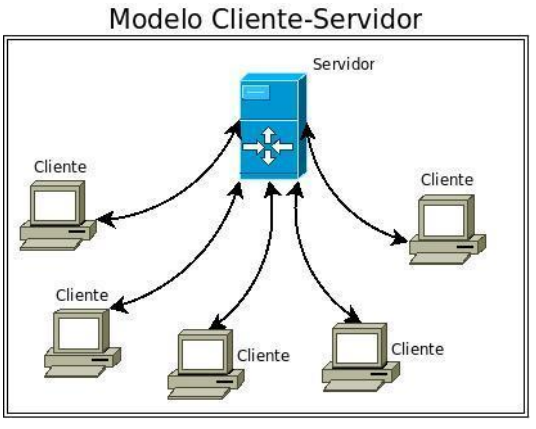


FIGURA 20. MODELO CLIENTE-SERVIDOR

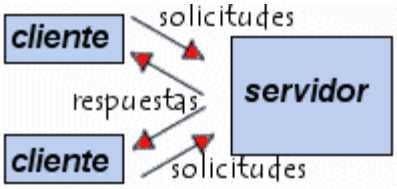


FIGURA 21. CLIENTE-SERVIDOR

El cliente envía una solicitud al servidor mediante su dirección IP y el puerto, que está reservado para un servicio en particular que se ejecuta en el servidor.

El servidor recibe la solicitud y responde con la dirección IP del equipo cliente y su puerto.

HTML

HTML es el lenguaje que se emplea para el desarrollo de páginas de internet. Está compuesto por una serie de etiquetas que el navegador interpreta y da forma en la pantalla. HTML dispone de etiquetas para imágenes, hipervínculos que nos permiten dirigirnos a otras páginas, saltos de línea, listas, tablas, etc.
Podríamos decir que HTML sirve para crear páginas web, darles estructura y contenido. Un ejemplo sencillo de código HTML podría ser:

```
<html>
<body>
<p>Esto es un párrafo. Bienvenidos a esta página web</p>
</body>
</html>
```

PROGRAMACIÓN CON CSS

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación es imprescindible para crear páginas web complejas.

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML muy bien definidos y con significado completo (también llamados "documentos semánticos"). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc.

Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc.

A continuación se presenta un ejemplo sin el uso de CSS

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de estilos sin CSS</title>
</head>
<body>
<h1><font color="red" face="Arial" size="5">Titular de la página</font></h1>
<p><font color="gray" face="Verdana" size="2">Un párrafo de texto
Introducción a CSS Capítulo 1. Introducciónwww.librosweb.es 7
largo.</font></p>
</body>
</html>
```

Acontinuación se presenta el mismo ejemplo con la utilización de CSS

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de estilos con CSS</title>
<style type="text/css">
h1 {color: red; font-family: Arial; font-size: large;}
 P {color: gray; font-family: Verdana; font-size: medium ;}
</style>
</head>
<body>
<h1>Titular de la página</h1>
<p>Un párrafo de texto no muy largo</p>
</body>
</html>
```

El ejemplo anterior utiliza la etiqueta <font> con sus atributos color, face y size para definir el color, la tipografía y el tamaño del texto de cada elemento del documento. El principal problema es definir el aspecto de los elementos, se puede ver claramente con el siguiente ejemplo: si la página tuviera 50 elementos diferentes, habría que insertar 50 etiquetas <font>. Si el sitio web entero se

compone de 10.000 páginas diferentes, habría que definir 500.000 etiquetas <font>. Como cada etiqueta <font> tiene 3 atributos, habría que definir 1.5 millones de atributos.

CSS permite separar los contenidos de la página y su aspecto o presentación. En el ejemplo anterior, dentro de la propia página HTML se reserva una zona en la que se incluye toda la información relacionada con los estilos de la página.

Utilizando CSS, en esa zona reservada se indica que todas las etiquetas <h1> de la página se deben ver de color rojo, con un tipo de letra Arial y con un tamaño de letra grande. Además, las etiquetas <p> de la página se deben ver de color gris, con un tipo de letra Verdana y con un tamaño de letra medio.

Definiendo los estilos de esta forma, no importa el número de elementos <p> que existan en la página, ya que todos tendrán el mismo aspecto establecido mediante CSS. Como se verá a continuación, esta forma de trabajar con CSS no es ideal, ya que si el sitio web dispone de 10.000 páginas, habría que definir 10.000 veces el mismo estilo CSS.

AJAX

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página, aunque existe la posibilidad de configurar las peticiones como síncronas de tal forma que la interactividad de la página se detiene hasta la espera de la respuesta por parte del servidor.

JAVASCRIPT

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java. Legalmente, JavaScript es una marca registrada de la empresa Sun Microsystems.

XML

XML (Extensible Markup Language) es un lenguaje de etiquetas, es decir, cada paquete de información está delimitado por dos etiquetas como se hace también en el lenguaje HTML, pero XML separa el contenido de la presentación.

HTTP

HTTP significa Protocolo de transferencia de hipertexto.
El propósito del protocolo HTTP es permitir la transferencia de archivos (principalmente, en formato HTML). Entre un navegador (el cliente) y un servidor web (denominado, entre otros, httpd en equipos UNIX) localizado mediante una cadena de caracteres denominada dirección URL. Un
navegador es un cliente HTTP, ya que envía las solicitudes a un servidor HTTP (servidor Web), que luego envía las respuestas de vuelta al cliente. La norma (y predeterminado) puerto para servidores HTTP para escuchar es el 80, aunque pueden utilizar cualquier puerto.

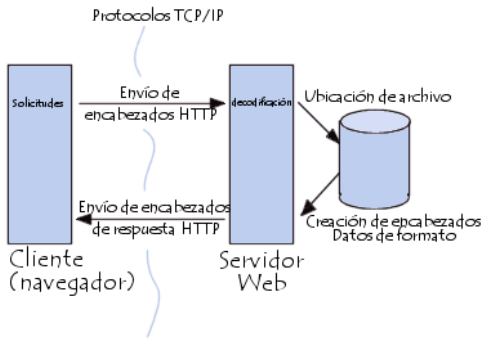


FIGURA 22. PROTOCOLOS TCP/IP

HTTP REQUEST

Una solicitud HTTP es un conjunto de líneas que el navegador envía al servidor. Incluye:

Una línea de solicitud: es una línea que especifica el tipo de documento solicitado, el método que se aplicará y la versión del protocolo utilizada. La línea está formada por tres elementos que deben estar separados por un espacio:

- Un método HTTP
- Un identificador universal de recursos (URI)
- Una versión del protocolo HTTP

El URI identifica el recurso que debe procesar la solicitud. Este URI se podrá especificar de forma absoluta o mediante una ruta relativa. Una solicitud con una URI no válida devolverá un código de error (normalmente el error 404).

- **Los campos del encabezado de solicitud**: es un conjunto de líneas opcionales que permiten aportar información adicional sobre la solicitud y/o el cliente (navegador, sistema operativo, etc.). Cada una de estas líneas está formada por un nombre que describe el tipo de encabezado, seguido de dos puntos (:) y el valor del encabezado.

- **El cuerpo de la solicitud**: es un conjunto de líneas opcionales que deben estar separadas de las líneas precedentes por una línea en blanco y, por ejemplo, permiten que se envíen datos por un comando POST durante la transmisión de datos al servidor utilizando un formulario.

Por lo tanto, una solicitud HTTP posee la siguiente sintaxis (<crlf> significa retorno de carro y avance de línea):

MÉTODO VERSIÓN URL<crlf>
ENCABEZADO: Valor<crlf>
. . . ENCABEZADO: Valor<crlf>
Línea en blanco <crlf>
CUERPO DE LA SOLICITUD

A continuación se presenta un ejemplo de una solicitud HTTP.

GET http://es.kioskea.net HTTP/1.0 Accept: Text/html If-Modified-Since: Saturday, 15-January-2000 14:37:11 GMT User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)

EL METODO

Comando	Descripción
GET	Solicita el recurso ubicado en la URL especificada
HEAD	Solicita el encabezado del recurso ubicado en la URL especificada
POST	Envía datos al programa ubicado en la URL especificada
PUT	Envía datos a la URL especificada
DELETE	Borra el recurso ubicado en la URL especificada

TABLA 7. METODOS DE HTTP REQUEST

METODO GET Y POST

El método GET solicita un recurso del servidor indicado en el campo URI. Si la URI apunta a una base de datos de producción de recursos como un server, los datos serán devueltos dentro del mensaje de respuesta.

El método POST se utiliza para pasar explícitamente datos al servidor en el propio mensaje de solicitud. Estos datos estarán a disposición del recurso URI especificado en la solicitud.

Las llamadas GET pueden ser cacheadas (historial del navegador), indexadas por buscadores, agregar los enlaces a nuestros favoritos o hasta pasar una URL completa a otra persona para que directamente ingrese a esa página. Con el método POST sin embargo no se puede hacer esto.

Encabezados

Nombre del encabezado	**Descripción**
Accept	Tipo de contenido aceptado por el navegador (por ejemplo, *texto/html*). Consulte Tipos de MIME
Accept-Charset	Juego de caracteres que el navegador espera
Accept-Encoding	Codificación de datos que el navegador acepta
Accept-Language	Idioma que el navegador espera (de forma predeterminada, inglés)
Authorization	Identificación del navegador en el servidor
Content-Encoding	Tipo de codificación para el cuerpo de la solicitud
Content-Language	Tipo de idioma en el cuerpo de la solicitud
Content-Length	Extensión del cuerpo de la solicitud
Content-Type	Tipo de contenido del cuerpo de la solicitud (por ejemplo, *texto/html*). Consulte Tipos de MIME
Date	Fecha en que comienza la transferencia de datos
Forwarded	Utilizado por equipos intermediarios entre el navegador y el servidor
From	Permite especificar la dirección de correo electrónico

	del cliente
From	Permite especificar que debe enviarse el documento si ha sido modificado desde una fecha en particular
Link	Vínculo entre dos direcciones URL
Orig-URL	Dirección URL donde se originó la solicitud
Referer	Dirección URL desde la cual se realizó la solicitud
User-Agent	Cadena con información sobre el cliente, por ejemplo, el nombre y la versión del navegador y el sistema operativo

TABLA 8. ENCABEZADOS

HTTP RESPONSE

Una vez que el servidor ha recibido y procesado la solicitud, éste debe devolver un mensaje de respuesta HTTP hacia el cliente. El mensaje de respuesta se compone de una línea de estado y un cero o más campos de cabecera, seguido por una línea vacía. También puede tener opcionalmente un cuerpo del mensaje.

La primera línea del mensaje de respuesta HTTP que se conoce como la línea de estado, se compone de HTTP versión del protocolo que se ajusta a la respuesta, seguida por un código numérico y su estatuto de texto explicación. Cada campo está separado del siguiente, por un espacio. Un ejemplo de línea de estado es la respuesta se muestra aquí: HTTP/1.1 200 OK. El código de estado es de tres dígitos de un valor numérico que se corresponde con el resultado del código del servidor intento de satisfacer la petición. El código de estado es para aplicaciones de programación, mientras que el texto que lo acompaña está destinado a los lectores humanos. El primer dígito del código de estado define la categoría del resultado código.

Una línea de estado: es una línea que especifica la versión del protocolo utilizada y el estado de la solicitud en proceso mediante un texto explicativo y un código. La línea está compuesta por tres elementos que deben estar separados por un espacio: La línea está formada por tres elementos que deben estar separados por un espacio:

- la versión del protocolo utilizada.
- el código de estado.
- el significado del código.

Los campos del encabezado de respuesta: es un conjunto de líneas opcionales que permiten aportar información adicional sobre la respuesta y/o el servidor.

Cada una de estas líneas está compuesta por un nombre que califica el tipo de encabezado, seguido por dos puntos (:) y por el valor del encabezado Cada una de estas líneas está formada por un nombre que describe el tipo de encabezado, seguido de dos puntos (:) y el valor del encabezado.

El cuerpo de la respuesta: contiene el documento solicitado.

A continuación se pone un ejemplo de una respuesta HTTP.

HTTP/1.0 200 OK Date: Sat, 15 Jan 2000 14:37:12 GMT Server : Microsoft-IIS/2.0 Content-Type : text/HTML Content-Length : 1245 Last-Modified : Fri, 14 Jan 2000 08:25:13 GMT

Código de estado	Significado del valor numérico
100-199	Informativo - La solicitud fue recibida y se está procesando.
200-299	Éxito - La acción fue recibida con éxito, entendido y aceptado.
300-399	Redirección - Otras peticiones hay que realizar para completar la petición.
400-499	Error en Cliente - La solicitud contiene mala sintaxis y no pueden llevarse a cabo.
500-599	Error de servidor - El servidor no pudo cumplir con una solicitud aparentemente válida.

Encabezados de respuesta

Nombre del encabezado	**Descripción**
Content-Encoding	Tipo de codificación para el cuerpo de la respuesta
Content-Language	Tipo de idioma en el cuerpo de la respuesta
Content-Length	Extensión del cuerpo de la respuesta
Content-Type	Tipo de contenido del cuerpo de la respuesta (por ejemplo, *texto/html*). Consulte Tipos de MIME
Date	Fecha en que comienza la transferencia de datos
Expires	Fecha límite de uso de los datos
Forwarded	Utilizado por equipos intermediarios entre el navegador y el servidor

Location	Re direccionamiento a una nueva dirección URL asociada con el documento
Server	Características del servidor que envió la respuesta

TABLA 9. ENCABEZADOS DE RESPUESTA

LED´S 5050 RGB

Los leds RGB no son más que tres leds en un mimo empaque, estos leds están compuestos de leds de colores primarios: rojo (Red), verde (Green), y azul (Blue), al variar la intensidad de corriente de cada led se producen diferentes colores. Una tira led RGB, cambia de colores. Se usa por excelencia para decoraciones, y para ofrecer en una misma instalación diversos tipos de ambientes. (Luz en movimiento). Es necesario un controlador (mando a distancia) o DMX para poder realizar los cambios de colores, programándolos o manualmente.

Características

Exterior Uso Rudo
300 LEDs 5050

FSL-5050RGB300-E/W ← Color de PCB

Color del LED

Especificaciones

Tipo de LED: Montaje superficial 5050

Cantidad de LEDs: 150 LEDs

Y= Colores disponibles

Cinta adhesiva 3M

Longitud total: 5 metros

Ángulo de apertura: 120 grados

	R	G	B
Flujo Luminoso (lm)	330-380	800-850	110-130
Longitud de onda (nm)	600-630	490-520	420-470
Consumo Watts/Amp.	18/1.5	15/1.3	14/1.2
Temperatura de color (K)			

Voltaje de operación: 12 Volts

Tipo de protección: Para uso en exteriores

Linea de corte: cada 3 LEDs, 5 cm de longitud

Dimensiones: 1.0x0.2x500 cm

Nota: No conectar tiras en serie.

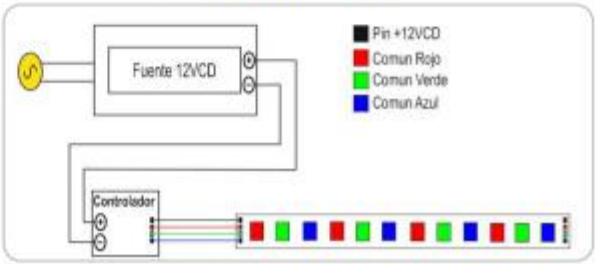


FIGURA 23. CARACTERISTICAS DE LOS LED´S RGB

CAPITULO III
PROCEDIMIENTO Y DESCRIPCIÓN DE LAS ACTIVIDADES REALIZADAS

A continuación se presenta un diagrama a pasos de las etapas realizadas en el proyecto. Cada etapa realizada se describe a detalle en todo el capítulo.

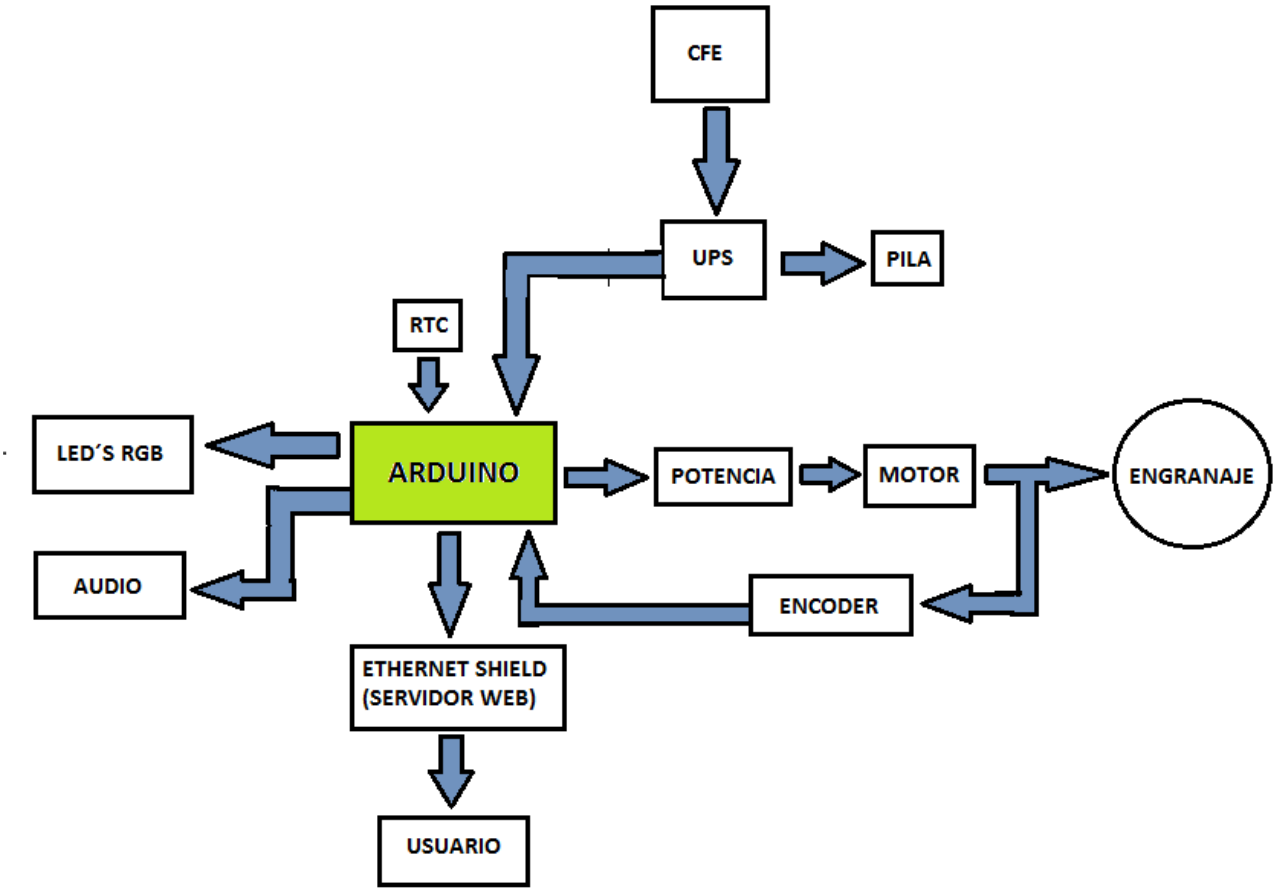


Figura 24. DIAGRAMA A BLOQUES DEL SISTEMA COMPLETO

Nota: cabe señalar que los últimos dos bloques son a nivel propuesta

El UPS (sistema ininterrumpido de potencia) alterna la alimentación de la línea de CFE a una de respaldo (pila), en caso de que la corriente eléctrica dejase de existir por un periodo de tiempo.
Mediante el uso del RTC se tendrá control de tiempo (horas y minutos), para con esto activar el motor de corriente directa cada minuto y con ello mover las manecillas del reloj hasta la posición requerida. Se necesita tener control sobre la vueltas que dará el motor para poder mover un minuto las manecillas, se utilizara un encoder, con el se sabrá la cantidad de pulsos requeridos para mover las manecillas hasta la posición deseada.
Mediante el uso de una página web se podrá observar que el tiempo que lleva el RTC no se atrase o adelante usando como referencia el tiempo de una computadora o dispositivo conectado al servidor en ese instante, si se llagase a dar el caso, existe un botón que es capaz de extraer el tiempo del dispositivo y cargárselo directamente al RTC. Una vez ajustado el tiempo el motor tendrá que girar las vueltas necesarias para poder llevar las manecillas hasta la hora correcta.
La página web estará almacenada en una micro SD la cual estará en la Ethernet Shield, la tarjeta tendrá el papel de servidor Web. El servidor estará en una red de área local (LAN).
Si hubiera ausencia de corriente eléctrica, el RTC es capaz de seguir llevando conteo del tiempo, ya que cuenta con una pila. Cuando regrese la luz eléctrica se comparara el tiempo del RTC contra el tiempo almacenado en la memoria EEPROM del arduino, se hará la diferencia y posteriormente se encenderá el motor hasta llevar a las manecillas a la posición correcta.
Con el arduino también se tendrá control sobre el encendido de la luminaria RGB.
Y mediante el uso de una Shield Mp3 se reproducirán tonos de campana en función de las horas y posteriormente una fracción de canción representativa del estado de Chiapas.

DIAGRAMA DE BLOQUES A NIVEL SOFTWARE

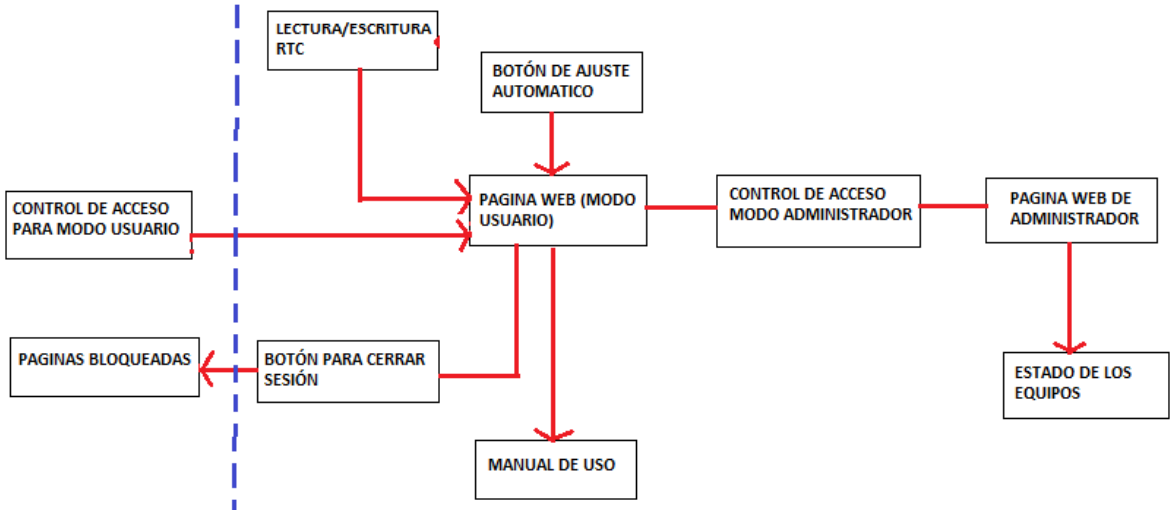


FIGURA 25. DIAGRAMA DE BLOQUES A NIVEL SOFTWARE

PROCESO SET TIME (AJUSTE DE TIEMPO EN EL RTC)

DIAGRAMA DE FLUJO

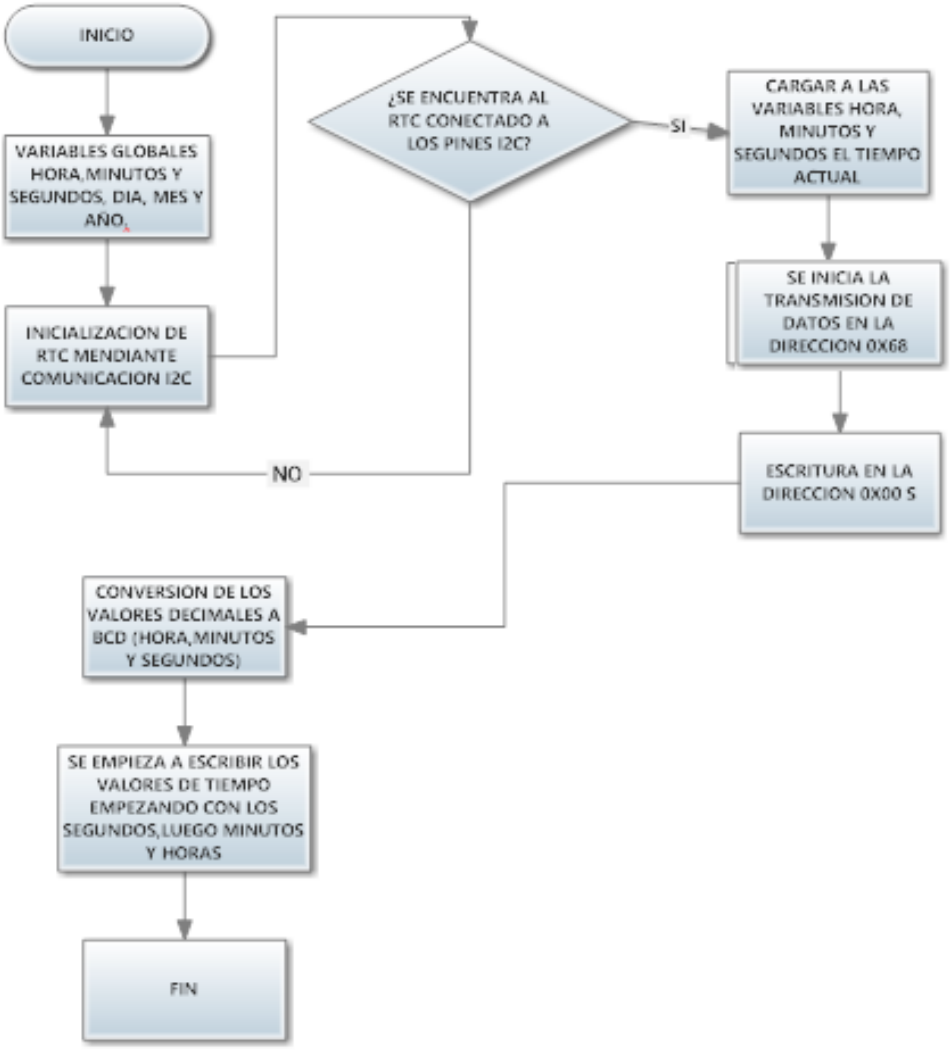


FIGURA 26. DIAGRAMA DE FLUJO (RTC)

CÓDIGO ARDUINO

```
#include<Wire.h>
#define rtc_add 0x68

int seconds=1;
int minutes=47;
int hours=17;
int dayweek=4;
int daymonth=14;
int month_rtc=1;
int year_rtc=16;

void setup()
{
  Wire.begin();
}

void loop()
{
  set_time();
  delay(1000);
}

byte decAbcd(byte conversion)
{
  return ((conversion / 10 * 16) + (conversion % 10));
}

  void set_time()
{
  Wire.beginTransmission(rtc_add);
  Wire.write(0x00); //(0)
  Wire.write(decAbcd(seconds));
  Wire.write(decAbcd(minutes));
  Wire.write(decAbcd(hours));
  Wire.write(decAbcd(dayweek));
  Wire.write(decAbcd(daymonth));
  Wire.write(decAbcd(month_rtc));
  Wire.write(decAbcd(year_rtc));
  Wire.endTransmission();
}
```

Para poder realizar la comunicación I2C hay que añadir a nuestro sketch de arduino la librería Wire.h.
Una vez añadida la librería, se definió una constante que almacena la **dirección 0x68,** dicha dirección le corresponde al DS1307.
El valor **0x00**, es la primera dirección en el mapa de memoria del RTC y le corresponde a los segundos. Dentro de esta dirección se pone a 0 el bit 7, se procede de esa manera debido a que este bit tiene la labor de indicar al DS1307 que debe empezar a realizar el conteo del tiempo. De manera genérica el RTC tiene ese bit puesto a uno, esto con el fin de que este en modo de reposo (no contabiliza tiempo).
Para inicializar la comunicación entre el Arduino Mega y el DS1307, se usó el código **Wire.begin()**, con ello exhortamos al Arduino que los pines 20 y 21 serán empleados para el uso de comunicación I2C, sobre estos pines se conectaron el SDA y SCL del DS1307.
Una vez escrito los datos sobre los registros de segundos, minutos y hora se cierra la transmisión de datos entre el arduino y el RTC con el fin de realizar la escritura de tiempo en el RTC y así no rescribir siempre los mismos datos.
Este código solo ajusta el tiempo del RTC por lo tanto no se podrá visualizar como el RTC empieza a hacer el conteo de los segundos y así sucesivamente hasta las horas.

Se tendrá el formato del tiempo de la siguiente manera:

15:47:1

LECTURA DE TIEMPO RTC

El siguiente diagrama de flujo sirve para leer los registros en el RTC

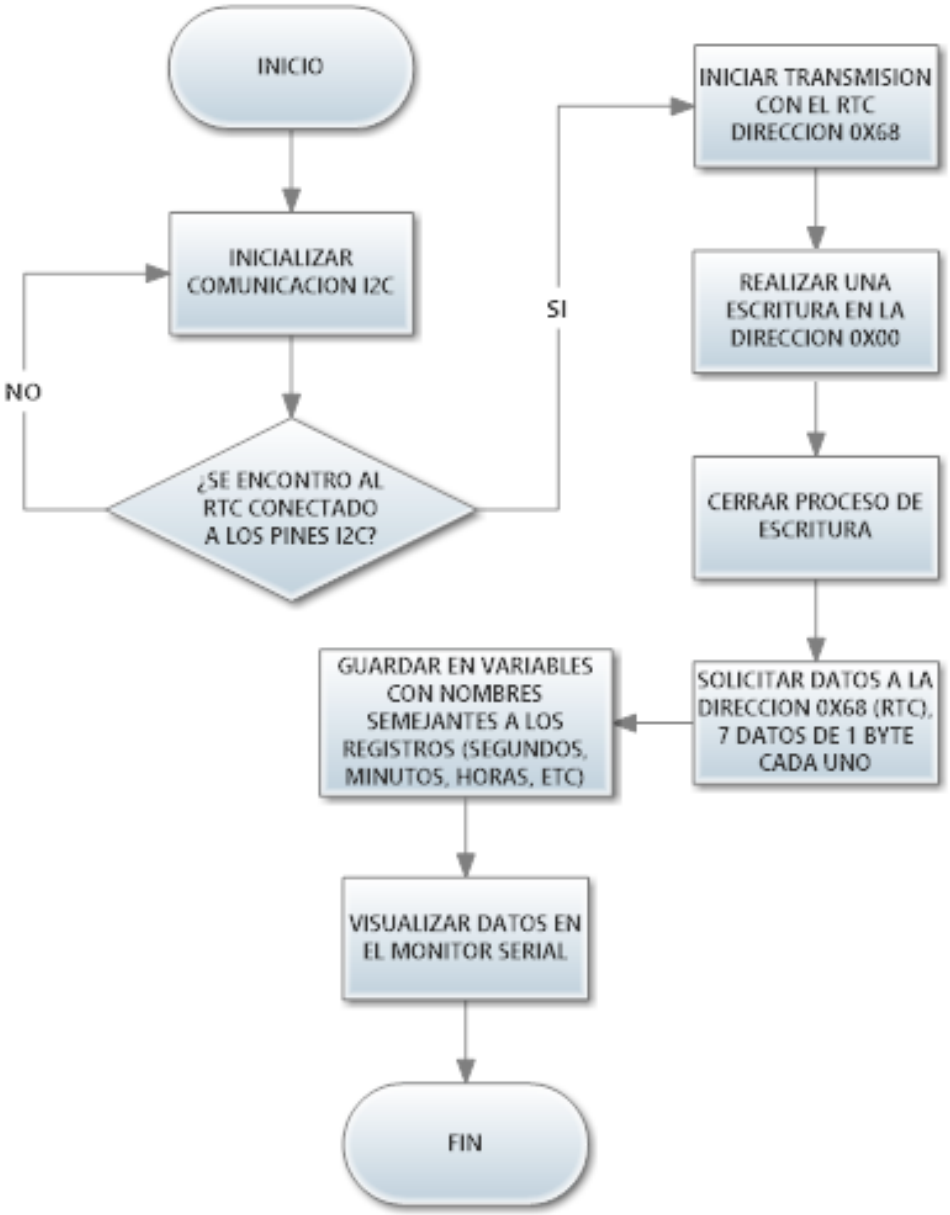


FIGURA 27. DIAGRAMA DE FLUJO DE LECTURA DEL RTC

- **CODIGO ARDUINO**

```
#include<Wire.h>
#define rtc_add 0x68

int seconds;
int minutes;
int hours;
int dayweek;
int daymonth;
int month_rtc;
int year_rtc;

void setup()
{
  Wire.begin();
  Serial.begin(9600);
}

void loop()
{
  get_time;
  delay(1000);
}

void get_time()
{
  Wire.beginTransmission(rtc_add);
  Wire.write(0);
  Wire.endTransmission();
  Wire.requestFrom(rtc_add, 3);
  seconds = bcdAdec(Wire.read());
  minutes = bcdAdec(Wire.read());
  hours = bcdAdec(Wire.read() & 0x3f);
  Serial.print(hours, DEC);
  Serial.print(":");
  Serial.print(minutes, DEC);
  Serial.print(":");
  Serial.print(seconds, DEC);
  Serial.print("  ");
}
```

```
byte bcdAdec(byte val)
{
  return ( (val/16*10) + (val%16) );
}
```

Al igual que en el set_time se hace uso de la librería Wire.h.
La notable diferencia de este código al anterior es que aquí se pueden empezar a leer los datos (tiempo) almacenado en el RTC y al igual que en el código pasado se inicia la comunicación al RTC con **Wire.beginTransmissio()**.
Para poder leer el registro 0x00 de los segundos primero hay que realizar una escritura y cerrar la transmisión de datos, después se realiza una solicitud de datos del RTC con **Wire.requestFrom(0x68,7)**, para recibir 7 datos, cada dato tiene el tamaño de una byte y son almacenados en las variables correspondientes al byte recibido, al igual manera que en la escritura el primer byte corresponde a los segundos y el segundo a los minutos y así sucesivamente.
Para visualizar que el RTC en realidad está llevando un conteo del tiempo, se usa el Serial.print que imprime en el monitor serial los datos leídos por el Arduino del RTC.

SERVIDOR WEB

ETHERNET SHIELD

La Ethernet Shield 2 se configuró para ser el servidor, para esto se usó el **CMD** del Sistema operativo Windows, con el fin de poder ver las direcciones que se le asignan a la Ethernet shield. Al trabajar en un esquema de Red de datos, se asignó una dirección IP estática, una MAC y se habilitó el puerto 80 del servidor con el fin de realizar la comunicación con los o el cliente que se conecten a nuestro servidor.

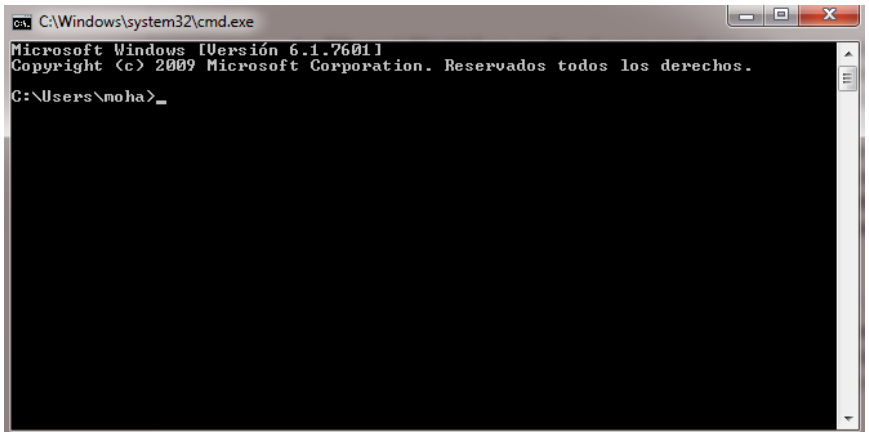


FIGURA 28. SIMBOLO DE SISTEMA (CMD)

La Ethernet shield se encuentra trabajando en una red LAN (loca área network o red de área local) por lo tanto no hay salida a la internet, solamente da servicio a la red de un área específica, en este caso la red perteneciente a la secundaria del estado turno vespertino.

En la ventana del CMD se introduce el comando “ipconfig” una vez tecleado el comando se despliega la información sobre la red a la que estamos conectados, desde la IP, la máscara de subred y la puerta de enlace, esta última le corresponde a la dirección del equipo que nos prevea el servicio de red a nivel LAN con salida hacia una red exterior.

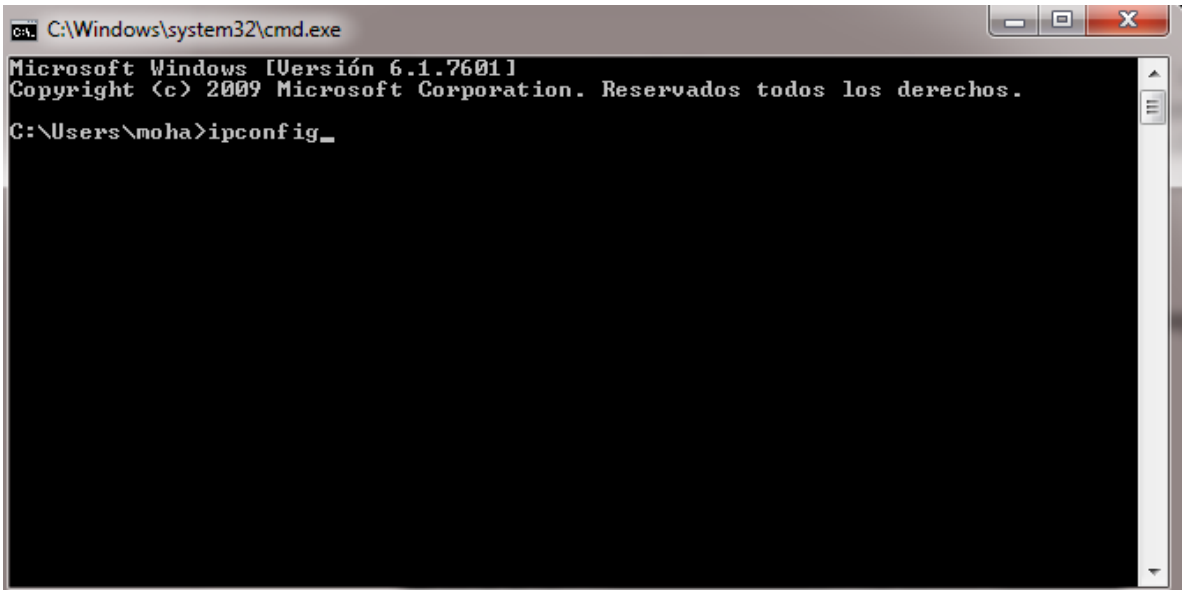


FIGURA 29. COMANDO IPCONFIG

Una vez visualizados los datos como se muestran en la imagen, proseguimos a configurar nuestra Ethernet shield.

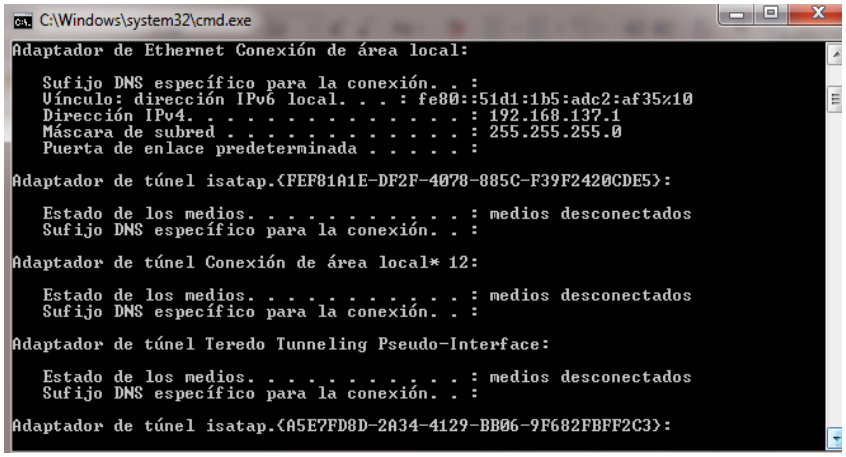


FIGURA 30. CONFIGURACION DE LA ETHERNET SHIELD

Debido a que nuestra Ethernet está conectada directamente a nuestra computadora mediante cable de red, aparecerá que la tarjeta de red tiene la dirección 192.168.137.1 y una máscara de subred 255.255.255.0. En función de estos datos podemos asignar una dirección ip a nuestra tarjeta Ethernet Shield que podrá ir desde la dirección 192.168.137.2 hasta 192.168.137.255
La dirección IP seleccionada para la Ethernet Shield es la 192.168.137.1.100 puede darse el caso de que esta dirección IP este ocupada por algún otro dispositivo. Para saber si esta o no ocupada esa dirección en el Símbolo de sistemas se teclea el comando “ping” acompañado de la dirección deseada. Ejemplo “ping 192.168.137.100”.

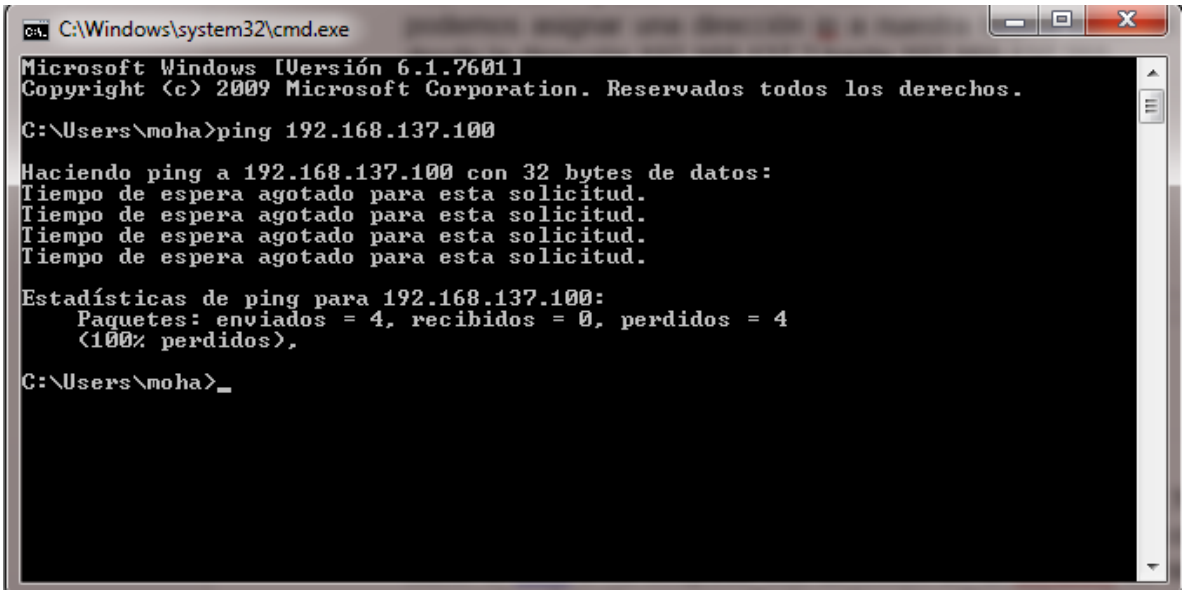


FIGURA 31. VERIFICACION DE IP DISPONIBLE

Se aprecia en la imagen que del ping realizado a la dirección IP se perdió el paquete enviado de datos, lo que da a entender que la dirección IP está disponible para su uso.

```
byte mac[] = {
0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
IPAddress ip(192, 168, 137, 100);
```

Para poder hacer uso de la Shield, en el Arduino se añadieron dos librerías:

- SPI.H
- ETHERNET2.h.

DIAGRAMA DE FLUJO

El siguiente diagrama de flujo muestra la shield Ethernet como servidor.

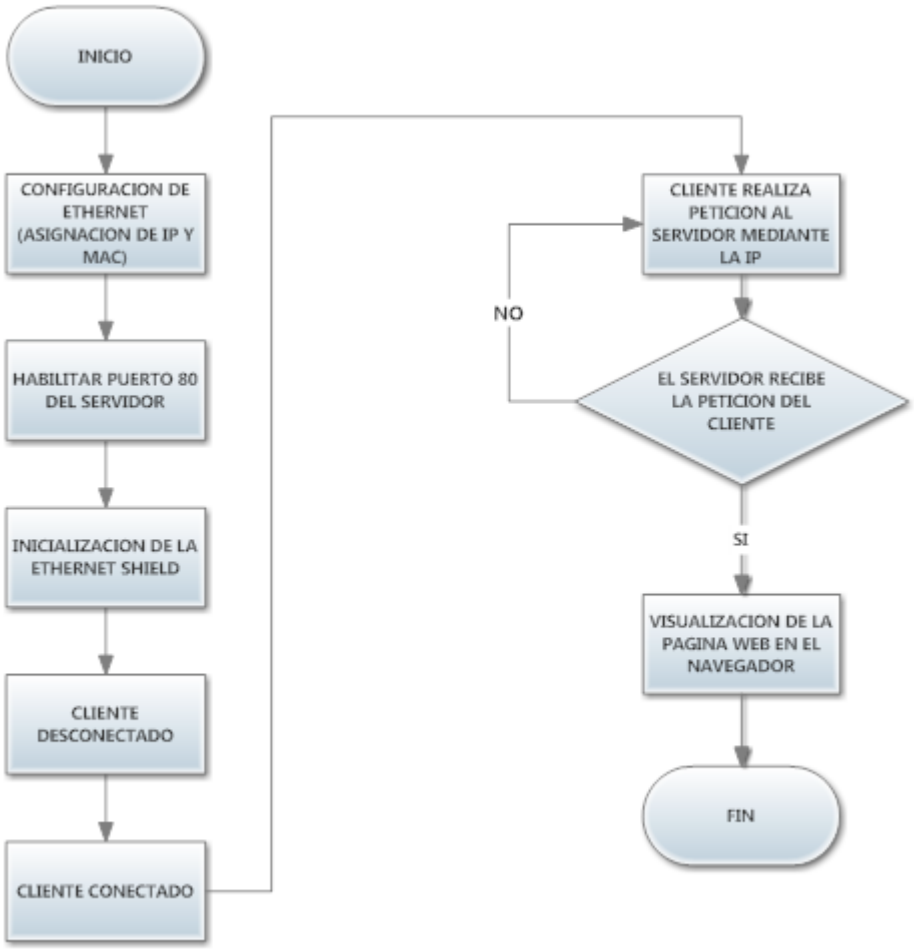


FIGURA 32. DIAGRAMA DE LA SHIELD ETHERNET

A continuación se muestra un ejemplo muy sencillo de un servidor con arduino y la Ethernet shield, este ejemplo puede ser encontrado en el compilador arduino.

CODIGO ARDUINO

```
#include <SPI.h>
#include <Ethernet2.h>

// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
IPAddress ip(192, 168, 1, 177);

// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }


  // start the Ethernet connection and the server:
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}


void loop() {
  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    Serial.println("new client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
```

```
        Serial.write(c);
        // if you've gotten to the end of the line (received a newline
        // character) and the line is blank, the http request has ended,
        // so you can send a reply
        if (c == '\n' && currentLineIsBlank) {
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");  // the connection will be closed after completion of the response
          client.println("Refresh: 5");  // refresh the page automatically every 5 sec
          client.println();
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");
          // output the value of each analog input pin
          for (int analogChannel = 0; analogChannel < 6; analogChannel++) {
            int sensorReading = analogRead(analogChannel);
            client.print("analog input ");
            client.print(analogChannel);
            client.print(" is ");
            client.print(sensorReading);
            client.println("<br />");
          }
          client.println("</html>");
          break;
        }
        if (c == '\n') {
          // you're starting a new line
          currentLineIsBlank = true;
        }
        else if (c != '\r') {
          // you've gotten a character on the current line
          currentLineIsBlank = false;
        }
      }
    }
    // give the web browser time to receive the data
    delay(1);
    // close the connection:
    client.stop();
    Serial.println("client disconnected");
  }}
```

Una vez asignada la IP a la Ethernet shield se creó un objeto llamado server (80), esto habilitó el puerto 80 que tiene como fin permitir que el servidor pueda escuchar a los clientes.
Dentro del void setup se inicializó el servidor con el comando **server.begin(),** y también se inicializó una comunicación serial con el fin de poder visualizar las peticiones realizadas por el cliente.
Sobre la URL hay información que interpreta el servidor para poder ejecutar la acción programada a la llegada de la información.
La condición para que el código HTML se reprodujera e interpretara por el navegador fue hacer un salto de línea y una línea en blanco, esto debido a que a la hora de acceder al contenido desde el navegador realiza un petición al servidor para que no pueda dar acceso a ese código, cuando la petición esta completa esta terminar con un salto de línea lo que en ASCII es (‘\n’) y una línea en blanco, esto da a entender que ya existe acceso a la página de internet alojada en el servidor, el navegador interpreta las líneas en HTML y es capaz de reproducir todo en una página de internet.

DISEÑO DE PÁGINA

Para la creación de la página, se usó el software NotePad++, a continuación se anexa el código CSS y HTML

CODIGO EN NOTEPAD++

```
<style type="text/css">
  body
   {
     padding-left: 11em;
     font-family: Georgia, "Times New Roman",
     Times, serif;
     color: black;
     background-color:white;
   }
  ul.navbar
   {
    list-style-type: none;
    padding: 0;
    margin: 0;
    position: absolute;
    top: 10em;
    left: 1em;
    width: 9em;
```

```
 }
h1
 {
  font-family: Helvetica, Geneva, Arial, SunSans-Regular,sans-serif
 }
ul.navbar li
 {
  background: #A9F5BC;
  margin: 0.5em 0;
  padding: 0.3em;
  border-right: 1em solid #088A08;
 }
ul.navbar a
{
 text-decoration: none
 background: green;
}
a:link
{
 color: black
}
a:visited
{
 color: black
}
address
{
margin-top: 1em;
padding-top: 1em;
border-top: thin dotted
}

h2 {
 animation-duration: 3s;
 animation-name: slidein
}

@keyframes slidein {
 from {
   margin-left: 100%;
   width: 300%
 }
```

```
  to {
    margin-left: 0%;
    width: 100%;
  }
}

.txt_block {
        border: 2px solid green;
        width: 70px;
        height: 130px;
        padding-left: 10px;
        float: center;
        margin-right: 16px;
      }

.txt_block1 {
        border: 2px solid green;
        width: 70px;
        height: 130px;
        padding-left: 10px;
        float: center;
        margin-right: 5px;
      }


</style>
</head>
<body onload="arduino();tiempo_maquina();" background-color="#D8F6CE">

<div style=" border-right : 50px  groove #01DF01;">
<div style=" border-left : 50px  groove #01DF01;">

<ul class="navbar">
  <li><a href="ese1.htm">Administrador</a>
  <li><a href="manual.htm">manual uso</a>
</ul>

<center>
<h1><marquee  align="center"  scrolldelay="5"> RELOJ  DE  LA  ESCUELA SECUNDARIA DEL ESTADO VESPERTINA</marquee></h1>
<br/>

<form name="check_hora">
```

```
<p><h2><font size=6>Hora Actual de Mexico</h2></font><br/>
<font size=8><span id="tiempo" class="txt_block">.....</span></font></p>
</form>
<br/>

<div class="IO_box">
<h2><font size=6>Tiempo del Reloj ESE</font></h2>
<p/><font size=8><span class="analog2 txt_block">???</span>:<span
class="analog1 txt_block">???</span>:<span class="analog3
txt_block">???</span></font><br/><br/>
<button type="button" onclick="agrega_hora()">hora(+)</button><button
type="button" onclick="agrega_minuto()">minuto(+)</button>
</div>
</center>
</br>
<center>
<button type="button" onclick="ajuste_automatico()">AJUSTE
AUTOMATICO</button>
</center>

<button type="button" onclick="cerrar_sesion()">Cierre su sesion </button>

<p>envio:<span id="envio">????</span></p>
<p>envio1:<span id="envio1">????</span></p>
<p>ajusto automatico hora:<span id="ajuste_hora">.....</span></p>
<p>ajusto automatico minuto:<span id="ajuste_minuto">.....</span></p>
<p>bandera:<span id="f_1">?????</span></p>
<p>conteo:<span id="dat">....</span></p>

</body>
</html>
```

Cuando se reproduce en un navegador como el Chrome la página tendrá el siguiente aspecto.

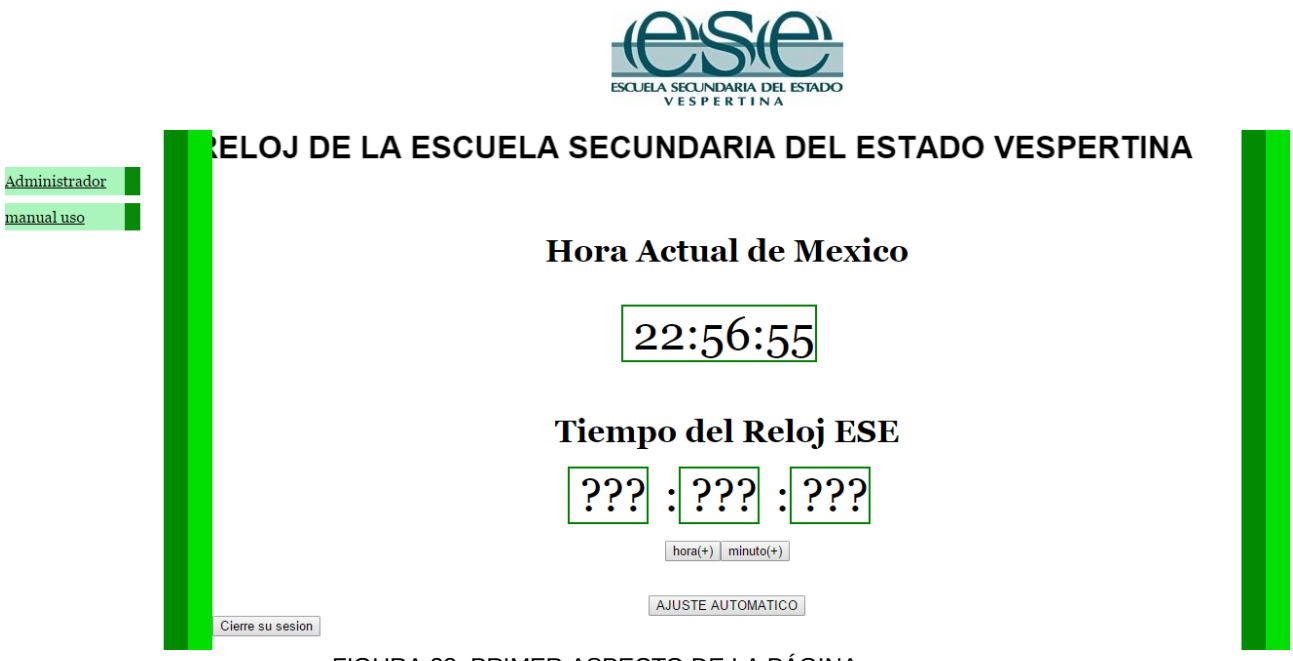


FIGURA 33. PRIMER ASPECTO DE LA PÁGINA

Una de las ventajas del uso de CSS en el diseño de páginas web, es que podemos hacer más general el contenido. Podemos modificar propiedades de las etiquetas del código, esto nos ayudó a que toda etiqueta que aparece en la página contendrá las características establecidas de CSS, por ejemplo:

```
body
  {
    padding-left: 11em;
    font-family: Georgia, "Times New Roman",
    Times, serif;
    color: black;
    background-color:white;
  }
```

Para hacer las modificaciones a las etiquetas en el lenguaje CSS es necesario poner entre las etiquetas <head> y </head> la etiqueta <style type="text/css"> y cerrar con </style> para indicar que se había terminado de hacer uso del lenguaje.

ENVIO DE DATOS

Los datos del tiempo del RTC, Es decirlas horas y minutos se envían hacia la página por medio de XML en etiquetas, las etiquetas se llaman:

- Analog1
- Analog2
- Analog3

Donde cada uno corresponde a las horas, minutos y segundos.
En la página web existe un botón con el nombre “ajuste automático”, el cual extrae el tiempo de la máquina que esté conectado al servidor en ese momento y se lo asigna al RTC.
A continuación se presente el código necesario para poder cargar al RTC el tiempo de un ordenador.

```
<script>
str="&min=0";
str1="&hor=0";
prub="0";
str2="";
str3="";
str4="";
var hora;
var fecha;
var minuto;
var texto="";
var pass="";

window.onbeforeunload = function exitAlert()
{
 var textillo = " ";
 return textillo;
}

function arduino()
{
 nocache="&nocache="+ Math.random()*1000000;
 var peticion= new XMLHttpRequest();
 peticion.onreadystatechange = function()
 {
        if(this.readyState==4&&this.status==200)
        {
                if(this.responseXML != null)
                {
```

```
                document.getElementsByClassName("analog1")[0].innerHTML
=this.responseXML.getElementsByTagName('analog1')[0].childNodes[0].nodeValu
e;a
                document.getElementsByClassName("analog2")[0].innerHTML
=this.responseXML.getElementsByTagName('analog2')[0].childNodes[0].nodeValu
e;
                document.getElementsByClassName("analog3")[0].innerHTML
=this.responseXML.getElementsByTagName('analog3')[0].childNodes[0].nodeValu
e;
                }
        }
 }
 peticion.open("GET","ajax_peticion" + str3 + str4 + str2 + pass + nocache, true);
 peticion.send(null);
 setTimeout('arduino()', 1000);
 str="min=0";
 str1="hor=0";
 str3="";
 str4="";
 pass="";
 }

 function tiempo_maquina()
{
fecha= new Date()
minuto=fecha.getMinutes()
hora=fecha.getHours()
segundo=fecha.getSeconds()
t_actual=hora+":"+minuto+":"+segundo
h_str="&hora"+hora
document.getElementById("tiempo").innerHTML=t_actual
setTimeout("tiempo_maquina()",1000)
}

function agrega_hora()
{
 str1="&hor=1";
 document.gtElementById("envio").innerHTML=str1;
}

function agrega_minuto()
{
str="&min=1";
```

```
document.getElementById("envio1").innerHTML=str;
}

function ajuste_automatico()
{
str3="&h="+hora;
str4="&m="+minuto;
document.getElementById("ajuste_hora").innerHTML=str3;
document.getElementById("ajuste_minuto").innerHTML=str4;
}

function cerrar_sesion()
{
 pass="&p="+"cerrar";
 document.getElementById("f_1").innerHTML=pass;
 setTimeout(function(){abrir()},5000)
}

function abrir()
{
 window.location.reload();
}
</script>
```

El código presentado hace que la página se convierta en una con contenido dinámico, esto quiere dar a entender que la página actualiza únicamente una sección de su contenido sin tener que actualizar toda la página.

CLAVE DE ACCESO

Se hizo una contraseña para el ingreso a la página, esto con el fin de que solamente una persona pueda hacer modificaciones, para esto se manipuló la memoria del arduino.

MAPA DE MEMORIA DE LA EEPROM

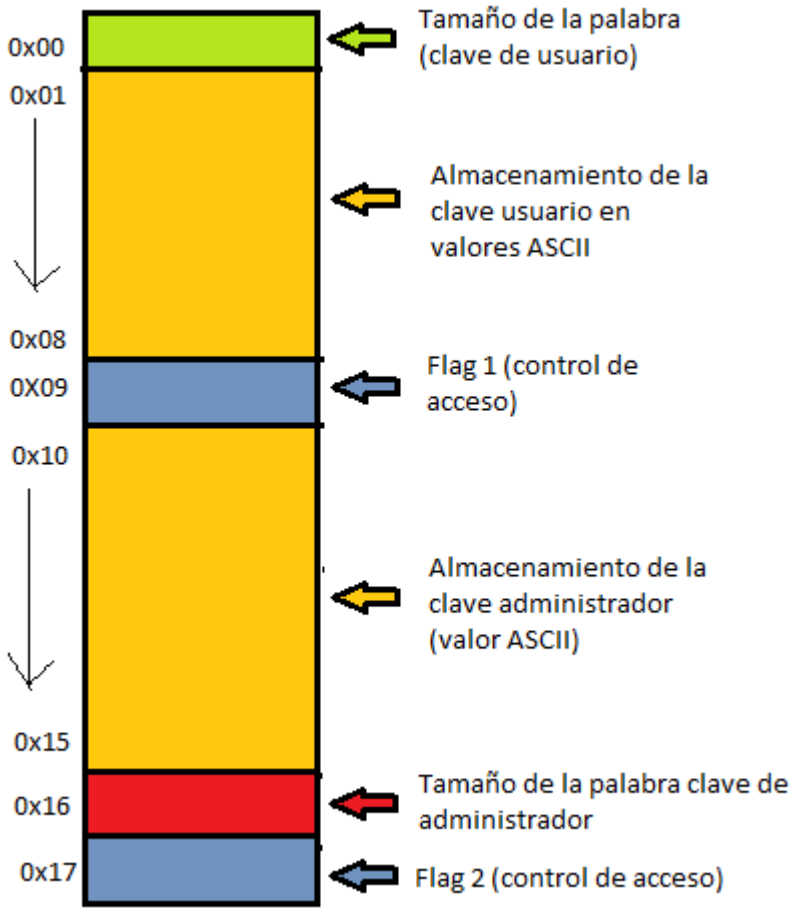


FIGURA 34. MAPA DE MEMORIA EEPROM

ACCESO CLAVE USUARIO

CÓDIGO DE PÁGINA WEB

```
<!DOCTYPE html>
<html>
<head>
<title>ESE ACCESO</title>
<img id="estirada" style="position: absolute; top: 0; left: 0; width: 100%; height:
100%" src="esc.jpg" />
<script>
<!--
var pass="";
var secret_word="";
function arduino_pass()
{
 nocache="&nocache="+ Math.random()*1000000;
 var peticion= new XMLHttpRequest();
 peticion.onreadystatechange = function()
 {
        if(this.readyState==4&&this.status==200)
        {
                if(this.responseXML != null)
                {
                  document.getElementsByClassName("pass1")[0].innerHTML =
this.responseXML.getElementsByTagName('pass1')[0].childNodes[0].nodeValue;

secret_word=this.responseXML.getElementsByTagName('pass1')[0].childNodes[0]
.nodeValue;
                  if(secret_word=="sesamo")
                  {
                   window.open("ese3.htm");
                        setTimeout(function(){cerrar()},10)

                  }
                }
        }
 }
 peticion.open("GET","acceso_reloj"+pass+nocache, true);
 peticion.send(null);
 setTimeout('arduino_pass()', 1000);
 pass="";
}
```

```
function clave()
{
pal=prompt('Control de acceso:','Por favor introduzca la clave de acceso');
pass="&x="+pal;
document.getElementById("text1").innerHTML=pass;
}

function cerrar()
{
 window.close();
}

//-->
</script>
</head>
<body onload="arduino_pass();clave();">
<p>texto:<span id="text1">....</span> </p>
<p>texto recivido:<span class="pass1">.....</span></p>
</body>
</html>
```

CODIGO DE ARDUINO

```
else if (contenedor(HTTP_req, "acceso_reloj"))
        {
          client.println("HTTP/1.1 200 OK");
          client.println("Connection: keep-alive");
          client.println();
          if (contenedor(HTTP_req, "&x="))
          {
            String pass_label = "x";
            char *pass = strstr(HTTP_req, "&x=");
            Serial.println("acceso:" + getPas(pas_label, pass));
            text_pass = getPass(pass_label, pass);
            long_pass = text_pass.length();
            Serial.println("*****");
            Serial.println(long_pass);
            Serial.println("*****");
            if (acceso_user(text_pas, long_pass))
            {
              xml_access(client);
            }
          }
```

```
String getPass(String index2, String web2)
{
  int lentext2 = web2.length();
  int lenindex2 = index2.length();
  int start2 = web2.indexOf(index2) + lenindex2 + 1;
  String valor2 = "";

  for (int i = start2; i < lentext2; i++)
  {
    String caracter2 = web2.substring(i, i + 1);
    if (caracter2 != "&")
    {
      valor2 = valor2 + caracter2;
    }
    else
    {
      i = lentext2;
    }
  }
  return valor2;
}

boolean acceso_user(String t, int long_t)
{
  bool flag = false;
  for (int x = 0; x <= len_eeprom; x++)
  {
    array_eeprom[x] = EEPROM.read(x + 1);
  }
  toString_eeprom = (String)array_eeprom;
  //--------------------------------------
  Serial.println("---------------");
  Serial.println(toString_eeprom);
  Serial.println("---------------");
  //--------------------------------------
  if (long_t == len_eeprom)
  {
    if (toString_eeprom.equals(t))
    {
      flag = true;
      EEPROM.write(9, 1);
    }
    else
```

```
    {
      flag = false;
      EEPROM.write(9, 0);
    }
  }
  return flag;
}

void xml_access(EthernetClient cl)
{
  cl.print("<?xml version = \"1.0\" ?>");
  cl.print("<inputs>");
  cl.print("<pass1>");
  cl.print("sesamo");
  cl.print("</pass1>");
  cl.println("</inputs>");
}
```

Cuando se teclea una contraseña en la página de internet ésta es enviada al servidor mediante la URL, bajo el método GET. La URL lleva un nombre de petición en este caso el nombre de la petición es “acceso_reloj”, si el servidor detecta este nombre, busca la etiqueta “&x=” dentro de la URL para posteriormente extraer el texto que la acompaña y se compara con el texto almacenado en la memoria EEPROM, si el texto coincide el sistema responde a la página con la palabra sésamo, esto con el fin de darle a entender que la palabra tecleada y la almacenada en la EEPROM son iguales, con lo que se accede la página de modo usuario.

NOTA.
El control de acceso al modo administrador funciona de la misma manera que el control de acceso a modo usuario. A continuación se presenta el código.

ACCESO CLAVE ADMINISTRADOR

CODIGO PÁGINA WEB

```
<script>
<!--
var pass="";
var secret_word="";
function arduino_pass()

{
 nocache="&nocache="+ Math.random()*1000000;
 var peticion= new XMLHttpRequest();
 peticion.onreadystatechange = function()
 {
        if(this.readyState==4&&this.status==200)
        {
                if(this.responseXML != null)
                {
                  document.getElementsByClassName("pass1")[0].innerHTML =
this.responseXML.getElementsByTagName('pass1')[0].childNodes[0].nodeValue;

secret_word=this.responseXML.getElementsByTagName('pass1')[0].childNodes[0]
.nodeValue;
                  if(secret_word=="ittg")
                  {
                   window.open("ese1.htm");
                  }
                }
        }
 }
 peticion.open("GET","admin_reloj"+pass+nocache, true);
 peticion.send(null);
 setTimeout('arduino_pass()', 100);
 pass="";
}

function clave()
{
pal=prompt('Control de acceso:','Por favor introduzca la clave de acceso');
pass="&k="+pal;
document.getElementById("text1").innerHTML=pass;
}
```

```
//-->
</script>

<style type="text/css">
```

CODIGO ARDUINO

```
else if (contenedor(HTTP_req, "admin_reloj"))
        {
          client.println("HTTP/1.1 200 OK");
          client.println("Connection: keep-alive");
          client.println();
          if (contenedor(HTTP_req, "&k="))
          {
            String pass_label2 = "k";
            char *pass2 = strstr(HTTP_req, "&k=");
            Serial.println("acceso:" + getPass(pass_label2, pass2));
            text_pass2 = getPass(pass_label2, pass2);
            long_pass2 = text_pass2.length();
            Serial.println("*****");
            Serial.println(long_pass2);
            Serial.println("*****");
            if (acceso_admin(text_pass2, long_pass2))
            {
              xml_admin(client);
            }
          }
        }

boolean acceso_admin(String t2, int long_t2)
{
  bool flag2 = false;
  for (int x = 0; x <= len_eeprom2; x++)
  {
    array_eeprom2[x] = EEPROM.read(x + 10);
  }
  toString_eeprom2 = (String)array_eeprom2;
  //----------------------------------------
  Serial.println("---------------");
  Serial.println(toString_eeprom2);
  Serial.println("---------------");
  //----------------------------------------
```

```
  if (long_t2 == len_eeprom2)
  {
    if (toString_eeprom2.equals(t2))
    {
      flag2 = true;
      EEPROM.write(17, 1);
    }
    else
    {
      flag2 = false;
      EEPROM.write(17, 0);
    }
  }
  return flag2;
}

void xml_admin(EthernetClient cl)
{
  cl.print("<?xml version = \"1.0\" ?>");
  cl.print("<inputs>");
  cl.print("<pass1>");
  cl.print("ittg");
  cl.print("</pass1>");
  cl.println("</inputs>");
}
```

CONFIGURACIÓN DE LA MP3 SHIELD

FIGURA 35. DIAGRAMA DE FLUJO DE LA MP3 SHIELD

Se utilizó una shield MP3 porque esta tarjeta será la encargada de reproducir las canciones y tonos para el reloj.
La tarjeta contiene un decodificador que convierte archivos a alguno de los formatos con los que trabaja. En nuestro caso se usó para que convirtiera nuestros archivos a formato MP3. Dentro de la librería **<SFEMP3Shield.h>**, ya está la programación para la decodificación, por lo que solamente se mandó a llamar en nuestro código.
Y la librería **<SdFat.h>,** es necesaria para que la tarjeta identifique la presencia de un micro SD en ella.

CODIGO ARDUINO

```
#include <SPI.h>
#include <SdFat.h>
#include <SFEMP3Shield.h>
SdFat sd;
SFEMP3Shield MP3player;
byte address= 0x00;
int CS1= 10; //led

void setup() {

pinMode(CS1,OUTPUT);
SPI.begin();
uint8_t resultado;

Serial.begin(115200);

  if(!sd.begin(SD_SEL, SPI_HALF_SPEED)) sd.initErrorHalt();
  if(!sd.chdir("/")) sd.errorHalt("sd.chdir");


resultado = MP3player.begin();

if(resultado != 0)
  {
    Serial.print(F("Codigo de error: "));
    Serial.print(resultado);
    Serial.println(F(" al intentar iniciar reproductor de MP3"));
    if( resultado == 6 )
     {
      Serial.println(F("Advertencia: archivo de revisión no se encontró "));
      Serial.println(F("Use la opcion \"d\" para verificar si la SD esta leyendo los
archivos"));
    }
  }
  comandos();

}
void loop() {

  for(int i=0; i<=128; i++)
  {
```

```
  digitalPotWriteRed(i);
  delay(10);
  }
  delay(500);
  for (int i=128; i>=0;i--)
  {
    digitalPotWriteRed(i);
    delay(10);
  }
  delay(1000);

  if(Serial.available())
  {
    menu(Serial.read());

  }
  delay(100);
}


int digitalPotWriteRed(int value)
{
  digitalWrite(CS1,LOW);
  SPI.transfer(address);
  SPI.transfer(value);
  digitalWrite(CS1,HIGH);
}


void menu(byte opcion)
{
  uint8_t resultado;
  char titulo[30];
  char artista[30];

Serial.print(F("Comando Recebido: "));
  Serial.write(opcion);
  Serial.println();

if(opcion == 's')
  {
    Serial.println(F("Parar reproducion"));
    MP3player.stopTrack();
```

```
 }


 else if(opcion >= '1' && opcion <= 'n')
 {
   opcion = opcion - 48; //conversion ASCII para un numero real
   resultado = MP3player.playTrack(opcion);


 if(resultado != 0)
   {
     Serial.print(F("Codigo de Error: "));
     Serial.print(resultado);
     Serial.println(F(" cuando trata de reproducir musica"));
   }
   else
   {


 MP3player.trackTitle((char*)&titulo);
     MP3player.trackArtist((char*)&artista);

Serial.print(F("Titulo: "));
     Serial.write((byte*)&titulo, 30);
     Serial.println();
     Serial.print(F("Artista: "));
     Serial.write((byte*)&artista, 30);
     Serial.println();
     }
 }

else if((opcion == '-') || (opcion == '+'))
{
 union twobyte mp3_vol; //crear opcion variable existente que puede ser tanto de palabra como de doble byte de la izquierda y la derecha.
 mp3_vol.word = MP3player.getVolume(); //Leer el valor actual de volumen

 if(opcion =='-') //Observe dB es negativo
 {

if(mp3_vol.byte[1] >= 254)
     {
       mp3_vol.byte[1] = 254;
```

```
    }
    else {
      mp3_vol.byte[1] += 2;
    }
  }
  else
  {
    if(mp3_vol.byte[1] <= 2)
    {
      mp3_vol.byte[1] = 2;
    }
    else {
      mp3_vol.byte[1] -= 2;
    }
  }

  MP3player.setVolume(mp3_vol.byte[1], mp3_vol.byte[1]); // Cambia el nivel de volumen en ambos lados ( L y R )
   Serial.print(F("el volumen cambia a -"));
   Serial.print(mp3_vol.byte[1]>>1, 1);
   Serial.println(F("[dB]"));
 }

else if(opcion == 'd') {
   if(!MP3player.isPlaying())
   {
     Serial.println(F("Archivos encontrados:"));
     sd.ls(LS_R | LS_DATE | LS_SIZE);
   }
   else {
     Serial.println(F("reproduccion en curso, intente mas tarde"));
   }
 }

else if(opcion == 'i') {
   MP3player.getAudioInfo();
 }


else if(opcion == 'p')
 {
   if( MP3player.getState() == playback) //obtener estado y reproduce
   {
```

```
      MP3player.pauseMusic();
      Serial.println(F("Pausa"));
    }
    else if( MP3player.getState() == paused_playback)
    {
      MP3player.resumeMusic();
      Serial.println(F("reanudar"));
    }
    else
    {
      Serial.println(F("No esta reproduciendo"));
    }
  }


else if(opcion == 'r') {
    MP3player.stopTrack();
    MP3player.vs_init();
    Serial.println(F("Reset del chip"));
    }


else if(opcion == 'S') {
    Serial.println(F("Estado actual de VS10xx:"));
    Serial.print(F("isPlaying() = "));
    Serial.println(MP3player.isPlaying());

    switch (MP3player.getState())
    {
    case uninitialized:
      Serial.print(F("No inicializado"));
      break;
    case initialized:
      Serial.print(F("Inicializado"));
      break;
    case deactivated:
      Serial.print(F("Desactivado"));
      break;
    case loading:
      Serial.print(F("Cargando"));
      break;
    case ready:
      Serial.print(F("listo"));
```

```
      break;
    case playback:
      Serial.print(F("Reproduciendo"));
      break;
    case paused_playback:
      Serial.print(F("en pausa"));
      break;
    }
    Serial.println();
  }

else if(opcion == 'c')
  {
    comandos();
  }
  }
void comandos()
{
  Serial.println(F("MP3 Player Shield:"));
  Serial.println();
  Serial.println(F("ESCUELA SECUNDARIA DEL ESTADO VESPERTINA "));
  Serial.println();
  Serial.println(F("Lista de comandos:"));
  Serial.println(F(" [1-n] para reproducir un track"));
  Serial.println(F(" [s] Stop parar reproduccion"));
  Serial.println(F(" [d] Lista archivos de la SD card"));
  Serial.println(F(" [+ o -] Aumentar o diminuir el Volumen"));
  Serial.println(F(" [i] Informacoes sobre o archivos de audio"));
  Serial.println(F(" [p] Pausar/reproducir"));
  Serial.println(F(" [r] Reseta MP3 Shield"));
  Serial.println(F(" [S] Estado do Shield MP3"));
  Serial.println(F(" [c] Mostra lista de comandos"));
  Serial.println();
}
```

CAPITULO IV
RESULTADOS

Se realizó la carga del tiempo de una computadora a un RTC mediante el botón llamado ajuste automático, que extrae la hora y los minutos que tiene una computadora y los manda mediante una URL al servidor. Cuando llega el dato completo al servidor se separan los datos hora y minutos, siendo almacenadas en variables con el mismo nombre, estas variables son las responsables de actualizar el tiempo del RTC.
Usando XML y JavaScript se crea una página dinámica, que cuando llega un dato nuevo a la página, solo se actualiza un fragmento de ésta que contenga el nombre de la etiqueta a ser modificada. Gracias a esto se realiza la impresión de la hora y los minutos que tiene el RTC sobre la página.
Se hizo una página de acceso por contraseña para el modo usuario y administrador. Solo tiene conocimiento de las claves de acceso a una persona designada de la escuela secundaria del estado. Para lograr poder controlar el acceso por contraseña se hizo uso de la memoria EEPROM del arduino.
Se probó la Shield MP3 se tuvieron problemas manejando varios esclavos en la comunicación SPI, actualmente se busca una manera de poder reproducir canciones utilizando un segundo arduino, así uno podrá estar dedicado única y exclusivamente para el manejo de la Ethernet shield y otro para la shield MP3, basándonos en el esquema divide y vencerás.

La página de acceso para el usuario quedo de la siguiente manera.

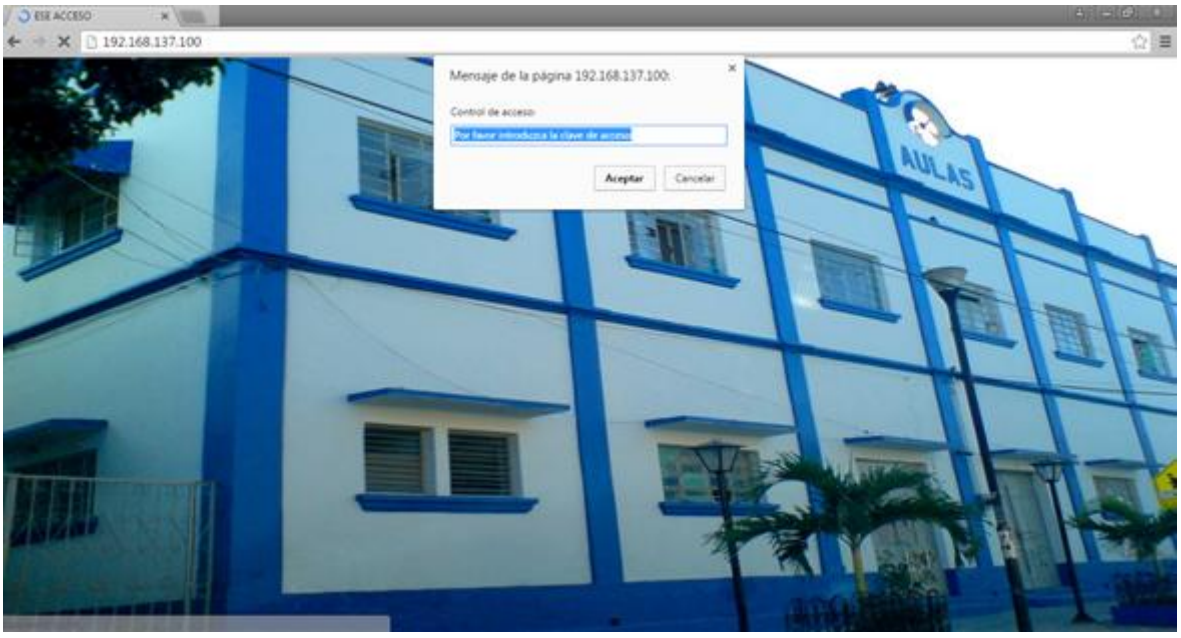


FIGURA 36. ACCESO DE USUARIO

La página de usuario se muestra enseguida

FIGURA 37. PAGINA DE USUARIO

Una vez accedido a la página podemos realizar ajustes de tiempo o cerrar sesión. Si se llega a cerrar sesión la página esperara un periodo de 5 segundos para poder realizar esa tarea. Se muestra el resultado en la siguiente imagen.

FIGURA 38. PAGINA DE BLOQUEO AL CERRAR SESION

Para ingresar al modo administrador basta con hacer clic sobre el link ubicado a la parte izquierda de la ventana. Posteriormente aparecerá una ventana que solicitara una contraseña. Como se muestra a continuación.

FIGURA 39. ACCESO AL ADMINISTRADOR

Al igual que en la primer página hay que ingresar la contraseña correcta para poder pasar al modo administrador.

CONCLUSIONES

Hacer un proyecto de residencia consiste en aplicar los conocimientos que a lo largo de la carrera nos enseñaron. Por ejemplo en nuestro proyecto se aplicó información acerca de redes. Para el control del reloj se necesitó conocimientos sobre sistemas embebidos como el uso de la tarjeta arduino entre otras más utilizadas, que fueron los encargados del control, estas se tuvieron que configurar, por medio de software, por lo que se aplicaron nuestros conocimientos en programación.

La elaboración de una página dinámica fue complicada debido a que no conocíamos cómo funcionaban y bajo qué lenguajes.

Debido a que el reloj se encuentra en la azotea del edificio y necesitamos que ésta pueda ser controlado desde la oficina de la directora de la secundaria, resulta muy conveniente el uso de una red inalámbrica, es por ello que se llegó a la conclusión de crear el servidor y una página de internet con la cual podremos realizar el ajuste del tiempo de reloj.

El uso del motor con el encoder se encuentra a nivel propuesta se sigue trabajando en su aplicación.

COMPETENCIAS DESARROLLADAS Y/O APLICADAS

Durante la elaboración de este proyecto, se entendió el funcionamiento de una estructura cliente-servidor, los métodos HTTP para el intercambio de información (GET y POST). Adquirimos conocimiento sobre el diseño de página web dinámicas, bajo el manejo de JavaScript, CSS y XML. En otras palabras el uso de JavaScript y XML se le conoce como AJAX (Asíncrono JavaScript XML). CSS sirve para mejor el estilo de página alterando las propiedades de las etiquetas.

REFERENCIAS BIBLIOGRAFICAS

- PROGRAMACIÓN WEB CON CSS, JAVASCRIPT, PHP Y AJAX
  Ing. Enrique E. Condor Tinoco
  Ing. Ivan Soria Solis

- ARDUINO SHIELD ETHERNET 2
  ARDUINO. ORG
  http://www.arduino.org/products/shields/5-arduino-shields/arduino-ethernet-shield-2

- SHIELD MP3 PLAYER
  SPARKFUN LEARN
  https://learn.sparkfun.com/tutorials/mp3-player-shield-hookup-guide-v15?_ga=1.180135012.1746191516.1448068223

- ARDUINO (PARA TODO LO REFERIDO A USO DE LAS LIBRERÍAS)
  https://www.arduino.cc/

ANEXOS

Dentro de este CD se anexa lo siguiente

- Carpeta con hojas de especificaciones de los circuitos y tarjetas de desarrollo.

- Programación con el programa arduino.

- Imágenes del reloj