INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ

Comisión Federal de Electricidad

Gerencia Regional de Transmisión Sureste
Zona de Transmisión Tuxtla

INFORME TÉCNICO DE RESIDENCIA

PROYECTO

Sistema de monitoreo y adquisición de datos de receptores VHF instalados en torres de líneas de 400 KV integrados a plataforma Zabbix.

PRESENTA
Ruiz Molina Rene Armando

ASESOR INTERNO
Ingeniero. Vicente León Orozco

AGOSTO – DICIEMBRE 2017

ÍNDICE

ÍNDICE DE FIGURAS

Capítulo I

1. introducción

Actualmente los sistemas de monitoreo de equipos electrónicos son muy importantes ya que con ello se pueden adquirir datos y después procesar la información obtenida para tener un control detallado del sistema, con la finalidad de hacer un análisis del comportamiento de los equipos en monitoreo; en caso de que alguno de ellos presente una anomalía en su funcionamiento, se podría realizar ajustes y correcciones, para prevenir fallas en sistemas que puedan ser importantes para los quipos que estén en funcionamiento.

Para la Empresa Productiva Subsidiaria (**EPS**) Transmisión, Zona de Transmisión Tuxtla, es importante tener monitoreado los equipos que se encuentran instalados en las casetas de comunicaciones, para tomar las acciones necesarias y prevenir fallas que se puedan presentar en los equipos de comunicaciones; estas variables a medir pueden ser temperatura del lugar, bajo voltaje, alto voltaje, corriente, puerta abierta, falla de VCA, VCD, etc. Por ello la importancia de enviar los datos censados hacia la Zona de transmisión Tuxtla donde se estará monitoreando constantemente. Tomando en cuenta, que muchos sitios se encuentran alejados de las instalaciones propias de dicha empresa, lo que dificulta las revisiones periódicas a cada uno de los sitios. Con esto se pretende reducir costos de gasolina, viáticos y tiempo extra por el traslado del personal hasta el sitio para la revisión de los equipos.

Para la Zona de Transmisión Tuxtla, se tiene una gran cobertura de Sistema **VHF** (Very High frequency) ya que se caracteriza por la transmisión confiable para él envió de información de un punto inicial a otro punto remoto (punto a punto), el sistema de radiocomunicación VHF permite hacer la comunicación de voz, en sitios donde es difícil contar con sistemas de comunicación convencional como es la telefonía fija o celular. Esto se realiza a través del uso de una infraestructura de Repetidores de VHF con la finalidad de poder llegar a sitios muy alejados.

Estos sitios de repetición de Sistema **VHF**, se encuentran instalados en lugares estratégicos, con el fin de cubrir toda la trayectoria de líneas de transmisión de alta tensión, que están bajo la responsabilidad de dicha empresa, con esto se garantiza tener comunicación directa con los centros de operación o reguladores de energía, además de tener comunicación en sitio para el mantenimiento a las estructuras de las líneas, para poder realizar maniobras específicas en la atención de fallas o trabajos relevantes que se realizan como parte del Control Físico de la Red Nacional de Transmisión, cumpliendo en todo momento con los requisitos de operación del Centro Nacional de Control de Energía (CENACE). Derivado a lo anterior, es importante que dichos sitios de repetición se encuentren monitoreados, por si se presenta algún tipo de falla que pueda afectar o interrumpir la comunicación y no tener disponible dicho servicio en toda la trayectoria de la línea.

En el presente trabajo se desarrolló un sistema de monitoreo y adquisición de datos de sitios repetidores de **VHF** instalados en torres de las líneas de 400 kV que será integrado a la plataforma de monitoreo **Zabbix**, el cual fue desarrollado por dicha empresa, haciendo uso de nuevas tecnologías, con el aprovechamiento de que dicho sistema consiste de un software libre que no requiere licenciamiento para su uso.

Se considera un proyecto de mucha relevancia y útil, ya que nos permite tener monitoreado el funcionamiento de los equipos que se encuentran instados en los sitios de repetición que están bajo la responsabilidad de la EPS Transmisión, Zona de Transmisión Tuxtla, la cual tiene la infraestructura necesaria para la transmisión de la información por Sistema **VHF**.

Los datos que se registren se enviarán al sistema de monitoreo **“Zabbix”,** que es el que se encargará de evaluar los parámetros si están en los rangos requeridos, realizando comprobaciones de ping y latencia, así como estadísticas del comportamiento de los equipos, notificando vía emails, WhatsApp o Telegram.

1.2. Información general de la CFE

La Comisión Federal de Electricidad es una empresa productiva del Estado de propiedad exclusiva del Gobierno Federal con personalidad jurídica y patrimonio propios, con forme a lo dispuesto en la nueva Ley de la Comisión Federal de Electricidad, y que tiene por objeto prestar, en términos de la legislación aplicable, el servicio público de Transmisión y Distribución de Energía Eléctrica, por cuenta y orden del Estado Mexicano para más de 39.6 millones de clientes, lo que representa más de 122 millones de habitantes e incorpora anualmente más de un millón de clientes nuevos.

El transporte de energía eléctrica es fundamental para el funcionamiento de un mercado de energía; es el punto de enlace entre la generación y la demanda, y el medio a través del cual se realizan los intercambios de energía eléctrica.

El 29 de marzo 2016 se publicó en el Diario Oficial de la Federación el acuerdo de creación de la Empresa Productiva Subsidiaria CFE Transmisión.

1.3. Visión 2018

Ser una empresa de servicio público de transmisión con un desempeño equiparable a las mejores empresas del mundo, con presencia internacional y fortaleza financiera, mediante el máximo aprovechamiento de su infraestructura y contribución de su capital humano.

1.4. Misión

Prestar servicio público de transmisión de energía eléctrica, mediante la operación, mantenimiento, expansión y modernización de la Red Nacional De Transmisión, garantizando un acceso abierto y no indebidamente discriminatorio y cumpliendo

con condiciones reguladas de disponibilidad, continuidad y eficiencia, para crear valor económico y rentabilidad para el Estado Mexicano.

1.5. Valores institucionales

- Integridad
- Productividad
- Responsabilidad

1.6. Objeto de la Empresa

Asegurar el acceso a la Red Nacional de Transmisión de la CFE, mediante su operación, mantenimiento, expansión y modernización, además de suministrar otros productos y servicios asociados para crear valor económico al Estado.

1.7. Imperativos Estratégicos

- Modernización de la Red Nacional de Transmisión.
- Lograr niveles de costo eficientes, en la operación y mantenimiento de la RNT.

1.8. Funciones

- Mantenimiento y Operación Física de la Red Nacional de Transmisión.
- Ampliar y Modernizar la Red Nacional de Transmisión.
- Celebrar Contratos de Interconexión con Generadores y de Conexión con Centros de Carga.
- Realizar Medición para liquidaciones para el Mercado Eléctrico Mayorista y emitir Facturación al CENACE de acuerdo a la tarifa regulada.

1.9 Retos

- Implementar la Gestión de Activos mediante el uso de la Red Eléctrica Inteligente, con el propósito de optimizar la vida útil de los activos de la Red Nacional de Transmisión.
- Cero Fallas en la Red Nacional de Transmisión.
- Desarrollo del Talento de los trabajadores de Transmisión, para el cumplimiento de las nuevas funciones.

1.10. Organigrama de la empresa de la CFE

La empresa se encuentra estructurada jerárquicamente en la Zona de Transmisión Tuxtla, su organigrama se muestra en la Figura 1.1.

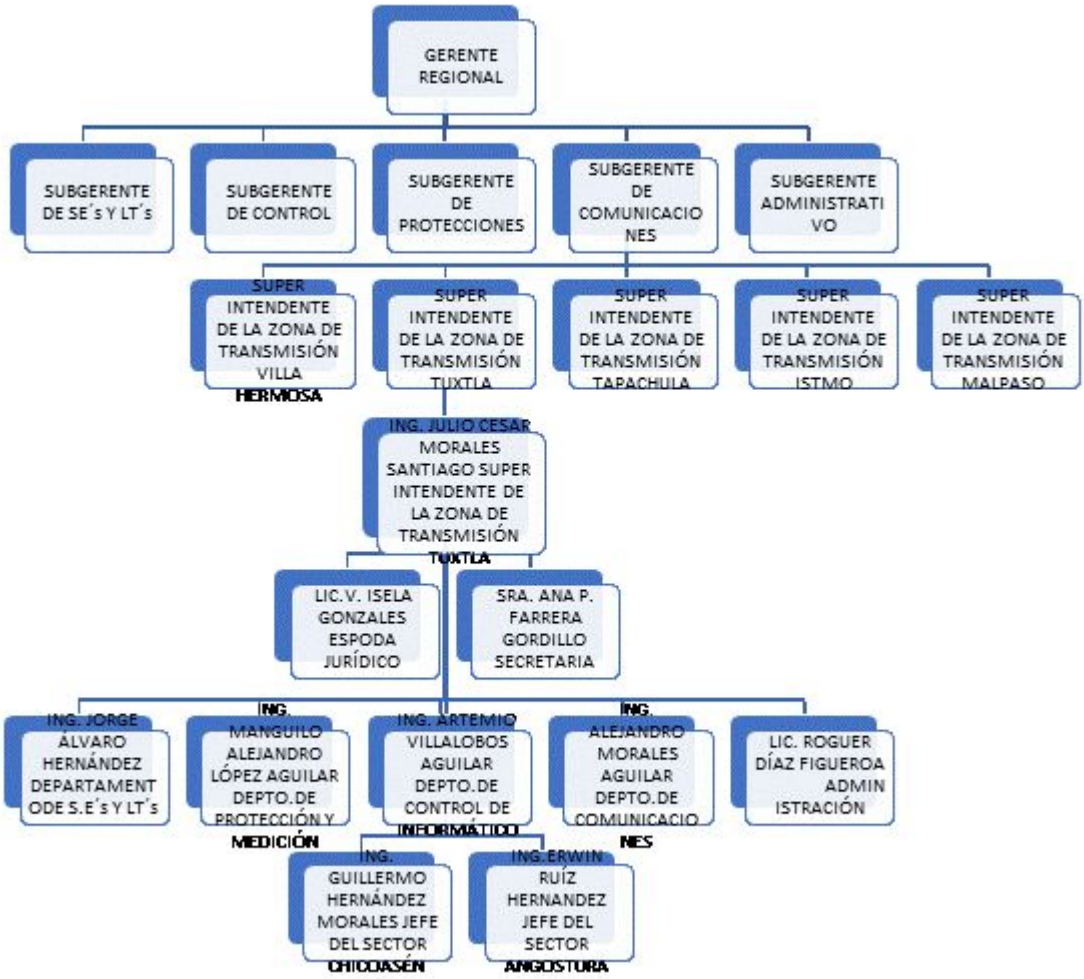


Figura 1.1: Organigrama de la empresa de la CFE:

1.11. Área específica relacionada directamente con el proyecto

La Empresa Productiva Subsidiaria (EPS) Transmisión, es columna vertebral del Sistema Eléctrico Nacional, ya que apoya a cumplir el compromiso de mantener, modernizar y rehabilitar las instalaciones como son las Subestaciones Eléctricas de Potencia que operan en los voltajes de 400, 230, 115 kV o voltajes iguales o superiores a 69 kV, ubicadas a lo largo y ancho de los casi 2 millones de kilómetros cuadrados que ocupa el territorio nacional; para lo cual se encuentra organizado en nueve Gerencias Regionales de Transmisión.

La Gerencia Regional de Transmisión Sureste **(GRTSE),** forma parte de la **EPS** Transmisión y tiene injerencia en los estados de Chiapas, Tabasco, Oaxaca y parte de Veracruz a través de sus cinco Zonas de Transmisión y una Zona de Operación que la conforman; asegurando en todo momento el servicio a sus clientes regionales y nacionales alrededor de 20 subestaciones eléctricas de potencia en voltajes de 400, 230 y 115 kV, mismas que están interconectadas y enlazadas al Sistema Eléctrico Nacional; y de acuerdo a la Ley de la Industria Eléctrica, publicado en el Diario Oficial de la Federación, tomará el control de la sección de las Subestaciones con tensión mayor o igual a 69 kV.

La Gerencia Regional de Transmisión Sureste **(GRTSE)**, tiene el control de Subestaciones de 400 kV, 230 kV y 115 kV, que forma parte del Sistema Eléctrico Nacional **(SEN)**, en donde las subestaciones deben estar supervisadas y tele controladas desde los centros de control del Centro Nacional de Control de Energía **(CENACE)**, con el fin de coordinar la operación del **SEN**. El proceso de transmisión requiere asegurar la confiabilidad física y operativa de sus instalaciones, orientada a mantener una alta seguridad de la red eléctrica.

La Zona de Transmisión Tuxtla, perteneciente a la Gerencia Regional de Transmisión Sureste, se encuentra en la ciudad de Tuxtla Gutiérrez, Chiapas. Atiende actualmente las Subestaciones Eléctricas Manuel Moreno Torres “Chicoasén”, Angostura y El Sabino, las cuales se encuentran conectadas al Sistema Interconectado Nacional mediante enlaces de 400 kV y a la red de 115 kV,

con la finalidad de suministrar la demanda de energía de las principales ciudades del Estado de Chiapas. Cuenta con 3 Secciones Sindicales del SUTERM: Sección 155 Chicoasén, Sección 130 Angostura y Sección 47 Tuxtla. Las oficinas Sedes están ubicadas en Carretera Panamericana km. 1077 No. 5675 Interior 300 mts. Col. Plan de Ayala de la Ciudad de Tuxtla Gutiérrez, como se muestra gráficamente en el mapa de la Figura 1.2.

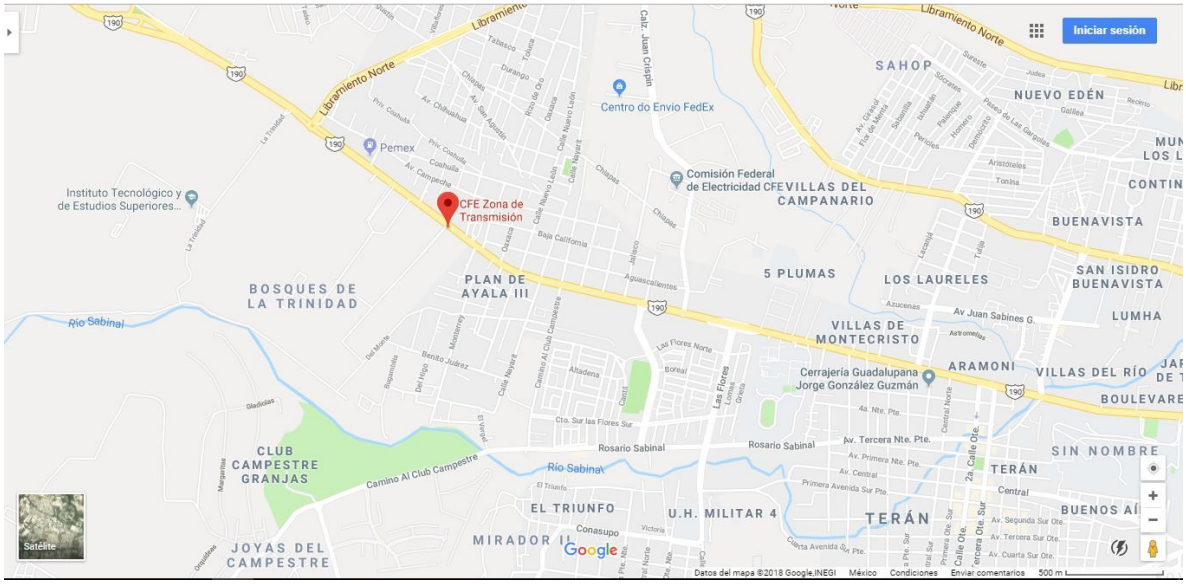


Figura. .1.2: Ubicación De La Empresa

La Zona de Transmisión Tuxtla, tuvo la necesidad y el compromiso de proporcionar servicios de Telecomunicaciones a clientes internos y externos, derivado a la importancia de la transmisión de datos y de establecer una buena comunicación, es necesario monitorear de manera constante el estado en que se encuentra los equipos que están dentro de las casetas de comunicaciones o sitios de Repetición de **VHF**, con la finalidad de detectar algún tipo de fallo en los equipos, emitiendo una alerta o notificación vía email, WhatsApp o Telegram para dar solución al problema y que los equipos sean restablecidos en tiempo y forma, cumpliendo en todo momento con mayor confiabilidad y seguridad.

Capítulo II

2.1 Antecedentes

En la escuela politécnica nacional se realizó un estudio para la migración del sistema de radiocomunicaciones **VHF** analógicos, a un sistema digital, donde habla sobre los sistemas **VHF** analógicos que están enfocados principalmente en proveer comunicación por voz.

Donde Algunos sistemas más avanzados proveen también servicios como: alarmas de emergencia, identificación de llamada, llamada selectiva e inhibición de radios. [1]

En la siguiente figura 2.1 se muestra un esquema típico de una instalación de un sistema **VHF** analógico y se describe brevemente sus componentes.

Figura 2.1: Diagrama del Sistemas VHF Analógico.

2.2 Planteamiento del problema

Actualmente las estaciones de los repetidores de **VHF**, bajo la responsabilidad de la Zona de Transmisión Tuxtla, no cuenta con un sistema que permita supervisar los equipos instalados, es por ello, que es necesario implementar un sistema de monitoreo y adquisición de datos, para tener un mejor control de los equipos que están en funcionamiento.
Derivado a los problemas que se pueden presentar en los sitios de Repetición en las torres de alta tensión, tales como temperatura, bajo voltaje o alto voltaje de VCD, puerta abierta y falta de voltaje, etc. donde se alojan los equipos de comunicaciones **VHF**, afectando la comunicación entre las subestaciones eléctricas y comunicación en la trayectoria de la línea de alta tensión, es importante poder desarrollar un control que pueda notificar en caso de alguna anomalía.

El desarrollo de este proyecto, ayudará en el monitoreo de los equipos de comunicaciones, que se encuentran instalados en los sitios de repetición de **VHF**, los cuales se encuentran alejados de la Zona de Transmisión Tuxtla. Este sistema permite obtener los datos y convertirlo en forma digital que será enviado por medio de Ethernet a una base de datos de la plataforma **Zabbix**.

2.3. Nombre del proyecto

Sistema de monitoreo y adquisición de datos de receptores **VHF** instalados en torres de líneas de 400 KV integrados a plataforma **Zabbix**.

2.4. Objetivo general

Desarrollar un sistema que permita el monitoreo y supervisión en tiempo real de equipos de radiocomunicación VHF instalados en torres de líneas de alta tensión.

2.5. Objetivos específicos

1. Implementar un prototipo con protocolo Modbus para establecer una comunicación de tipo maestro – esclavo.
2. Enviar las variables de monitoreo por una señal **DTMF** a los dispositivos de radios **VHF**.
3. Implementar una interfaz para establecer comunicación entre los sistemas embebidos en una tarjeta de desarrollo Arduino y la plataforma **Zabbix** para el almacenamiento y procesamiento de datos.

2.6. Justificación del proyecto

Los sitios que alojan los equipos Repetidores de radiofrecuencia de **VHF**, son de suma importancia para la empresa, ya que se requiere comunicación hacia las subestaciones que se encuentran alejadas de la Zona de Transmisión Tuxtla, así como también en toda la trayectoria de líneas de transmisión, los cuales son sitios de difícil acceso en donde no se tiene cobertura por telefonía convencional.

Cuando en los sitios de repetición, los equipos instalados sufren algún desperfecto no se cuenta con un sistema de monitoreo en tiempo real, que pueda notificar de alguna alarma, avisando que dichos equipos se encuentran vulnerables a algún desperfecto o en su caso dañado.

Debido a la problemática de no tener un sistema de monitoreo, no podemos saber cómo se encuentran operando los equipos repetidores de **VHF** y mucho menos poder realizar el envío de alarmas a través de la red de datos, ya que dicho sitio no se cuenta con infraestructura vía Ethernet, el cual impide que los datos se envíen a la plataforma **Zabbix** para su monitoreo en la Zona Tuxtla.

La finalidad de este proyecto es monitorear, censar y adquirir datos en tiempo real de los sitios de repetición de comunicación, para después enviar los datos censados mediante la señal de radiofrecuencia **VHF** (Very High Frequency) e integrarlo a la plataforma de desarrollo **Zabbix.**

El trabajo de nuestro sistema provoca un impacto económico dentro de la empresa, ya que al tener monitoreados los equipos de los sitios de repetición en líneas de alta tensión, instaladas en torres de línea de alta tensión, reduciendo las visitas a dichos sitios y trasladarse únicamente cuando se emita una alerta por alguna condición anormal.

2.7. Alcances del proyecto

Tener un sistema de monitoreo **VHF** de diferentes variables que son sumamente necesarias para el proceso de comunicación que la empresa requiere y que pueda informar las condiciones anormales del sistema, con la finalidad de tomar las acciones necesarias para solucionar los problemas que se presenten.

2.8. limitaciones del proyecto.

Dentro de las limitaciones que presenta es que la velocidad en la que transmite la información que es en orden de los 600-1200 baudios, las cuales no se consideran elevadas. Además de la topografía del terreno y la distancia de los sitios Repetidores, lo cual implica mejor precisión para lograr el alcance del proyecto.

2.9. Metodología del proyecto

1. Realizar una investigación teórica durante cuatro semanas en la cual proporcione información necesaria, para poder realizar el proyecto.
2. Verificar los materiales que se necesitan, para realizar el sistema de adquisición de variables.
3. Realizar diseño y simulación para el correcto funcionamiento y adquisición de datos del sistema de monitoreo.
4. Pruebas de operación, ajustes y anotaciones de los resultados obtenidos durante el desarrollo del proyecto.

Capítulo III

3. Fundamento teórico

El objetivo de un sistema electrónico de comunicaciones es transferir información entre dos o más lugares, cuyo nombre común es estaciones. Esto se logra convirtiendo la información original a energía electromagnética, para transmitirla a continuación a una o más estaciones receptoras, donde se reconvierte a su forma original. La energía electromagnética se puede propagar en forma de voltaje o corriente, a través de un conductor o hilo metálico, o bien en forma de ondas de radio emitidas hacia el espacio libre, o como ondas luminosas a través de una fibra óptica. La energía electromagnética se distribuye en un intervalo casi infinito de frecuencias.

La frecuencia no es más que la cantidad de veces que sucede un movimiento periódico, como puede ser una onda senoidal de voltaje o de corriente, durante determinado periodo. Cada inversión completa de la onda se llama ciclo. La unidad básica de frecuencia es el hertz (Hz), y un hertz es igual a un ciclo por segundo (1 Hz = 1 cps). En electrónica se acostumbra usar prefijos métricos para representar las grandes frecuencias. Por ejemplo, se usa el kHz (kilohertz) para indicar miles de hertz, y el MHz (megahertz) para indicar millones de hertz.

3.1 Frecuencias de transmisión

El espectro electromagnético de frecuencias total, donde se muestran los lugares aproximados de diversos servicios, se ve en la fig: 3.1. Este espectro de frecuencias va desde las subsónicas (unos pocos hertz) hasta los rayos cósmicos (1022 Hz). El espectro de frecuencias se subdivide en subsecciones o bandas. Cada banda tiene un nombre y sus límites. En los Estados Unidos, las asignaciones de frecuencias para radio propagación en el espacio libre son realizadas por la Comisión Federal de Comunicaciones (FCC). Por ejemplo, la banda de emisión comercial en FM tiene asignadas las frecuencias de 88 MHz a 108 MHz. Las frecuencias exactas asignadas a transmisores específicos que funcionan en las diversas clases de servicio se actualizan y alteran en forma constante, para cumplir con las

necesidades de comunicaciones en una nación. El espectro total útil de radiofrecuencias (RF) se divide en bandas de frecuencia más angostas, a las que se dan nombres y números descriptivos, y algunas de ellas se subdividen a su vez en diversos tipos de servicios. [1]

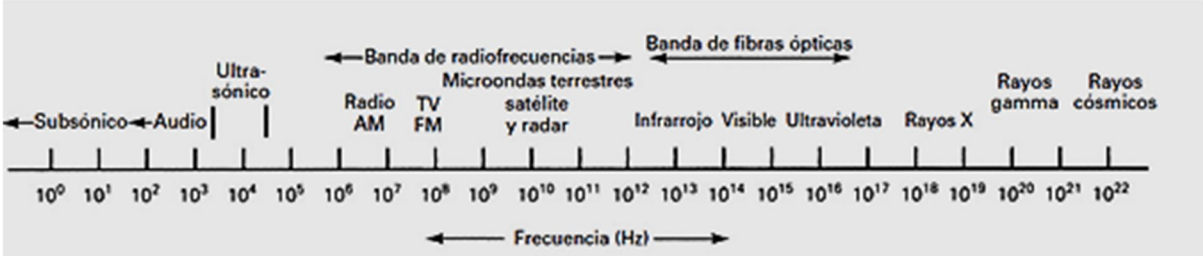


Figura 3.1: Espectro Electromagnético De Frecuencias

Las designaciones de banda según el Comité consultivo internacional de radio (CCIR) se muestran en la figura 3.2. Estas designaciones se resumen como sigue Frecuencias extremadamente bajas (ELF, de extremely low frequencies). Son señales en el intervalo de 30 a 300 Hz, y comprenden las señales de distribución eléctrica (60 Hz) y las de telemetría de baja frecuencia. Frecuencias de voz (VF, de voice frequencies). Son señales en el intervalo de 300 a 3000 Hz, e incluyen a las que generalmente se asocian a la voz humana. Los canales telefónicos normales tienen un ancho de banda de 300 a 3000 Hz, y con frecuencia se llaman canales de frecuencia de voz, o canales de banda de voz.

Frecuencias muy bajas (VLF, de very low frequencies). Son señales dentro de los límites de 3 a 30 kHz, que comprenden al extremo superior del intervalo audible humano. Las VLF se usan en algunos sistemas especiales, del gobierno y militares, como por ejemplo las comunicaciones con submarinos. Frecuencias bajas (LF, de low frequencies). Son señales en el intervalo de 30 a 300 kHz, y se usan principalmente en la navegación marina y aeronáutica. Frecuencias intermedias (MF, de medium frequencies). Son señales de 300 kHz a 3 MHz, y se usan principalmente para emisiones comerciales de radio AM (535 a 1605 kHz). Frecuencias altas (HF, de high frequencies). Señales en el intervalo de 3 a 30 MHz, con frecuencia llamadas ondas cortas. La mayoría de las radiocomunicaciones en dos sentidos usa este intervalo, y la Voz de América y la Radio Europa Libre

transmiten en él. También los radio aficionados y la banda civil (CB) usan señales de HF. Muy altas frecuencias (VHF, por very high frequencies). Son señales de 30 a 300 MHz, y se usan en radios móviles, comunicaciones marinas y aeronáuticas, emisión comercial en FM (de 88 a 108 MHz) y en la emisión de televisión, en los canales 2 a 13 (54a 216 MHz).

Número de banda	Intervalo de frecuencias*	Designación
2	30 Hz–300 Hz	ELF (frecuencias extremadamente bajas)
3	0.3 kHz–3 kHz	VF (frecuencias de voz)
4	3 kHz–30 kHz	VLF (frecuencias muy bajas)
5	30 kHz–300 kHz	LF (bajas frecuencias)
6	0.3 MHz–3 MHz	MF (frecuencias intermedias)
7	3 MHz–30 MHz	HF (frecuencias altas)
8	30 MHz–300 MHz	VHF (frecuencias muy altas)
9	300 MHz–3 GHz	UHF (frecuencias ultra altas)
10	3 GHz–30 GHz	SHF (frecuencias super altas)
11	30 GHz–300 GHz	EHF (frecuencias extremadamente altas)
12	0.3 THz–3 THz	Luz infrarroja
13	3 THz–30 THz	Luz infrarroja
14	30 THz–300 THz	Luz infrarroja
15	0.3 PHz–3 PHz	Luz visible
16	3 PHz–30 PHz	Luz ultravioleta
17	30 PHz–300 PHz	Rayos X
18	0.3 EHz–3 EHz	Rayos gamma
19	3 EHz–30 EHz	Rayos cósmicos

*10^0, hertz (Hz); 10^3, kilohertz (kHz); 10^6, megahertz (MHz); 10^9 gigahertz (GHz); 10^{12}, terahertz (THz); 10^{15}, petahertz (PHz); 10^{18} exahertz (EHz)

Figura 3.2: Designaciones De Banda CCIR

Frecuencias ultra altas (UHF, de Ultrahigh Frequencies). Son señales entre los límites de 300 MHz a 3 GHz, y las usa la emisión comercial de televisión, en los canales 14 a 83, en los servicios móviles de comunicaciones terrestres, teléfonos celulares, algunos sistemas de radar y de navegación, y los sistemas de radio por

microondas y por satélite. Hablando con generalidad, se considera que las frecuencias mayores que 1 GHz son de microondas, y eso incluye al extremo superior del intervalo de UHF. Frecuencias super altas (SHF, por Superhigh Frequencies). Son señales de 3 a 30 GHz, donde está la mayoría de las frecuencias que se usan en sistemas de radiocomunicaciones por microondas y satelitales. Frecuencias extremadamente altas (EHF, de extremely high frequencies). Son señales entre 30 y 300 GHz, y casi no se usan para radiocomunicaciones, a excepción de aplicaciones muy complicadas, costosas y especializadas. Infrarrojo. Las frecuencias del infrarrojo son señales de 0.3 a 300 THz, y por lo general no se les considera como ondas de radio. [2]

3.2. Ancho de banda

Las dos limitaciones más importantes en el funcionamiento de un sistema de comunicaciones son el ruido y el ancho de banda. El ancho de banda de una señal de información no es más que la diferencia entre las frecuencias máxima y mínima contenidas en la información, y el ancho de banda de un canal de comunicaciones es la diferencia entre las frecuencias máxima y mínima que pueden pasar por el canal (es decir, son su banda de paso). El ancho de banda de un canal de comunicaciones debe ser suficientemente grande (ancho) para pasar todas las frecuencias importantes de la información. En otras palabras, el ancho de banda del canal de comunicaciones debe ser igual o mayor que el ancho de banda de la información. Por ejemplo, las frecuencias de voz contienen señales de 300 a 3000 Hz. Si un sistema de transmisión de televisión por cable tiene una banda de paso de 500 a 5000 kHz, su amplitud de banda es 4500 kHz. Como regla general, un canal de comunicaciones no puede propagar una señal que contenga una frecuencia que cambie con mayor rapidez que la amplitud de banda del canal. [1]

3.3. Modulación

La modulación engloba el conjunto de técnicas para transportar información sobre una onda portadora, típicamente una onda sinusoidal. Estas técnicas permiten un mejor aprovechamiento del canal de comunicación lo que posibilita transmitir más información en forma simultánea, protegiéndola de posibles interferencias y ruidos. La modulación consiste en hacer que un parámetro de la onda portadora cambie de valor de acuerdo con las variaciones de la señal moduladora, que es la información que queremos transmitir.

Existen dos tipos de modulación: modulación analógica, que se realiza a partir de señales analógicas de información, por ejemplo, la voz humana, audio y video en sus formas eléctricas, y la modulación digital que se lleva a cabo a partir de señales generadas por las fuentes digitales, por ejemplo, una computadora, ver figura 3.3.

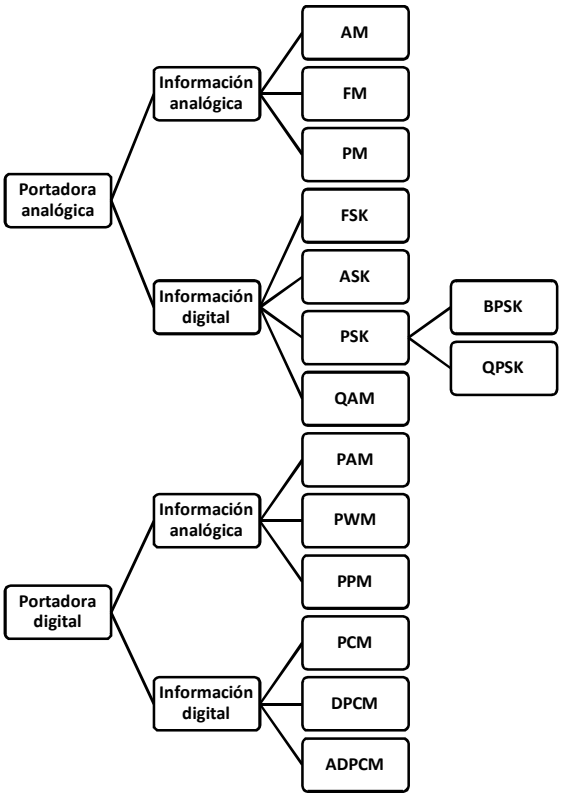


Figura 3.3: Tipos de Modulación

La manipulación por desplazamiento de frecuencia (FSK, de frequency-shift keying) es otro tipo relativamente sencillo y de baja eficiencia de modulación digital. En la figura 3.4 de pude ver las entras y salidas una señal. La FSK binaria es una forma

de modulación de ángulo, de amplitud constante, parecido a la modulación convencional de frecuencia (FM), pero la señal moduladora es una señal binaria que varía entre dos valores discretos de voltaje, y no es una forma de onda analógica que cambie continuamente. [1]

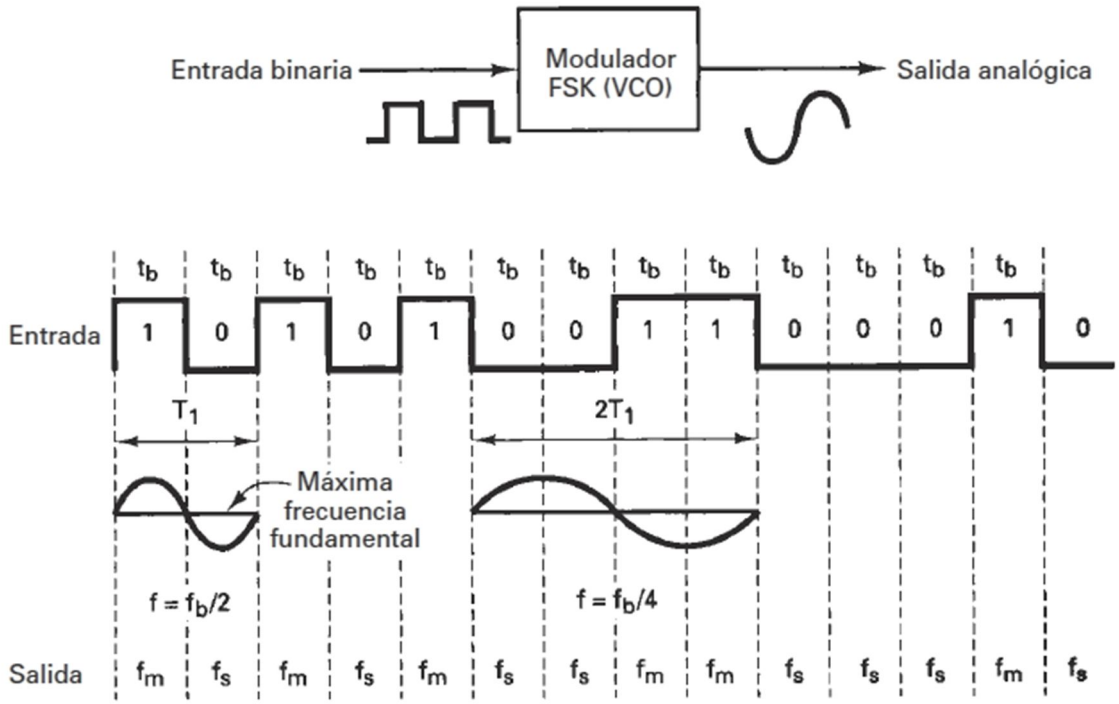


Figura 3.4: Formas de onda de entrada de entradas y salidas binaria FSK

3.4. VHF

Las redes privadas de comunicación de voz en banda VHF utilizan la banda de frecuencia de 30-300 MHz. En ellos se alcanzan distancias de enlace en torno a los 70 Km, limitados por la potencia de transmisión y la altura de las antenas. Éstas deberán compensar la curvatura de la tierra y salvar los obstáculos que se presentan en el camino, aunque tiene bastante tolerancia a los mismos. En la propagación directa desde la antena transmisora a la antena receptora es recomendable que exista “línea de vista” entre ellas, es decir, que exista visibilidad óptica entre ambas. Sin embargo, se soportan obstáculos vegetales o invasiones no muy profundas de la línea de vista por elevaciones del terreno.

El inconveniente de no lograr un enlace debido a obstrucción severa de la línea de vista puede superarse utilizando equipos intermedios o repetidores, usualmente ubicados en zonas elevadas, de forma que permitan la comunicación, a través de ellos, entre dos o más puntos que no tienen visibilidad directa. Aunque esta banda está pensada solamente para la transmisión de voz y, por tanto, los equipos de radio se diseñan y fabrican para ese fin, mediante software se puede conseguir utilizar este medio para comunicaciones de datos. Existen diferentes tipos de herramientas software para la transmisión de datos. El más eficiente de ellos es el protocolo AX.25 que incluso permite instalar el protocolo TCP/IP sobre él. AX.25 es un protocolo de nivel de enlace habitualmente usado por radio aficionados para bandas VHF/UHF y HF. Aunque la velocidad que se consigue es muy baja, apenas comparable a la velocidad de un módem telefónico, puede aumentar con la compresión que incorpora el sistema de correo, permitiendo utilizar aplicaciones de correo electrónico, mensajería y navegación (restringida) en Internet a velocidades aceptables. [1]

Ventajas:

- Enlaces a largas distancias. Aunque requiere línea de vista pueden salvarse algunos obstáculos vegetales o no muy profundos. Estos enlaces suelen implicar menor número de emplazamientos aislados necesarios para conectar establecimientos.
- Fácil reutilización de frecuencias.
- Tecnología radio muy conocida en los entornos rurales.
- La calidad de los enlaces es similar 24 horas al día al no verse especialmente afectada la propagación por los cambios climatológicos.

Inconvenientes:

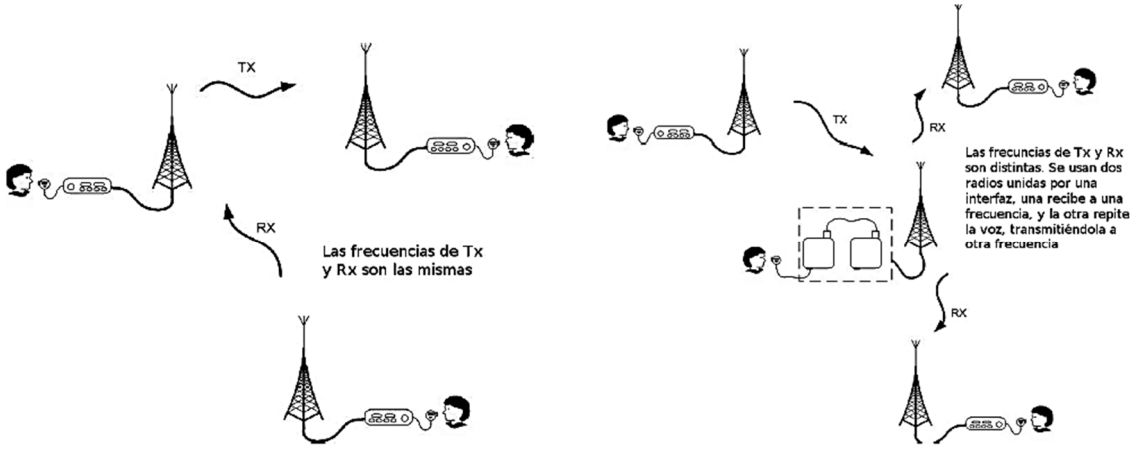
- El uso de la banda VHF requiere de la obtención oficial de una licencia de servicio.

- Velocidades menores que para otras tecnologías como WiFi.
- Mayor consumo en torno a los 100 W en transmisión frente a los menos de 10 W requeridos para transmitir con una tecnología WiFi (considerando el consumo completo de un enrutador inalámbrico).
- Al requerirse potencias mayores, se tiene que proveer al sistema de paneles solares de superficie mucho mayor y baterías de mayor capacidad, lo que eleva su costo.

3.5. Comunicaciones de voz

La comunicación de voz es el servicio natural de los sistemas VHF. Cada red utiliza un canal para la comunicación de voz entre sus miembros, como se aprecia en la Figura 3.5(a). En caso que existan estaciones bastante alejadas entre sí, se usan repetidores de voz, por lo que la red utiliza dos canales de voz, uno para transmisión y otro para recepción, como se muestra en la Figura 3.5(b).

Cuanto más aumente la cantidad de repetidores de voz, mayor será el número de canales usados. [3]



(a) *Voz en VHF sin Repetidor de voz.* (b) *Voz en VHF con Repetidor de voz*

Figura 3.5: Voz en VHF

Todos estos canales han de ser configurados en la radio del repetidor de voz que de servicio a cada estación y a través del cual se retransmiten las comunicaciones hacía su destino, en particular, hacia la estación que cuente con salida a la RTPC. Esta estación será denominada estación pasarela. Al igual que en la comunicación de datos, todas las estaciones tienen la posibilidad de establecer comunicaciones de voz hacia el exterior a través de una de ellas, que está conectada a la RTPC. Sin embargo, dada la baja calidad de las comunicaciones HF, únicamente las estaciones conectadas con enlaces VHF tienen la posibilidad de ofrecer de este servicio con una calidad aceptable. Las estaciones conectadas mediante enlaces HF no se pueden comunicar con los sistemas VHF, que usan otra banda de frecuencia.

3.6. Protocolo Modbus

Modbus es un protocolo industrial que fue desarrollado en 1979 para hacer posible la comunicación entre dispositivos de automatización. Originalmente implementado como un protocolo al nivel de la aplicación con la finalidad de transferir datos por una capa serial, Modbus se ha expandido para incluir implementaciones a través de protocolo serial, TCP/IP y el User Datagram Protocol (UDP).

Modbus es un protocolo de solicitud-respuesta implementado usando una relación maestro-esclavo figura3.6. En una relación maestro-esclavo, la comunicación siempre se produce en pares, un dispositivo debe iniciar una solicitud y luego esperar una respuesta y el dispositivo de inicio (el maestro) es responsable de iniciar cada interacción. Por lo general, el maestro es una interfaz humano-máquina (HMI) o sistema SCADA y el esclavo es un sensor, controlador lógico programable (PLC) o controlador de automatización programable (PAC). El contenido de estas solicitudes y respuestas, y las capas de la red a través de las cuales se envían estos mensajes, son definidas por las diferentes capas del protocolo. [3]

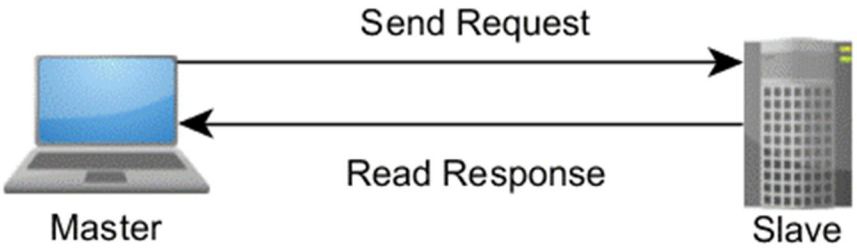


Figura 3.6: Relación de Red Maestro-Esclavo

3.7. Acceso de Datos en Modbus y el Modelo de Datos de Modbus

Los datos disponibles por medio de Modbus son almacenados, en general, en uno de los cuatro bancos de datos o rangos de dirección: bobinas, entradas discretas, registros de retención y registros de entrada. Al igual que con gran parte de la especificación, los nombres pueden variar dependiendo de la industria o de la aplicación. Por ejemplo, los registros de retención pueden denominarse como registros de salida y las bobinas pueden denominarse como salidas digitales o discretas. Estos bancos de datos definen el tipo y los derechos de acceso de los datos contenidos. Los dispositivos esclavos tienen acceso directo a estos datos, los cuales son alojados localmente en los dispositivos. Los datos disponibles por medio de Modbus generalmente son un subconjunto de la memoria principal del dispositivo. En contraste, los maestros Modbus deben solicitar el acceso a estos datos a través de diversos códigos de función. El comportamiento de cada bloque se describe en la Figura 3.7. [3]

Bloque de Memoria	Tipo de Datos	Acceso de Maestro	Acceso de Esclavo
Bobinas	Booleano	Lectura/Escritura	Lectura/Escritura
Entradas Discretas	Booleano	Solo Lectura	Lectura/Escritura
Registros de Retención	Palabra Sin Signo	Lectura/Escritura	Lectura/Escritura
Registros de Entrada	Palabra Sin Signo	Solo Lectura	Lectura/Escritura

Figura 3.7: Bloque De Datos De Modbus

Estos bloques le brindan la habilidad de restringir o permitir el acceso a los diferentes elementos de datos y también de proporcionar mecanismos simplificados en la capa de aplicación para tener acceso a diferentes tipos de datos.

Los bloques son completamente conceptuales. Pueden existir como direcciones de memoria separadas en un sistema determinado, pero también pueden traslaparse. Por ejemplo, la bobina uno puede existir en la misma ubicación en memoria como el primer bit de la palabra representada por el registro de retención uno. El esquema de dirección se define completamente por el dispositivo esclavo y su interpretación de cada bloque de memoria es una parte importante del modelo de datos del dispositivo.

3.8. Arduino Mega 2560

Arduino es una plataforma de electrónica de código abierto basada en hardware y software fácil de usar. El Arduino Mega (Figura 3.8) está basado en el microcontrolador ATMega2560. [4]

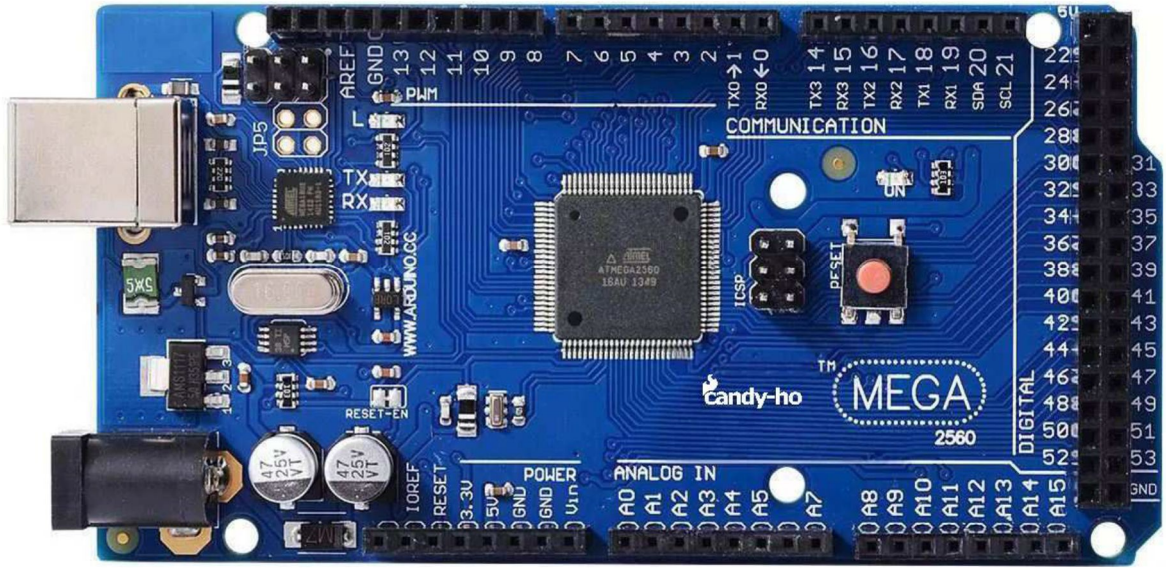


Figura 3.8: Arduino Mega 2560

3.8.1. Características de la Tarjeta Arduino

Dispone de 54 entradas/salidas digitales, 14 de las cuales se pueden utilizar como salidas PWM (modulación de anchura de pul so). Además, dispone de 16 entradas analógicas, 4 UARTs (puertas series), un oscilador de 16MHz , una conexión USB, un conector de alimentación, un conector ICSP y un pulsador para el reset. Para empezar a utilizar la placa sólo es necesario conectar la al ordenador a través de un cable USB, o alimentar la con un adaptador de corriente AC/DC. También, para empezar, puede alimentarse mediante una batería. Una de las diferencias principales de la tarjeta Arduino MEGA 2560 es que no utiliza el convertidor USB-serie de la firma FTDI. Por lo contrario, emplea un microcontrolador Atmega8U2 programado como actuar convertidor USB a serie. Esta placa debido a su gran poder es utilizada para grandes proyectos, entre los más importantes se encuentras los de DOMOTICA y IMPRESORAS 3D.

El Arduino MEGA2560 es compatible con la mayoría de los shield o tarjetas de aplicación/ampliaciones disponibles para las tarjetas Arduino UNO original.

- Las características principales son:
- Microprocesador ATmega2560
- Tensión de alimentación (recomendado) 7-12V
- Integra regulación y estabilización de +5Vcc
- 54 líneas de Entradas/Salidas Digitales (14 de ellas se pueden
- utiliza como salidas PWM)
- 16 entradas Analógicas
- Máxima corriente continua para las entradas: 40 mA
- Salida de alimentación a 3.3V con 50 mA
- Memoria de programa de 256Kb (el bootloader ocupa 8Kb)
- Memoria SRAM de 8Kb para datos y variables del programa
- Memoria EEPROM para datos y variables no volátiles
- Velocidad del reloj de trabajo de 16MHz
- Reducidas dimensiones de 100 x 50 mm

3.8.2. Datos técnicos de la Tarjeta Arduino

El Arduino mega puede ser alimentado vía conexión USB o con una fuente de alimentación externa. El origen de la alimentación se selecciona automáticamente. Las fuentes de alimentación externas (no -usb) pueden ser tanto un transformador o una batería. El transformador se puede conectar usando un conector macho de 2.1mm con centro positivo en el conector hembra de la placa. El cable de la batería puede conectarse a los pines gnd y vin en los conectores de alimentación (power) la placa puede trabajar con una alimentación externa de entre 6 a 20 voltios. si el voltaje suministrado es inferior a 7v, el pin de 5v puede proporcionar menos de 5 voltios y la placa puede volverse inestable; si se usan más de 12v los reguladores de voltaje se pueden sobre calentar y dañar la placa. el rango recomendado es de 7 a 12 voltios. los pines de alimentación son los siguientes:

- VIN: La entrada de voltaje a la placa Arduino cando se está usando una fuente externa y alimentación (en opuesto a los 5 voltios de la conexión USB). Se puede proporcionar voltaje a través de este pin, o, si se está alimentando a través de la conexión de 2.1mm, acceder a ella a través de este pin.
- 5V: La fuente de voltaje estabilizado usado para alimentar el microcontrolador y otros componentes de la placa. Esta puede provenir de VIN a través de un regulador integrado en la placa, o proporcionada directamente por el USB u otra fuente estabilizada de 5V.
- 3.3V: Una fuente de voltaje de 3.3 voltios generada por un regulador integrado en la placa. La corriente máxima soportada 50mA.
- GND: Pines de toma de tierra.

3.9. Arduino Ethernet Shield HR911105A

El Arduino Ethernet Shield permite a una placa Arduino conectarse a internet. Está basada en el chip ethernet Wiznet W5100. El Wiznet W5100 provee de una pila de

red IP capaz de TCP y UDP. Soporta hasta cuatro conexiones de sockets simultáneas. Usa la librería Ethernet para escribir programas que se conecten a internet usando la shield. Figura 3.9.

Es compatible con el Arduino UNO y Arduino Mega. El shield provee un conector ethernet estándar RJ45 y un conector lector de tarjeta Micro SD. El botón de reset en la shield resetea ambos, el W5100 y la placa Arduino. [6]

Figura 3.9: Arduino Ethernet Shield HR911105A

El shield contiene un número de LEDs para información:

- PWR: indica que la placa y la shield están alimentadas
- LINK: indica la presencia de un enlace de red y parpadea cuando la shield envía o recibe datos
- FULLD: indica que la conexión de red es full duplex
- 100M: indica la presencia de una conexión de red de 100 Mb/s (de forma opuesta a una de 10Mb/s)
- RX: parpadea cuando la shield recibe datos
- TX: parpadea cuando la shield envía datos

- COLL: parpadea cuando se detectan colisiones en la red

3.10. XR-2206

Es un circuito integrado generador de función monolítica capaz de generar o producir alta calidad de señales (seno, cuadrada, triangular, rampa y pulso en forma de ola), de estabilidad y exactitud. El resultado en forma de ola puede ser ambas amplitud y frecuencia modular por parte externa, la frecuencia de operación puede ser seleccionada externamente en más de un rango de 0.01Hz a más de 1Mhz. Como de observa en la figura 3.10. [2]

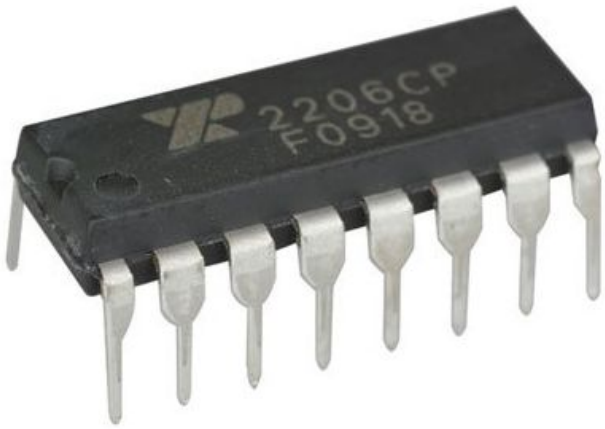


Figura 3.10: XR-2206

3.10.1. Características de circuito XR-2206

- Distorsión de onda seno baja: 0.5% típica con ajuste, y 2.5 % típica sin ajuste. (Hasta 200kHz).
- Ancho de banda: 0.01Hz a 1MHz

- Rango de barrido amplio: 2000:1 típico
- Baja sensibilidad a cambios en la alimentación: 0.01%V típico
- Ciclo de trabajo ajustable entre 1% y 99%
- Salida de sincronismo
- Voltaje de alimentación: ±5V a ±13V fuente dual (10V a 26V fuente sencilla)
- Encapsulado: DIP 16 pines
- Producto genuino

3.10.2. Aplicaciones de circuito XR-2206

- Generador de ondas seno, triangular, rampa (diente de sierra), cuadrada y pulsos
- Generación de AM / FM / ASK / FSK
- Generador de barrido
- Conversor de voltaje a frecuencia (V/F converter)
- Osciladores controlados por voltaje (VCO)
- Entre otros.

3.11. XR-2211

Imposición de una señal a una onda portadora alternante de manera que la amplitud de la portadora permanece constante, mientras que su frecuencia varía proporcionalmente con la amplitud de la señal. El proceso inverso de modulación consiste en la recuperación de la señal moduladora (o información) de una onda portadora después de pasar por el medio de transmisión. Como se observa en la Figura 3.11. [3]

Figura 3.11: XR-2211

3.12. Características de circuito XR-2211

- Frecuencia de operación: 0.01 Hz a 300 kHz
- Voltaje de alimentación: 4.5 V a 20 V
- Compatibilidad con lógica HCMOS y TTL
- Rango dinámico amplio: 10 mV a 3 V rms
- Rango de tracking ajustable: ±1% a 80%
- Excelente estabilidad con la temperatura: 100 ppm/ºC típicamente
- Encapsulado: DIP 14 pines

3.13. Aplicaciones de circuito XR-2211

- Demodulación FSK
- Detección de portadora
- Detección de FM
- Decodificación de tonos

- Identificadores de llamadas
- Sincronización de datos
- Entre otros

3.14. Sensor de voltaje FZ0430

El FZ0430 es un módulo comercial que nos permite medir tensiones de hasta 25V de forma sencilla con un procesador como Arduino, como se muestra en la figura3.12. FZ0430 es un simple divisor de tensión con resistencias de 30kOhm y 7.5kOhm, lo que supone que la tensión percibida tras el módulo sea de divida por un factor de 5 (7.5/(30+7.5)).

Por tanto, la tensión máxima que podemos medir será 25V para un procesador de tensión de alimentación Vcc 5V, y 16.5V para un procesador de Vcc 3.3V. Superar esta tensión en el input del FZ0430 dañará el pin analógico de Arduino. [9]

Por supuesto, esta ampliación del rango de medición tiene una consecuencia negativa en la precisión de la medición. En los modelos de Arduino que incorporan un ADC de 10 bits alimentados a 5V, la resolución normal es de 4.88mV. Tras el FZ0430 la resolución de la medición es de 24.41mV.

En caso de emplear una carga de 25V, la corriente que atraviesa el divisor es de 0.7mA, y las pérdidas del divisor 16.67mW.

Figura 3.12: Sensor de voltaje FZ0430

3.15. Módulo de Reloj en Tiempo Real DS1307

Este Módulo de Reloj en Tiempo Real (RTC) emplea el DS1307 para mantener el registro del año, mes, día, así como el tiempo actual. El DS1307 se accede a través del protocolo I2C. figura 3.13. [10]

Figura 3.13: DS1307

3.15.1. Especificaciones de Reloj

- Protocolo: I2C
- Interfaz de dos alambres
- Completamente ensamblado
- Hora: Minutos: Segundos AM/PM.
- Día, Mes, Fecha – Año.
- RTC basado en DS1307
- Pin de salida de 1Hz
- 56 bytes de memoria disponible para el usuario
- Tamaño: 2.9cm x 2.6cm – 1.14” x 1.02”

3.16. Motorola GM300

Radios GM300 funcionan con una frecuencia de walkie-talkie a una distancia 9.65KM. Los usuarios tienen acceso a ocho o 16 canales con una radio figura 3.14. Los usuarios pueden conectar múltiples radios para la operación de banda cruzada y el acceso a una amplia gama de canales de comunicación. El GM300 recibe las

siguientes frecuencias: muy alta frecuencia: 136 a 162 megahercios / 146-174 megahertz y ultra alta frecuencia: 403-433 MHz / 438-470 MHz / 465-495 MHz / 490-520 megahertz. [8]

Figura 3.14: GM300

3.16.1. Capacidades de programación GM300

GM300 usuarios pueden programar las líneas privadas de libre disposición y líneas privadas digitales. Canales con la actividad se controlan a través de una búsqueda de canales prioridad. Una característica de tiempo de espera desconecta canales que alcanzan su límite de transmisión. El GM300 tiene una reputación entre los usuarios por ser fácilmente programable.

3.17. Plataforma Zabbix

Zabbix es un sistema para monitorear la capacidad, el rendimiento y la disponibilidad de los servidores, equipos, aplicaciones y bases de datos. Además, ofrece características avanzadas de monitoreo, alertas y visualización, que incluso, algunas de las mejores aplicaciones comerciales de este tipo no ofrecen. [9]

Características:

- Monitoreo centralizado a través del administrador web.
- Disponible para diferentes sistemas operativos linux, windows, mac os, entre otros.
- envío de alertas vía correo electrónico.
- envío de alertas vía correo electrónico.
- Agregar y monitorear servidores, equipos, servicios, aplicaciones específicas, dispositivos físicos como impresoras, routers, entre otros.
- Reporte en tiempo real a través de gráficas, datos y alertas visuales que muestran el estado y rendimiento de los servicios y equipos monitoreados.
- Inventario de equipos para mantener al día la infraestructura tecnológica.
- Mapas de la red de la empresa.
- Configuración de notificaciones vía correo electrónico.
- Perfiles de usuarios para el uso del administrador Web.

Ventajas:

- Interfaz basada en la web.
- Reportes detallados.
- Fácil configuración.
- Estadísticas en tiempo real del estado de los servidores.
- Reduce los costos de operación al evitar el tiempo de inactividad.

Capítulo IV

4. Desarrollo e implementación del proyecto

En el siguiente diagrama bloque se muestra los pasos que tiene que realizar el proyecto para adquirir las variables sensadas, desde que el maestro solicite alguna respuesta o cuando se presenta algún evento que este fuera de lo normal, como temperatura alta 35°, temperatura baja 15°, temperatura la temperatura normal se encuentra en el rango de 15° a 35°, alto voltaje 14 voltios y bajo voltaje 11 voltios, el volteje normal se encuentra en el rango de 11 y 14 voltios, puerta abierta, puerta cerrada, esta le enviara una señal al Arduino para después procesarlo y enviarle en un señal digital al modem que hará un proceso de conversión Digital – Analógico y Analógico – Digital que será enviada a través de los radios por medio de la señal de **VHF.**

Que será recibida por el Arduino que tendrá la función de Maestro donde este recibirá la señal que se envió de los sensores el Arduino maestro enviara los datos sensados a la plataforma **Zabbix ,** donde se mantendrá el monitoreo constante de los sensores y así saber que no existe ningún problema en la caso se exista un comportamiento fuera de normal este enviara un mensaje a través de msm,whatsapp o correo electrónico para dar solución al problema de la manera más rápida posible para evitar daños considerables en el sistema. Figura 4.0

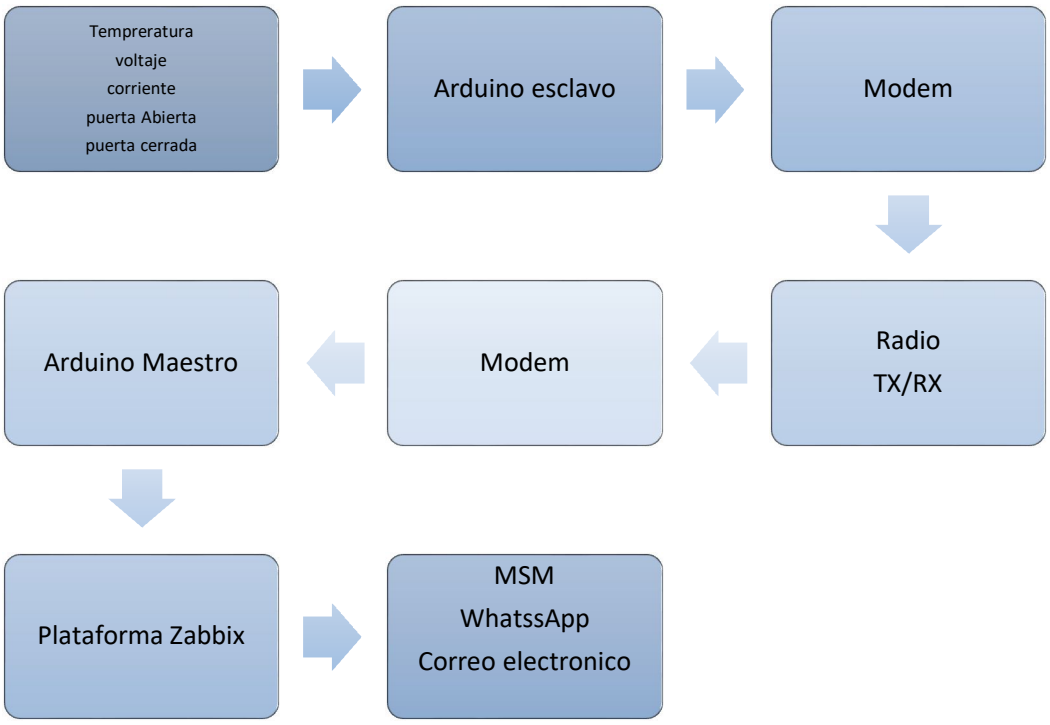


Figura 4.0: Digrama bloque del proyecto

4.1 Simulación del proyecto.

Para la construcción del proyecto se realizó la simulación en el software Isis Proteus que nos ayudará para hacer pruebas y ajustes necesarios, que se requieran para el diseño y construcción del proyecto sin el riesgo de dañar el circuito, si eso llegara a ocurrir.

En la figura 4.1 se observa la conexión que se realizó para hacer que la tarjeta Arduino pueda realizar la comunicación entre el sensor de temperatura LM35 en donde se estará adquiriendo los datos para que después se puede procesar la información y enviarlos al Arduino maestro, que se encargar de enviar los datos a la plataforma de monitoreo **Zabbix.**

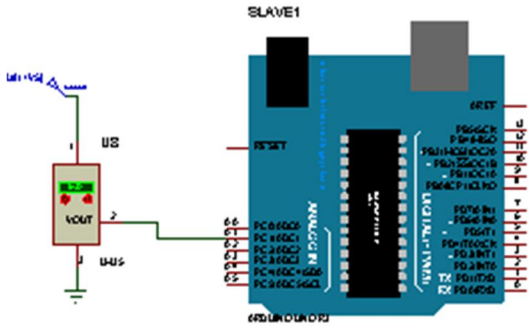


Figura 4.1: Esclavo 1 conexión de sensor de temperatura

Para medir el banco de baterías se realizó la simulación de un circuito capaz de medir el voltaje y la carga de una batería, que pueda indicar en forma de porcentaje la carga restante de las baterías. En la figura 4.2 se observa la conexión que se hizo en el software Isis Proteus para la simulación del sensor.

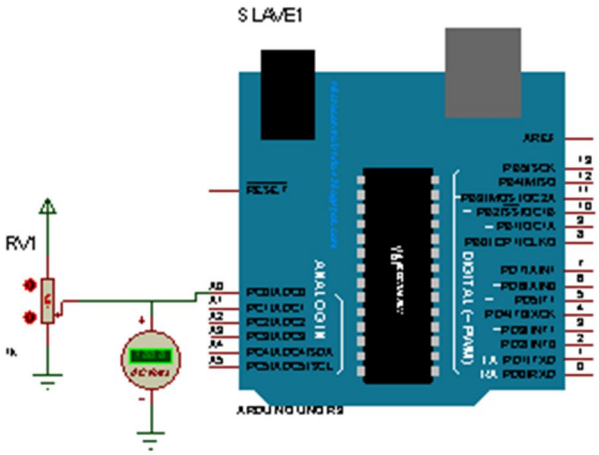


Figura 4.2: Esclavo 1 conexión para medir voltaje

Se realizo la comunicación con RS-485 empleando el protocolo MODBUS para mantener una comunicación entre las demás tarjetas Arduino, esta nos permite tener la comunicación en más de un dispositivo que nos permitirá mantener comunicado a una tarjeta Arduino que estará funcionando como maestro En la figura 4.3 se muestra la conexión realizada.

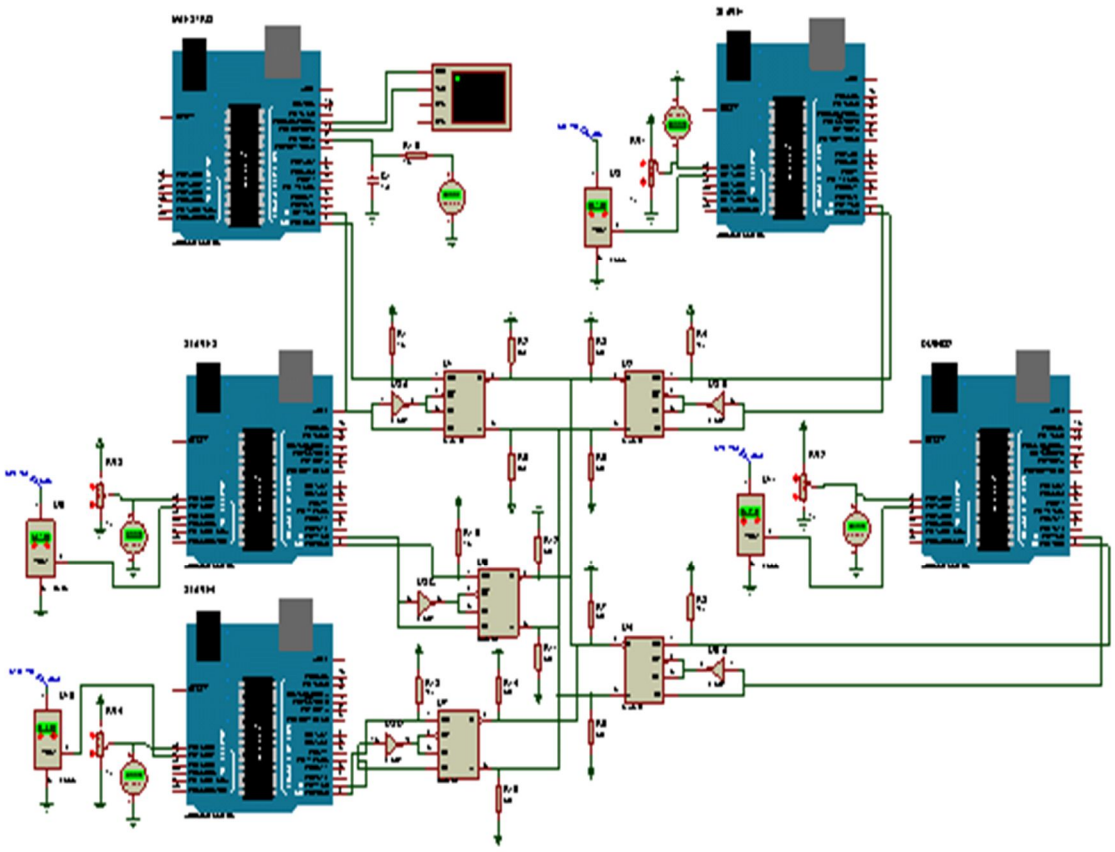


Figura 4.3: Comunicación con RS-485 entre Arduino maestro- esclavo

4.2 Pruebas del sensor de voltaje y DTMF

para el funcionamiento del sensor de voltaje se desarrolló un código de programación en el software Arduino IDE. El codigo fue cargado a la placa Arduino ATmega 2250, que se utiliza como esclavo figura…

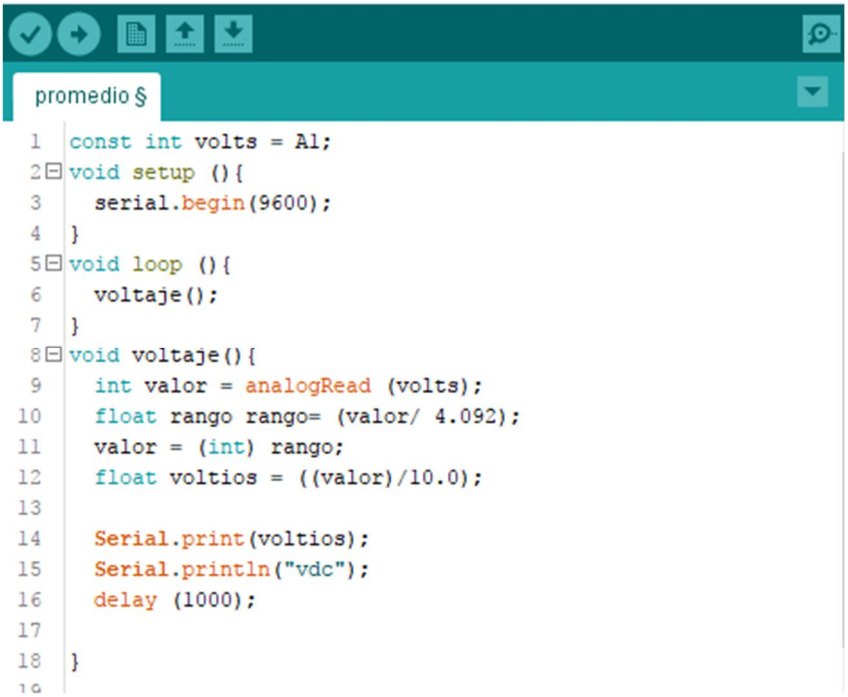


Figura 4.4: programa para el sensor de voltaje

Para la prueba del DTMF se realizó, el siguiente circuito que se muestra en la Figura 4.4 se conectó el Arduino con el circuito integrado DTMF el cual es una configuración para Arduino y circuito decodificador de tonos CM8870 el cual los convierte a un número binario para que pueda ser enviados por medio de la radio.

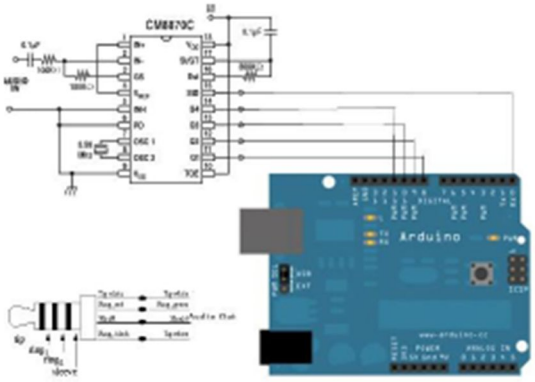


Figura 4.5: Circuito TDMF Arduino

Para comprobar si este circuito funcionaba de manera adecuada como se deseaba se realizó una prueba armando el circuito atentes de hacer la placa para ello se utilizó un protoboard y con ayuda de un radio portátil y un cable auxiliar para enviar los tonos del radio al circuito como se observa en la Figura 4.6.

Figura 4.6: prueba de la radio y el TDMF para generar tonos.

4.2 Pruebas Físicas

Después de realizar la simulación y tener los resultados requeridos se realizó una serie de pruebas con los Arduinos. en la figura 4.7 se puede observar la conexión de entre los Arduinos, el cual está conectado como maestro- esclavo, en donde la función es hacer que el maestro debe iniciar una solicitud y luego esperar una respuesta por parte del esclavo. [2]:

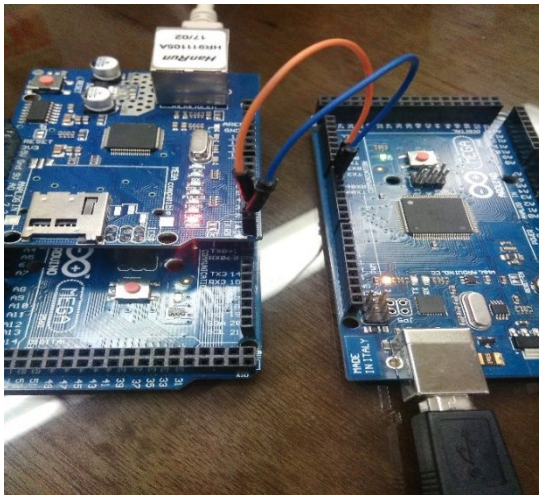


Figura 4.7: conexión entre maestro esclavo

En la figura 4.8 se puede observar que en el monitor serie del software Arduino (IDE), se aprecia que el maestro está esperando respuesta del esclavo, en donde si la conexión no se puede dar, ésta mandará un mensaje “esperando respuesta”.

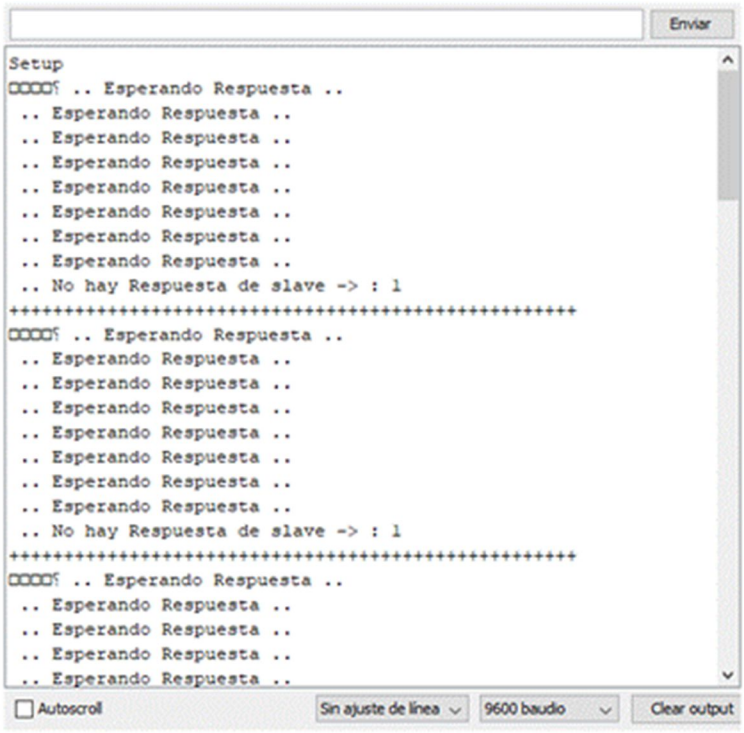


Figura 4.8: Monitor serie del software Arduino (IDE).

Después que el maestro solicita el estado de algunos de los esclavos este mandara un mensaje con los datos adquiridos al Arduino que funciona como esclavo, estos datos se enviasen al servidor **Zabbix.**

Cuando el maestro solicita información los esclavos mandaran un mensaje como se muestra en la figura 4.9, donde se observa los datos adquiridos del sensor de voltaje y de temperatura estos datos los recibirá el Arduino que se utilizara como esclavo para enviarlos al Arduino que será nuestro Maestro, posteriormente se enviaran al sistema de monitoreo **Zabbix**, el que se encargara de procesar la información obtenida.

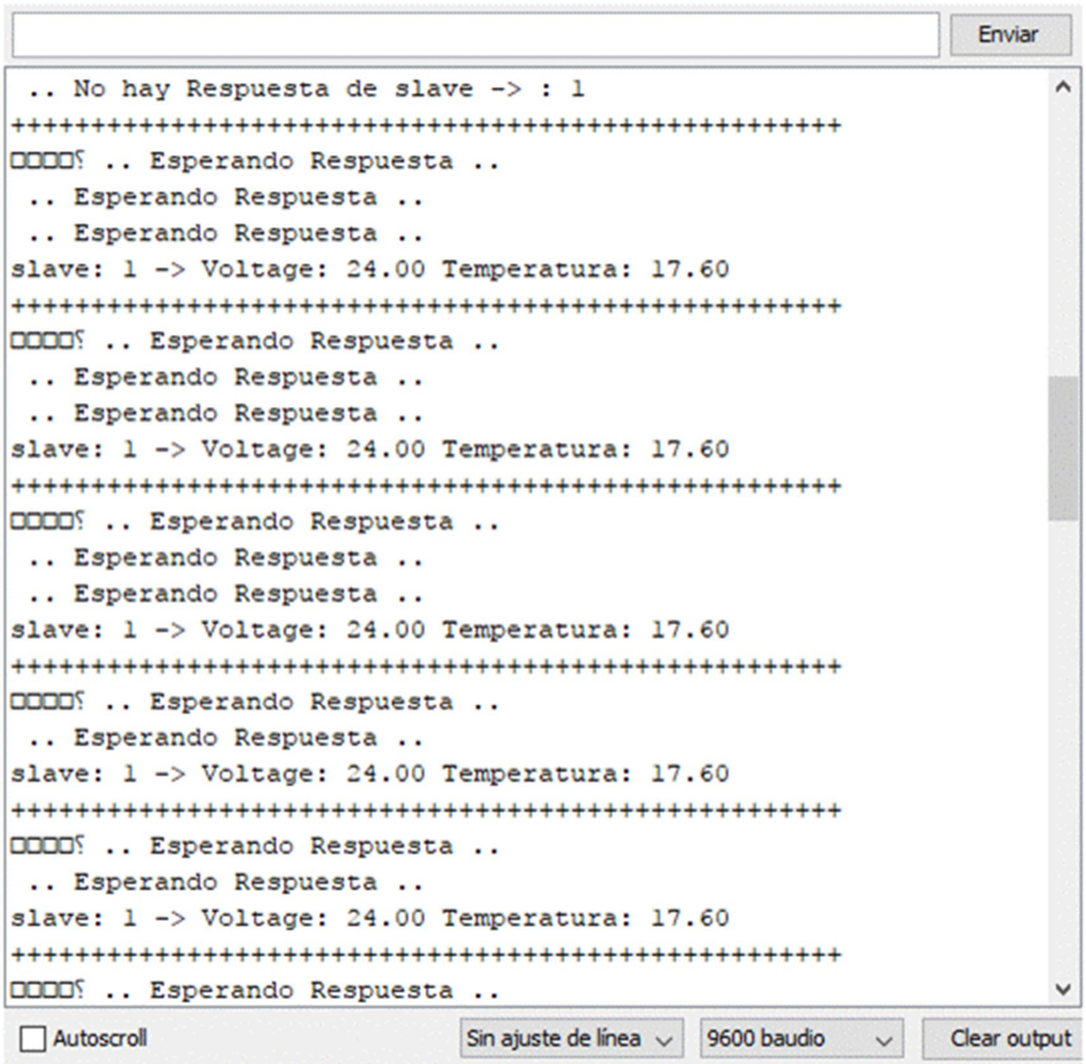


Figura 4.9: Arduino (IDE) respuesta del esclavo.

Cuando uno de los esclavos no responde a la solicitud del maestro en el monitor serie del software Arduino (IDE) nos enviará un mensaje diciendo no hay respuesta ("no hay respuesta del slave 2"), como se muestra en la figura 4.10.

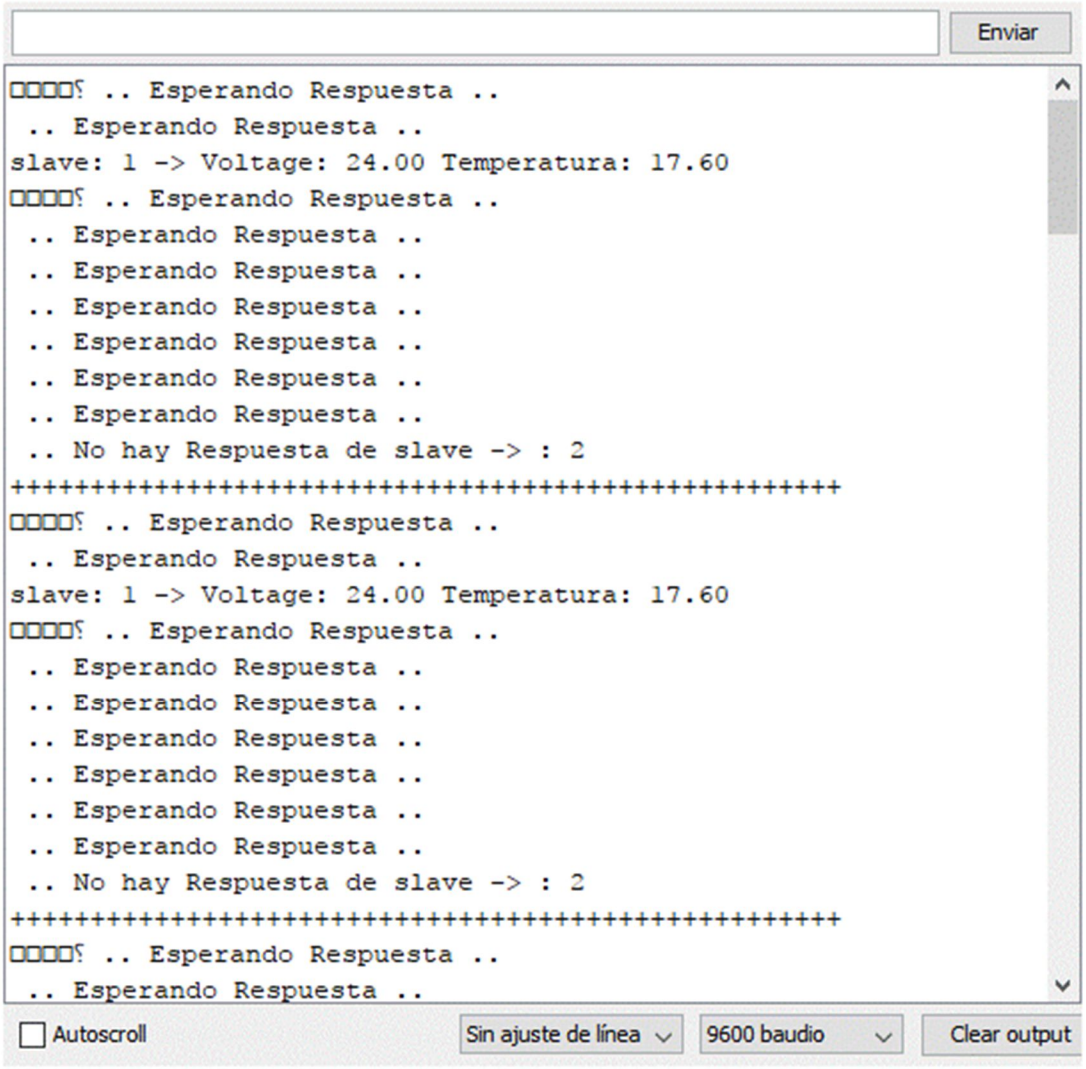


Figura 4.10: Arduino (IDE) no hay respuesta del esclavo

Para poder acceder al sistema de **Zabbix** , es necesario registrar al usuario que tendrá acceso a la plataforma de donde se mantendrá monitoreando los datos adquiridos de los Arduinos que se utilizan como Maestro-Esclavo. Con el sistema operativo de **Centos** se registra y se obtiene las IP figura 4.11 y figura 4.12, para el direccionamiento del ethernet shield que se adaptará al Arduino maestro que estará enviando los valores adquiridos de las casetas de comunicación a la red de datos con la que cuenta la empresa.

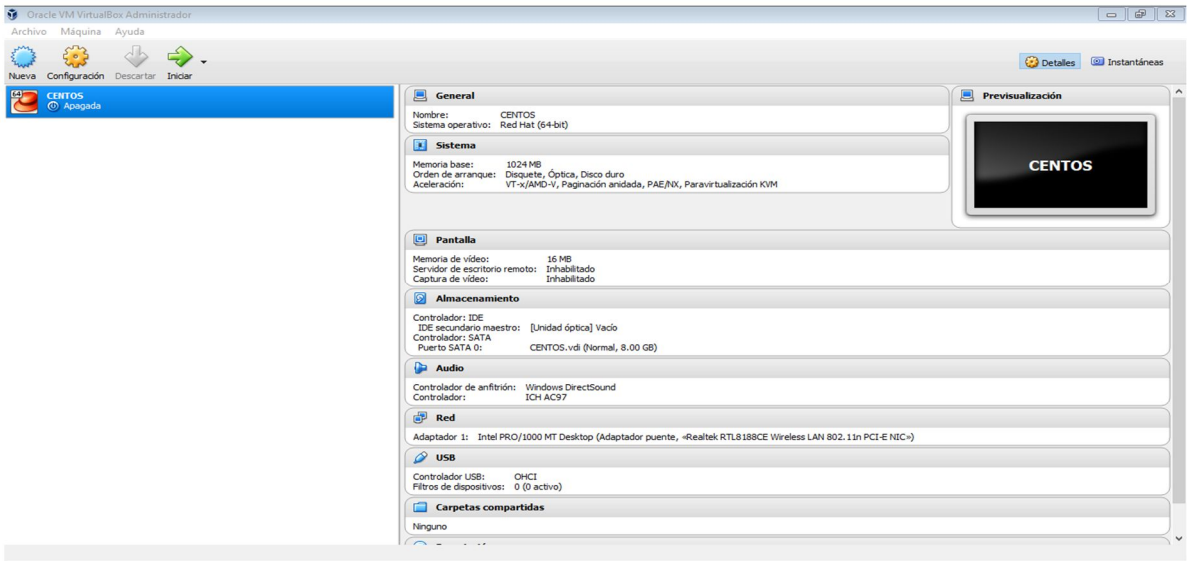


Figura 4.11: página principal Centos

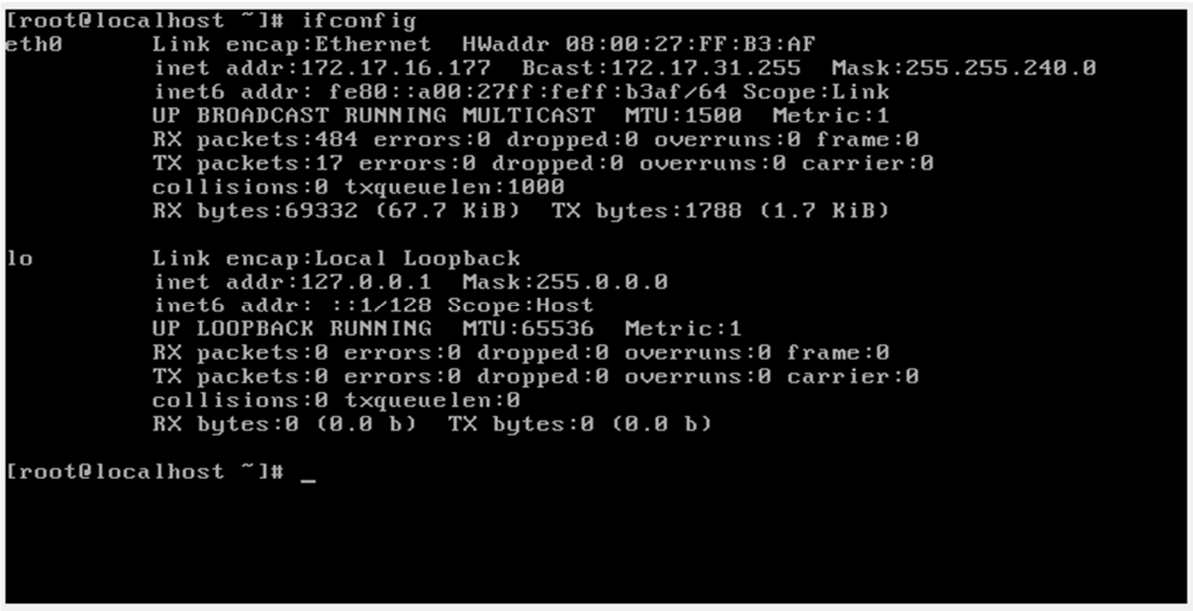


Figura 4.12: registro del usuario y direcciones IP

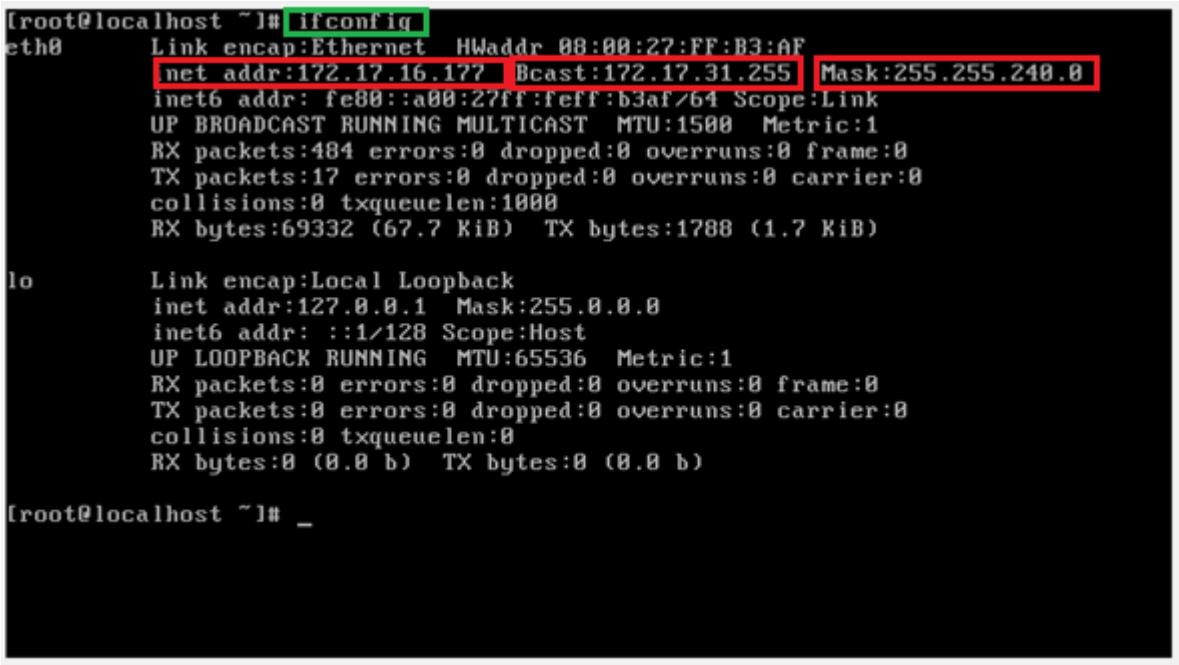


Figura 4.13: direcciones IP

En la siguiente figura 4.13 se puede observar las direcciones IP, que nos servirá para ingresar a la página web que se creó, al poner en el buscador la IP 172.17.16.177. figura 4.14 ingresara a la página creada por **Centos** con el nombre de “ESTA PAG RENE” figura 4.15 y así acceder a la página principal de **Zabbix.**

Figura 4.14: Pagina web

Los datos que se adquieran serán procesados en la plataforma Zabbix en donde se en encargará de analizar la información que se obtengan, para ello se tendrá que configurar la direcciones IP del Ethernet Shield, de la PC usando el navegador Chrome, como se muestra en la figura 4.12, para entrar necesita el nombre del usuario y la contraseña, la dirección IP de nuestra plataforma es 172.17.16.177.

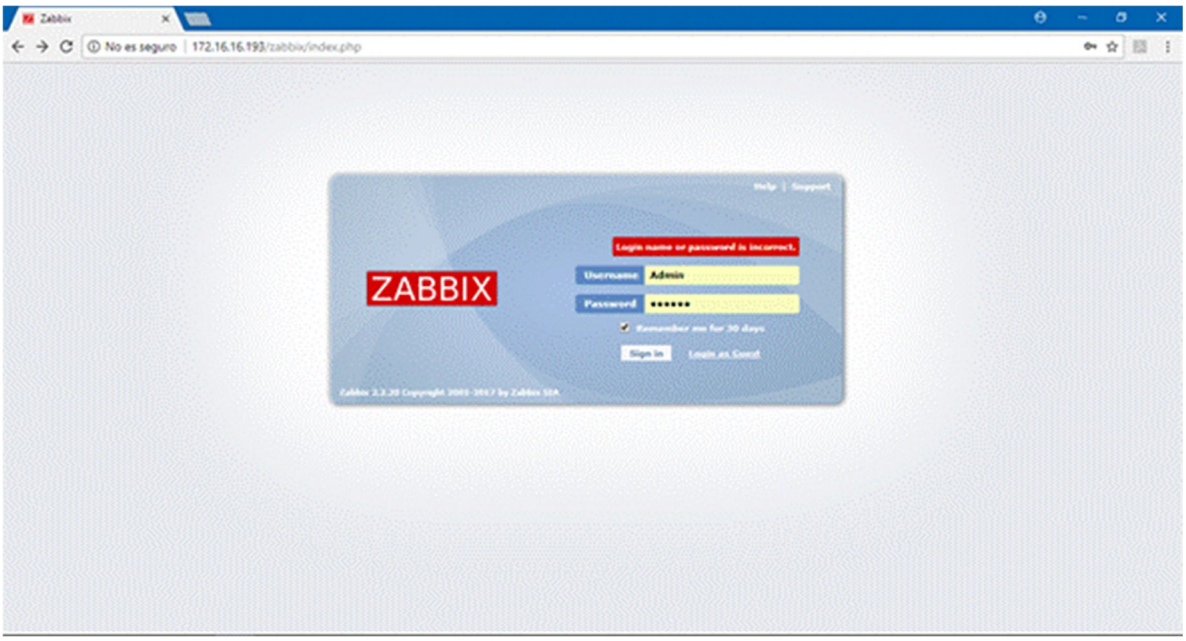


Figura 4.15: plataforma Zabbix

Con esta dirección IP se puede monitorear desde cualquier punto donde exista red de datos de la misma, teniendo un mejor control de la plataforma.

Después de tener acceso a la plataforma, estamos la pantalla principal de Zabbix como se muestra en la figura 4.16 en esta parte se muestra el menú en donde des puedes se hace las configuraciones necesarias para se pueda analizar las variables necesaria que se necesita controlar.

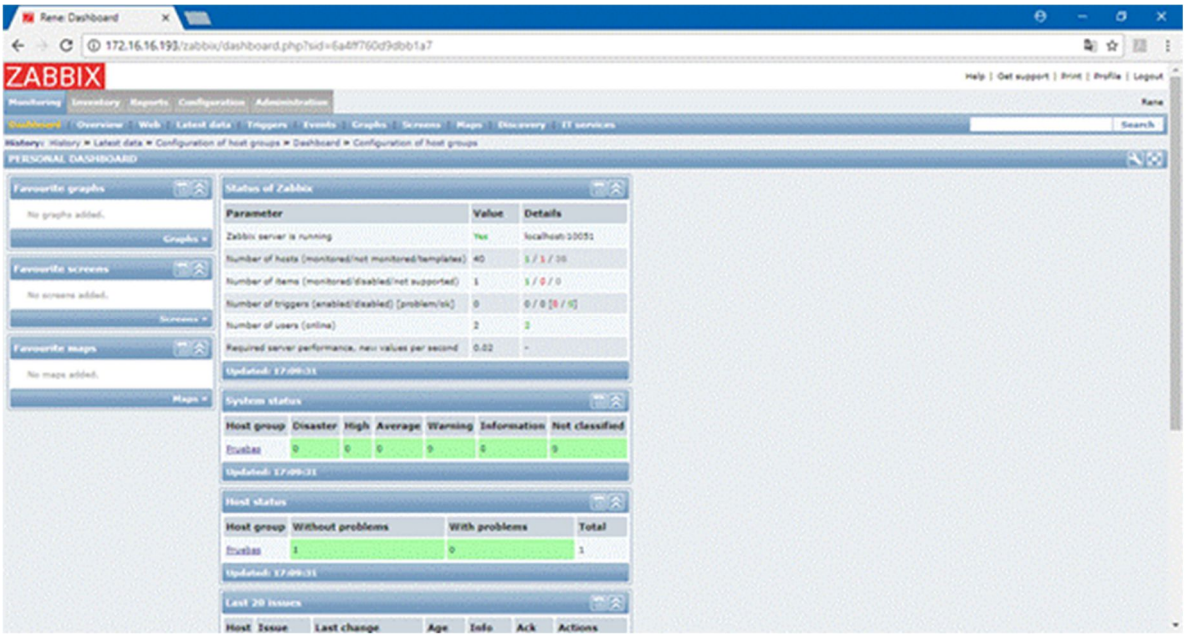


Figura 4.16: pantalla principal de la plataforma Zabbix

Para agregar los las variables que se necesita es necesaria configurarlo de la siguiente manera, en la figura 4.17 se puede observar, que para agregar una variable es necesario a la parte de configuración, en donde se tiene agregar la tarjeta Arduino que funciona como maestro, cual es que se encargará de estar enviando recopilar los dados de los demás dispositivos que funcionan como esclavo.

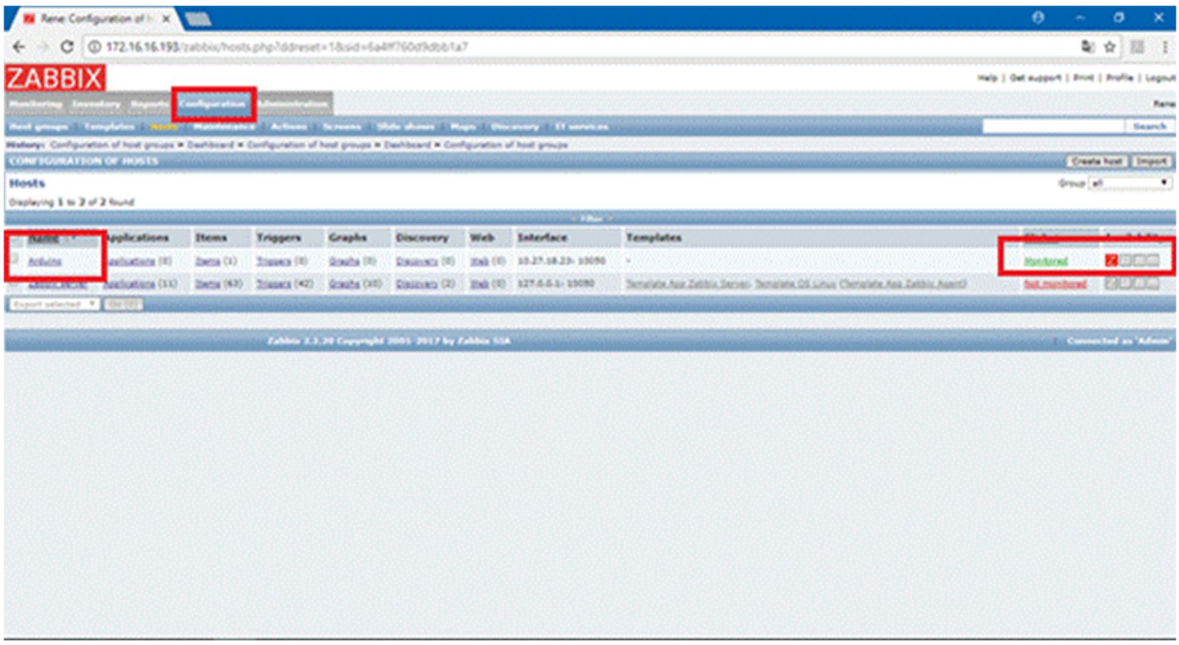


Figura 4.17: Configuración de la tarjeta Arduino a la plata forma Zabbix.

Para Arduino Ethernet Shield es necesario agregar la dirección IP 172.16.16.193, figura 4.18 y tipo de variable que se estará monitoreando para tenga los datos correcto correctos.

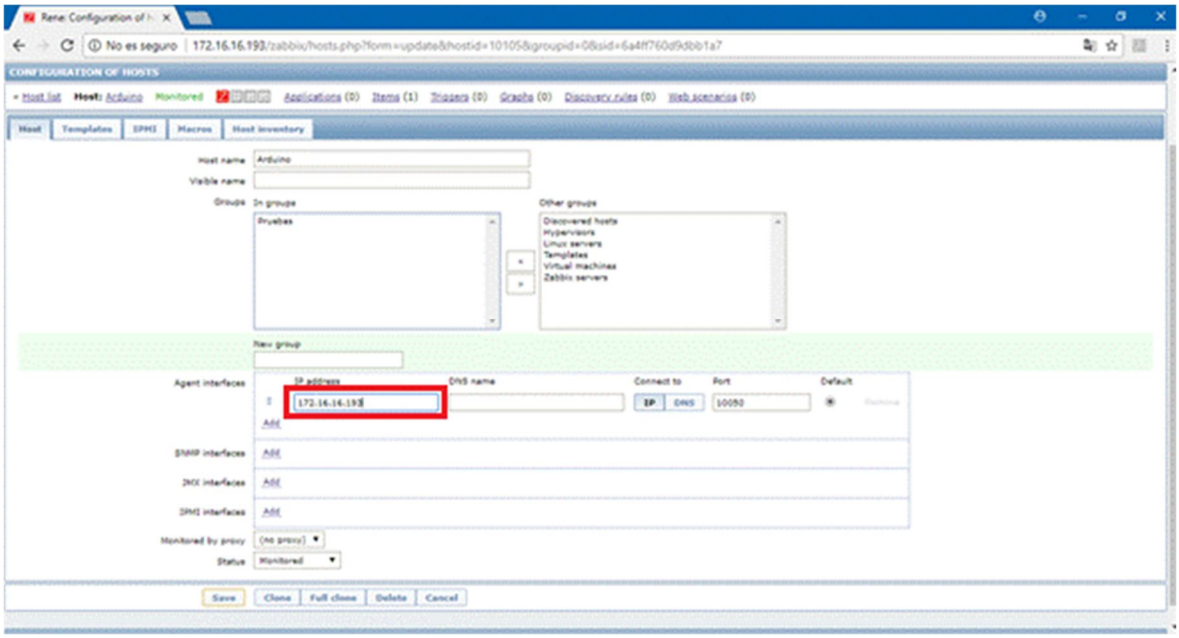


Figura 4.18: configuración Ethernet Shield al Zabbix.

Después de hacer configurado correctamente el maestro a la plataforma Zabbix los datos que se obtienen, son graficados para que se te pueda apreciar gráficamente en la figura se 4.19 se muestra, los datos obtenidos de la variable de temperatura.

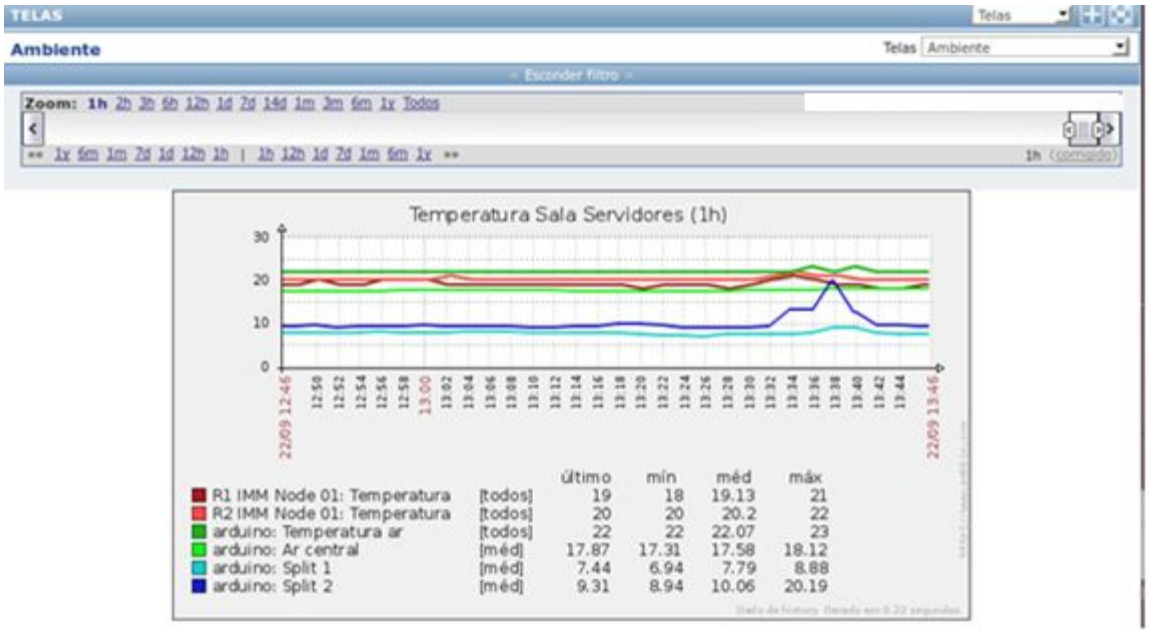


Figura 4.19: Datos registrado por la tarjeta Arduino maestro.

Diseño del circuito en Isis Proteus PCB

Para el diseño y construcción de las placas se utilizó el software Proteus **8** esta plataforma nos ayuda a simular los circuitos y comprobar si funcionan de manera correcta antes de realizar pruebas físicas y con la ayuda de este simular se puede realizar las pistas donde se conducirá la señal de las variables a monitorear, así como la entrada de voltaje. Con se muestra en la figura 4.20

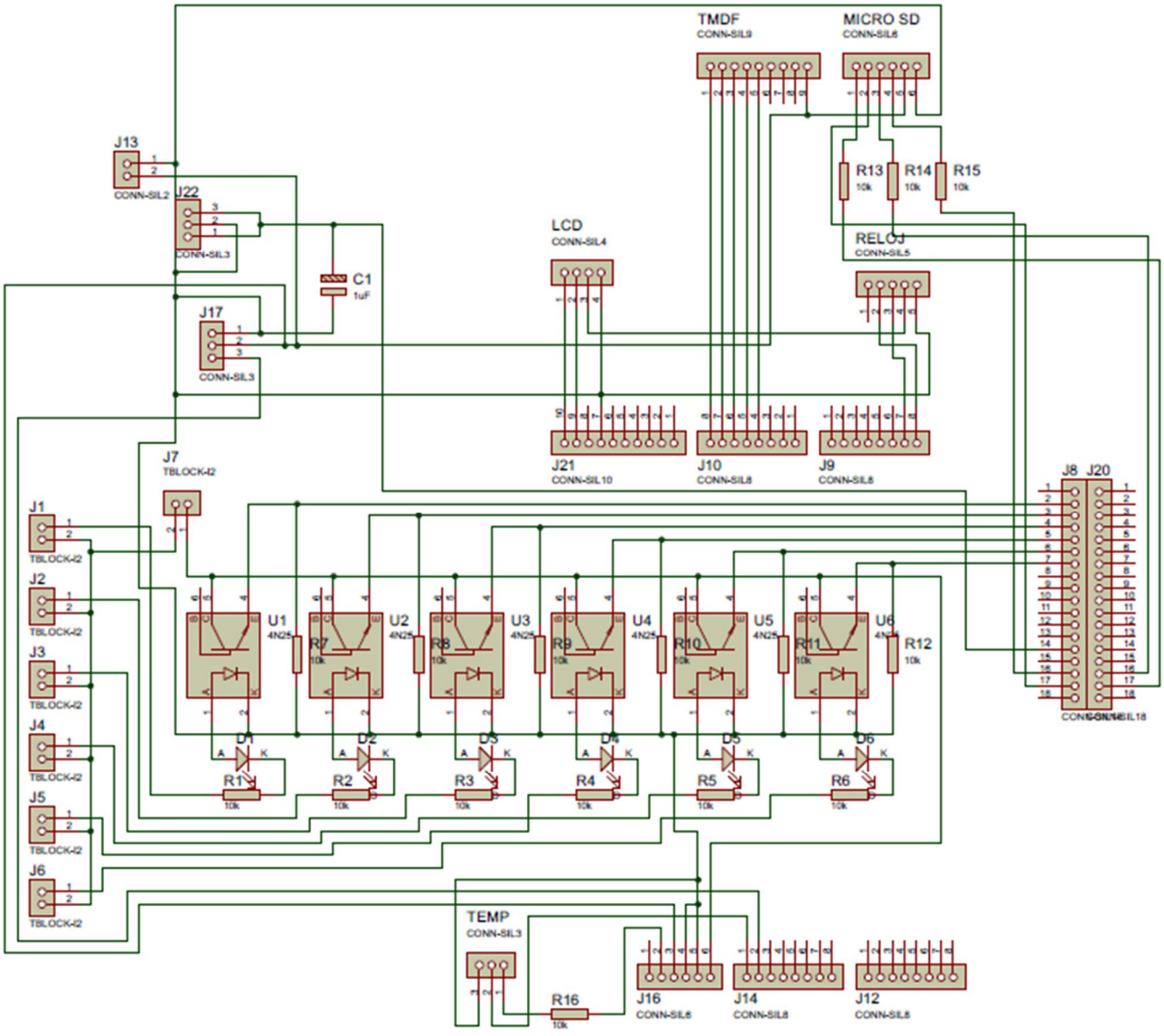


Figura 4.20: Diseño de circuito en ISIS Proteus.

Se usaron los Optocopladores 4N25 para diseñar una etapa de potencia el cual sirve para aislar el alto voltaje de los valores adquiridos de las corrientes alternas de las casetas donde se encuentran los repetidores , como la de 220 AC y 127 AC para que no exista algún daño hacia la placa Arduino .

Diseño en 3D en proteus

En esta figuras 4.21 se aprecia la vista superior de la placa del circuito que se diseño y se puede visualizar los componentes que lo integran .

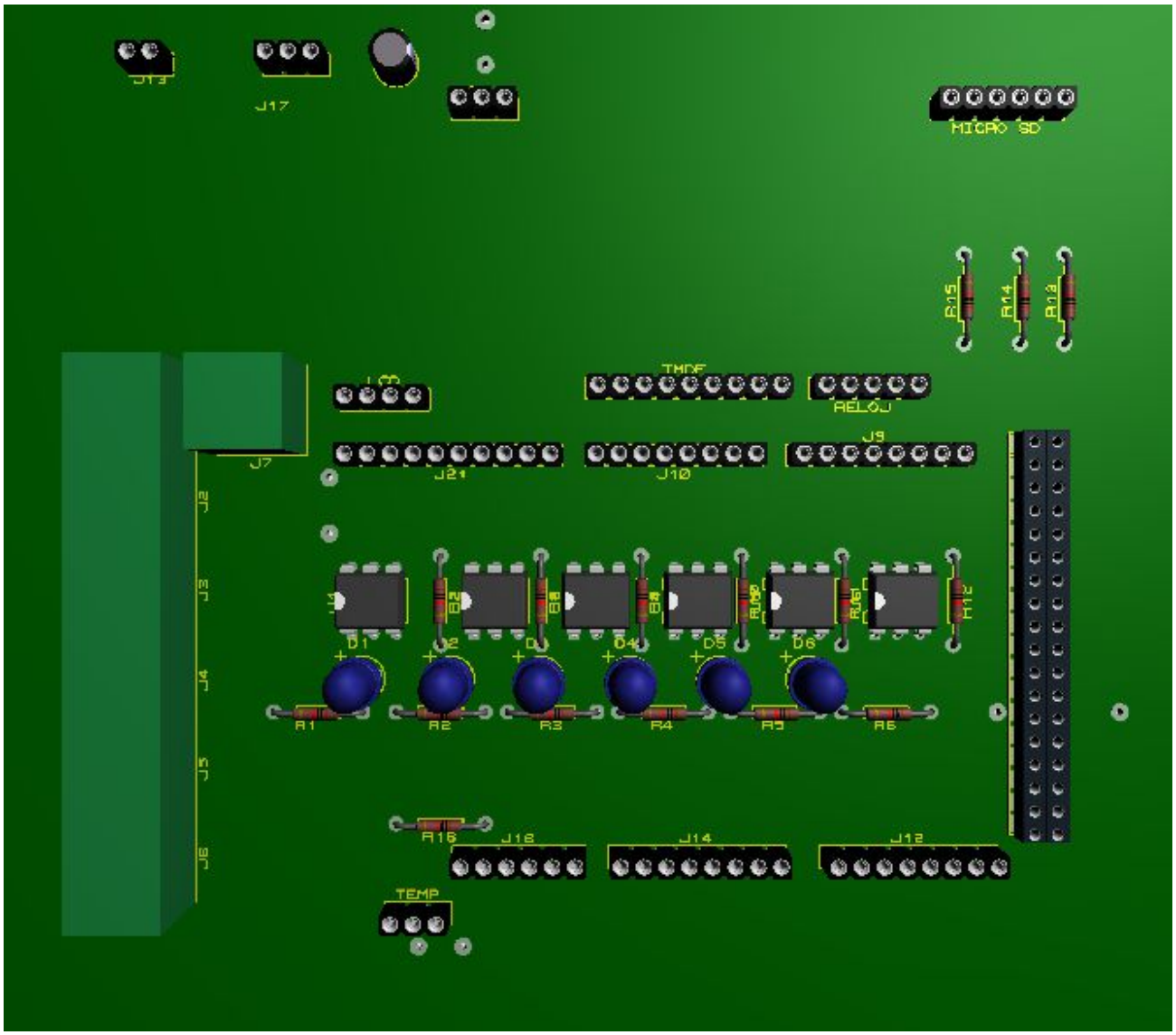


Figura 4.21: Visualizacion del circuito en 3D

Como se puede apreciar en la figura 4.22 se observa el ruteo de las pistas junto con los componentes que integran el circuito, se diseñó manualmente para que no existiera muchos errores en la conexión de las pistas con los componentes.

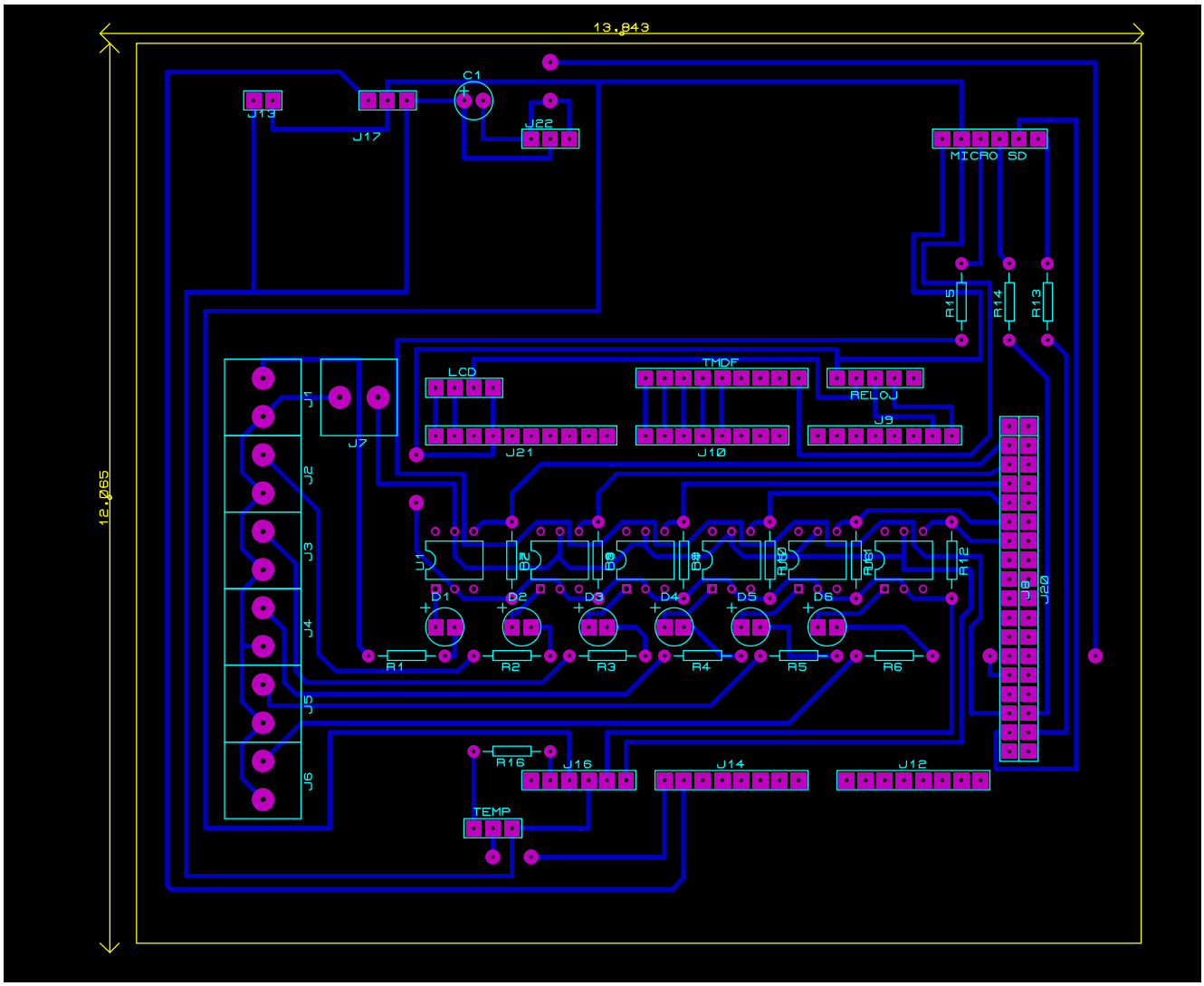


Figura 4.22: Diseño de las pistas y componentes que integran el circuito.

Después de realizar el diseño en Isis Proteus y de obtener las pistas que se necesitara para poder soldar los componentes necesarios se imprimió en una hoja llamada papel Couche el cual nos permitirá transferir las pistas a la palca fenólica y con el uso del cloruro férrico se lavara la placa eliminando el cobre que no se necesita así dejando solo las pistas. Figura 4.23

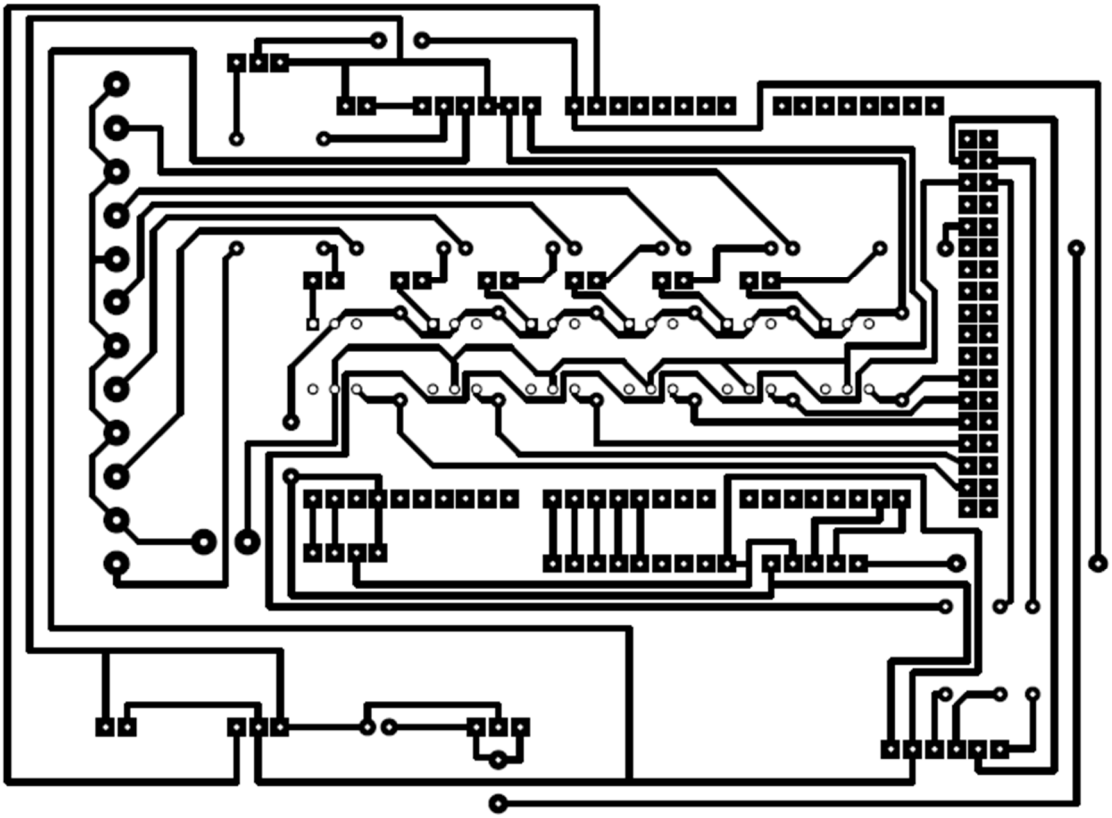


Figura 4.23: Pistas del circuito.

4.3 Construcción del circuito

Para la creación de la placa se necesitó los siguientes materiales:
1. Arco con segueta
2. Recipiente de platico
3. Plancha
4. Placa fenólica
5. Cautín para soldar con base
6. Cloruro férrico
7. Solvente para limpiar la placa (Tiner)
8. Lija 1200
9. Mesa de trabajo

Para la construcción se imprimió el circuito en una hoja de papel Couche el cual es un papel que nos ayudara transferir las pistas a la placa fenólica que se muestra en la figura 4.24 para ello es necesario lijarla primero para poder transferir el circuito que se imprimió, después del lijado se limpia la placa para eliminar la grasa que impida que el circuito quede plasmado en la placa fenólica.

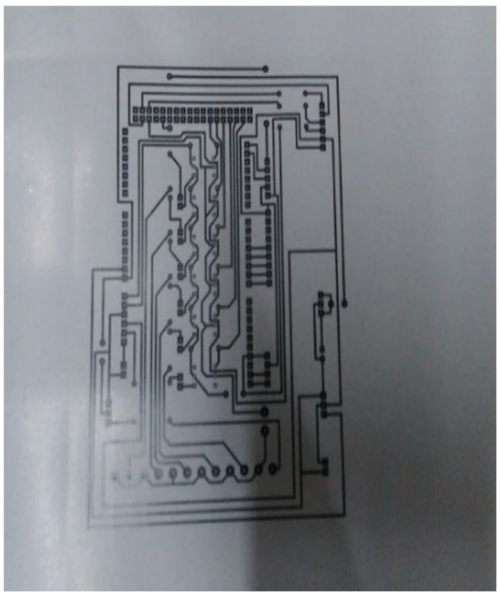


Figura 4.23: Impresión del circuito en papel Couche.

Figura 4.24: Placa fenólica.

El siguiente paso que se realizo fue que se colocó el papel Couche figura 4.24 encima de la placa fenólica en la parte donde ese encuentra cubierta de cobre y se aseguró con cinta de aislar de manera que no se moviera al momento del planchado como se observa en la figura 4.25 durante 20 minutos hasta que la tinta quedara impregnada en la placa.

Figura 4.25: Placa cubierta con el papel Couche.

Después de que se usó el método del planchado en la placa para transferir la impresión. En un recipiente lleno de agua lo sumergimos para eliminar el papel sobrante figura 4.26, para dejar solo la tinta con la forma del circuito que se utilizara, luego de este paso en otro recipiente agregamos cloruro férrico este acido nos ayudara a eliminar el exceso de cobre dejando solamente las pistas del circuito.
Para eliminar el cobre en necesario cubrir la placa con el cloruro férrico figura () se agito el recipiente hasta que el cobre que no está cubierto por la tinta se disuelva por completo

Figura 4.26: Planchado del circuito la placa fenólica.

Después de que se usó el método del planchado en la placa para transferir la impresión. En un recipiente lleno de agua lo sumergimos para eliminar el papel sobrante figura 4.27, para dejar solo la tinta con la forma del circuito que se utilizara, luego de este paso en otro recipiente agregamos cloruro férrico este acido nos ayudara a eliminar el exceso de cobre dejando solamente las pistas del circuito.
Para eliminar el cobre en necesario cubrir la placa con el cloruro férrico figura 4.28 se agito el recipiente hasta que el cobre que no está cubierto por la tinta se disuelva por completo.

Figura 4.27: Eliminación del papel Couche de la placa.

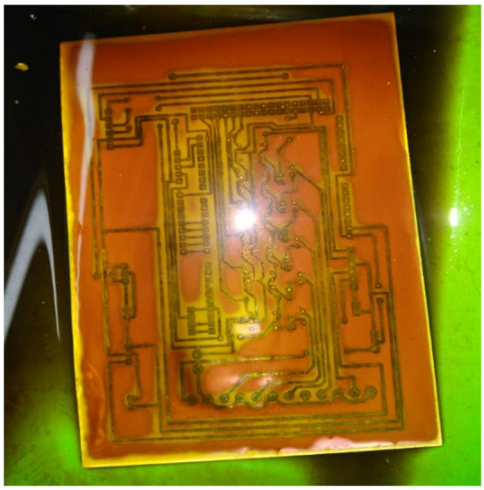


Figura 4.28: Lavado de la placa fenólica.

Cuando el cobre se ha eliminado por completo es necesario volver a lavar la placa con agua limpia figura 4.29, ya que se encuentra cubierta del ácido el cual puede ser de peligro para nuestra salud. luego de a ver lavado la placa se puede observar que el cobre que no está cubierto con la tinta se ha quitado con la ayuda del cloruro férrico figura 4.30.

Figura 4.29: Lavado para quitar el cloruro férrico.

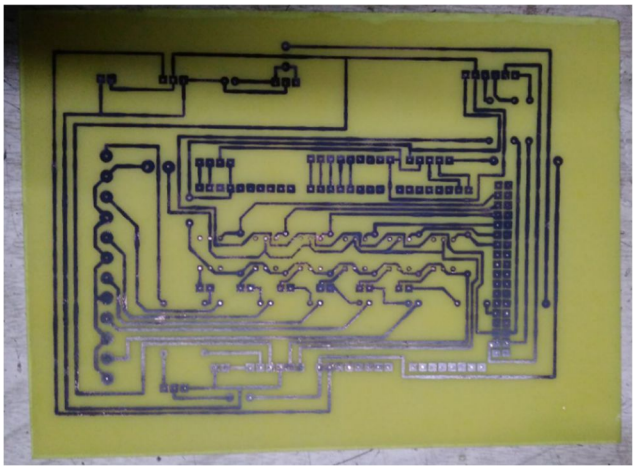


Figura 4.30: Circuito impreso y lavado.

Ya que se realizó todo el proceso para imprimir el circuito en la placa fenólica, quitar el cobre que no se utilizara y dejando solamente el circuito se observa en la figura 4.31 que lo que resalta es la tinta de la impresión que nos permitió plasmar el circuito, es necesario limpiar la tinta con un solvente en nuestro caso utilizamos tiner. Como proceso final se limpia la tinta con el solvente dejando solamente las pistas de cobre del circuito figura 4.32.

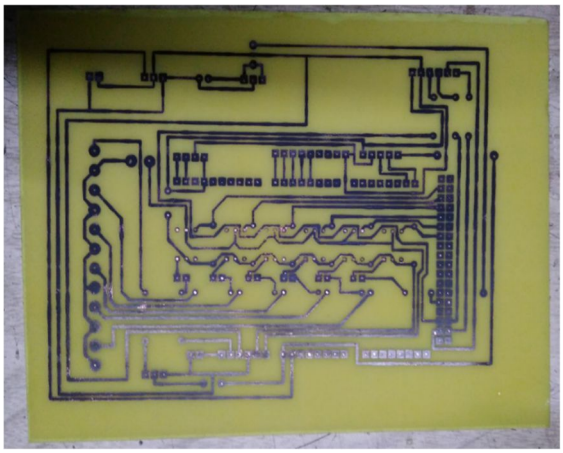


Figura 4.31: Tinta de la impresión que forma el circuito.

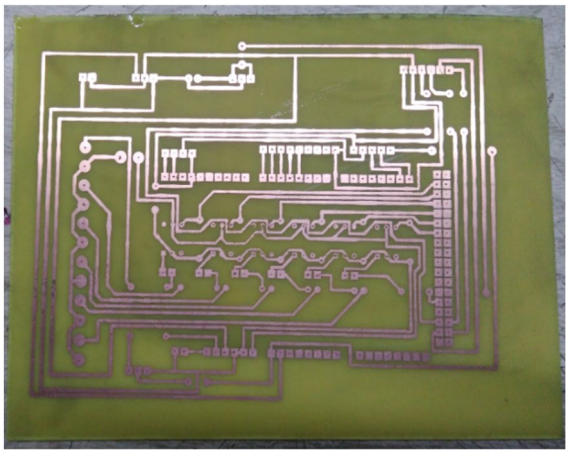


Figura 4.32: Pistas del circuito.

Construcción de código de Arduino para el funcionamiento del proyecto

En la elaboración del código se llevó a cabo mediante funciones individuales para realizar pruebas y verificar el funcionamiento de manera correcta de los componentes que se usaran para el monitoreo de los dispositivos que se encuentran dentro de la caseta donde están ubicados los repetidores.

Tanto como el código que se utilizó para el funcionamiento de los sensores de temperatura y voltaje de igual forma se utilizó una librería de Arduino figura 4.33 para controlar el módulo de Ethernet Shield que permitirá que el Arduino maestro se conecte a la red de datos de la empresa.

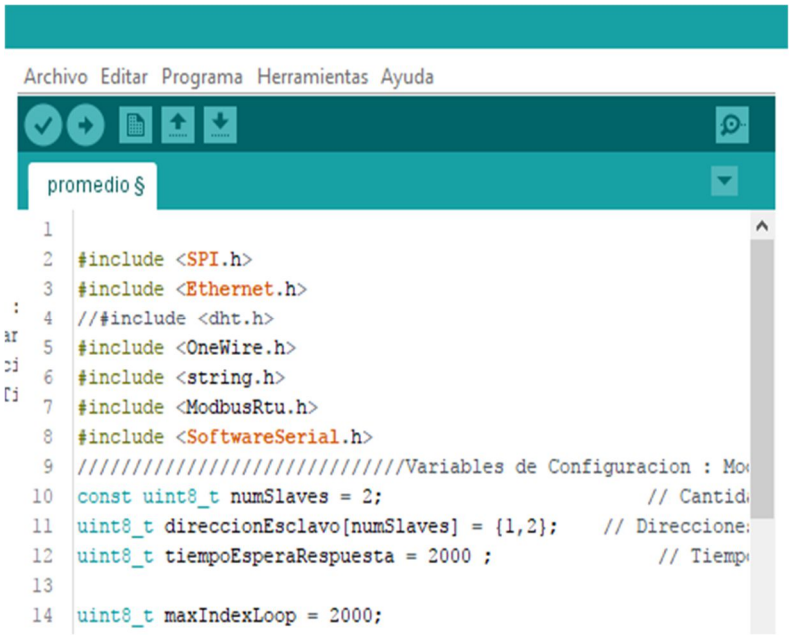


Figura 4.33: código de Arduino utilizado para todo el proceso de monitoreo.

Observaciones.

Para la empresa Zona de Transmisión Tuxtla es de suma importancia mantener monitoreado y sensado los equipos de comunicación instalados en torres de líneas de transmisión, es por ello que se diseñó el presente proyecto para mantener el monitoreo constante en equipos repetidores una de las observaciones más importantes es que estas torres se encuentran retiradas de la Zona de Transmisión Tuxtla, impidiendo así el traslado constante del personal.

Por motivos presupuestales la empresa **EPS** no pudo proporcionar los materiales necesarios para el desarrollo del presente proyecto para abastecernos de los materiales necesarios.

Por tal motivo no se pudo implementar de forma física pruebas de operación, ajustes y anotaciones de los resultados obtenidos durante el desarrollo del proyecto solamente se pudo realizar pruebas simuladas y pruebas físicas con los materiales que se contaba como la tarjeta Arduino que se utilizó como Maestro-Esclavo.

El cloruro férrico, es un coagulante inorgánico muy eficaz en la eliminación de sólidos suspendidos, rastros de metales. Por tal motivo el manejo de esta sustancia fue con mucha responsabilidad ya que es un agente muy contaminante para el medio ambiente, al término de su uso se dé deposito en un recipiente y se selló bien evitando el derrame del cloruro férrico para después ser llevado aun deposito donde se almacena sustancias toxicas.

Conclusión

Para tener un control eficiente en diferentes sitios, es importante tener monitoreado los diferentes tipos de variables en que se manejan en dichos sitios, importante tener una información en tiempo real para que se pueda estar visualizando en una plataforma que nos mantenga informado de los eventos que se puedan suceder, como es el incremento de temperatura, alto o bajo voltaje, entre otros. Hacer un poleo cada cierto tiempo para tener una información actualizada es importante para tener un mejor control y prevenir daños en los sitios que se estén monitoreando.

Referencias

[1] M. J. Fabra, «Estudio para la migración del sistema de radiocomunicaciones VHF analógicos de la empresa eléctrica quito,» Octubre 2017.

[2] W. Tomasi, Sistemas de Comunicaciones Electrónicas, México: PEARSON EDUCACIÓN, 2003.

[3] A. Gerson , L. Camacho, D. Chávez, C. Córdova y D. Espinoza, REDES INALAMBRICAS PARA ZONAS RURALES, Lima: Pontificia Universidad Católica del Perú, Enero 2008.

[4] I. national, «Información Detallada sobre el protocolo Modbus,» 16 de octubre, 2014.

[5] «ARDUINO USA ONLINE,» 12 diciembre 2017. [En línea]. Available: http://forum.arduino.cc/index.php?topic=70235.0.. [Último acceso: 12 12 2017].

[6] «web Robotica,» 2017. [En línea]. Available: http://www.web-robotica.com/arduino/como-funciona-el-modulo-arduino-ethernet-shield. [Último acceso: 7 1 2017].

[7] «Datasheet,» *EXAR,* February 2008.

[8] EXAR Corporation, «XR-2211A FSK Demodulator/Tone Decoder,» *Datasheet,* June 1997.

[9] «Cronos. Electronica,» 2017. [En línea]. Available: https://www.cronoselectronica.com/sensores/47-fz0430.html. [Último acceso: 7 1 2018].

[10] «Tuelectrinica.es,» 2017. [En línea]. Available: https://tuelectronica.es/modulo-rtc-ds1307-arduino/. [Último acceso: 7 1 2018].

[11] M. P. a. D. a. t. o. M. I. Motora, «Operating instructions for the Motorola Radius GM300 8-Channel, Conventional FM Radio,» 1997.

[12] «Zabbix- IEEE SDN CatalogueConfluence,» 24 Octubre 2016. [En línea]. Available: https://wiki.sdn.ieee.org/display/sdn/Zabbix.

[13] C. Valencia y G. A. A.H. Echeverry , «Prototipo de un Sistema de Telemetría y Control para Seguridad en vehículos,» 2012.

Anexos

Código de Arduino del que funciona como el maestro para introducir al Zabbix.

```
#include <SPI.h>
#include <Ethernet.h>
//#include <dht.h>
#include <OneWire.h>
#include <string.h>
#include <ModbusRtu.h>
#include <SoftwareSerial.h>
//////////////////////////////Variables de Configuracion : Modificar de acuerdo a lo requerido
//////////////
const uint8_t numSlaves = 2;                    // Cantidad de esclavos a monitorear.
uint8_t direccionEsclavo[numSlaves] = {1,2};    // Direcciones de los esclavos
uint8_t tiempoEsperaRespuesta = 2000 ;              // Tiempo en ms para esperar la
respuesta de un esclavo

uint8_t maxIndexLoop = 2000;



////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////// Variables del algoritmo :   No modificar ////////////////////////////////////////
uint8_t esclavoFallado[numSlaves];
uint16_t au16data[16];               // data array for modbus network sharing
uint8_t u8state;                    // Estado actual de la comunicacion 0 = menor a 1
segundo, 1 = configura y realiza una peticion , 2 = Recibe los datos de una peticion
e imprime
unsigned long u32wait;               // millis() + x segundo entre preguntas
```

```
uint8_t indexLoop = 0;
float p;
int adc;
int temperatura1;
int corrient;
int indiceIteracion = 0;
Modbus master(0, 0, 0);              // this is master and RS-232 or USB-FTDI
//SoftwareSerial portOne(10, 11);       /// pines software serial  -  software serial #1: TX = digital pin 10, RX = digital pin 11
modbus_t telegram;                  /**    This is an structe which contains a query to an slave device */
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//----------------------------- Network settings ------------------------------
byte mac[] = { 0x90, 0xA2, 0xDA, 0x00, 0xE3, 0x1B };
//IPAddress ip(10, 1, 2, 235);
IPAddress ip(10,27,18,23);
IPAddress gateway(10,27,18,62);
IPAddress subnet(255, 255, 255, 0);

//------------ Pins 10, 11, 12 e 13 are used by ethernet shield! ---------------
#define MAX_CMD_LENGTH 25
#define LED_PIN 3              // LED pin with 1k resistor
#define DHT11_PIN 4            // DHT11 pin with a 10k resistor
#define ONE_WIRE_PIN 5         // One wire pin with a 4.7k resistor
#define PIR_PIN 6              // PIR pin
#define SOIL_PIN A0            // Soil humidity sensor pin

EthernetServer server(10050);
EthernetClient client;
OneWire ds(ONE_WIRE_PIN);
//dht DHT;
```

```
boolean connected = false;
byte i;
byte present = 0;
byte type_s;
byte data[12];
byte addr[8];
double temp = 0;              // Temperature
double umid = 0;              // Humidity
float celsius;
float oneWire17 = 0;
float oneWireB6 = 0;
float oneWireD3 = 0;
String cmd;                   //FOR ZABBIX COMMAND
String serialNum;
int counter = 1;              // For testing
int chk;
int presence = 0;
int lastPresence = 0;
int soil = 0;                 // Soil humidity
int limite = 1;               // Command size. Using 1 for better performance.
int waitTime = 15000;         // Default waiting time before reading sensor again.
int pirWaitTime = 200000;     // Default waiting time for PIR.
unsigned long dhtLastCheck = 0;
unsigned long pirLastCheck = 0;
unsigned long oneWireLastCheck = 0;
// Read all DS18b20 and saves the result on variables every 15 seconds.--- Lee todo
DS18b20 y guarda el resultado en una variables cada 15 segundos
void readOneWire() {
  if (millis() - oneWireLastCheck > waitTime) {
    if ( !ds.search(addr)) {
      //Serial.println("No more addresses.");
```

```
  ds.reset_search();
  //delay(250);
  return;
}
if (OneWire::crc8(addr, 7) != addr[7]) {
  Serial.println("CRC is not valid!");
  return;
}
switch (addr[0]) {
  case 0x10:
    type_s = 1;
    break;
  case 0x28:
    type_s = 0;
    break;
  case 0x22:
    type_s = 0;
    break;
  default:
    Serial.println("Device is not a DS18x20 family device.");
    return;
}
ds.reset();
ds.select(addr);
ds.write(0x44, 1);      // start conversion, with parasite power on at the end
//delay(1000);          // maybe 750ms is enough, maybe not
// we might do a ds.depower() here, but the reset will take care of it.
present = ds.reset();
ds.select(addr);
ds.write(0xBE);         // Read Scratchpad
for ( i = 0; i < 9; i++) {           // we need 9 bytes
```

```
      data[i] = ds.read();
    }
    int16_t raw = (data[1] << 8) | data[0];
    if (type_s) {
      raw = raw << 3; // 9 bit resolution default
      if (data[7] == 0x10) {
        raw = (raw & 0xFFF0) + 12 - data[6];
      }
    } else {
      byte cfg = (data[4] & 0x60);

      if (cfg == 0x00) raw = raw & ~7;      // 9 bit resolution, 93.75 ms
      else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
      else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
    }
    celsius = (float)raw / 16.0;
    //Serial.print(celsius);
    //Serial.println(" Celsius");
    serialNum = String(addr[7], HEX);
    if (serialNum == "17") oneWire17 = celsius;
    else if (serialNum == "b6") oneWireB6 = celsius;
    else if (serialNum == "d3") oneWireD3 = celsius;
    // If Fahrenheit needed, (Fahrenheit = Celsius * 1.8) + 32;

    oneWireLastCheck = millis();
  }
}

// Read DHT11 every 15 seconds and save values on variables
void readDHT11() {
 if (millis() - dhtLastCheck > waitTime) {
```

```
    chk = DHT.read11(DHT11_PIN);
    switch (chk) {
      case DHTLIB_OK:
        break;
      case DHTLIB_ERROR_CHECKSUM:
        Serial.print("Checksum error,\t");
        break;
      case DHTLIB_ERROR_TIMEOUT:
        Serial.print("Time out error,\t");
        break;
      default:
        Serial.print("Unknown error,\t");
        break;
    }

    //temp = DHT.temperature;
    temp=millis()/1000;
    umid= 1;

    //umid = DHT.humidity;
    dhtLastCheck = millis();

  }
}

// Read command received.
void readTelnetCommand(char c) {
  if (cmd.length() == MAX_CMD_LENGTH) {
    cmd = "";
  }
```

```
  cmd += c;
  if (c == '\n' || cmd.length() == limite) {
    parseCommand();
  }
  else {
  }
}
// Read soil humidity sensor.
void readSoil() {
  soil = digitalRead(SOIL_PIN);
}

// Read PIR and if positive, keep the value for pirWaitTime seconds.
void readPresence() {
  if (digitalRead(PIR_PIN)) {
    pirLastCheck = millis();
    lastPresence = 1;
  }
  if (digitalRead(PIR_PIN) && (lastPresence)) {
    presence = 1;
  }
  else {
    if (millis() - pirLastCheck > pirWaitTime) {
      presence = 0;
      lastPresence = 0;
    }
  }
}
//Commands received by agent on port 10050 parsing
void parseCommand()
{
```

```
if (cmd.equals("")) {  }
else {
  counter = counter + 1;
  Serial.print(" Tempo: ");
  Serial.print(millis() / 1000);
  Serial.print("\t");
  Serial.print("Cmd: ");
  Serial.print(cmd);
  Serial.print("\t\t");
  Serial.print("Resposta: ");
  // AGENT ping
  if (cmd.equals("p")) {
    server.println("1");
  } // Agent version
  else if (cmd.equals("l")) {
    //Serial.println("Version");
    server.println("Arduino Zabbix Agent 1.0");
    delay(100);
  } // Agent soil humidity
  else if (cmd.equals("q")) {
    readSoil();
    Serial.print(soil);
    server.println(soil);
    // NOT SUPPORTED
  } // Agent air temperature
  else if (cmd.equals("w")) {
    readDHT11();
    Serial.print(temp);
    server.println(temp);
    dhtLastCheck = millis() - waitTime;
  } // Agent air humidity
```

```
    else if (cmd.equals("e")) {
      readDHT11();
      server.println(umid);
      Serial.print(umid);
    } else if (cmd.equals("r")) {
      server.println(oneWire17);
      Serial.print(oneWire17);
    } else if (cmd.equals("f")) {
      server.println(oneWireB6);
      Serial.print(oneWireB6);
      oneWireLastCheck = oneWireLastCheck - (waitTime/3);
    } else if (cmd.equals("v")) {
      server.println(oneWireD3);
      Serial.print(oneWireD3);
      oneWireLastCheck = oneWireLastCheck - (waitTime/3);
    } else if (cmd.equals("t")) {
      server.println(presence);
      Serial.print(presence);
      oneWireLastCheck = oneWireLastCheck - (waitTime/3);
      pirLastCheck = millis() - pirWaitTime;
    } else { // Agent error
      //server.print("ZBXDZBX_NOTSUPPORTED");
      //server.print("Error");
    }
    cmd = "";
    Serial.println("");
    client.stop();
  }
}

// The loop that Arduino runs forever after the setup() function.
```

```
void loop() {
  if( indexLoop == 0){
    queryModbus();
  }
  indexLoop++;
  if(indexLoop > maxIndexLoop){
    indexLoop = 0;
  }


  client = server.available();
  if (client) {
    if (!connected) {
      Serial.println("Conection not available");
      client.flush();
      connected = true;
      client.stop();
    }
    if (client.available() > 0) {
      Serial.println("Client Available");
      Serial.println("Conection ok");
      int clientread = client.read();
      Serial.print(clientread);
      char charcr = clientread;
      digitalWrite(LED_PIN, HIGH);
      readTelnetCommand(clientread);
      digitalWrite(LED_PIN, LOW);
    }
  }
  if (millis()%2) {
    readPresence();
```

```
  }
  else {
    readOneWire();
  }
}

// This function runs once only, when the Arduino is turned on or reseted.
void setup() {

  inicializarModbus();
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
  //Serial.begin(115200);
  pinMode(SOIL_PIN, INPUT);
  Ethernet.begin(mac, ip, gateway, subnet);
  server.begin();
  Serial.println("Setup");
  delay(1000);
}

void inicializarModbus(){
  master.begin(9600); // baud-rate at 19200
  master.setTimeOut(tiempoEsperaRespuesta); // if there is no answer in 500 ms,
roll over
  u32wait = millis() + 1500;
  u8state = 0;
//  portOne.begin(9600);   // Puerto Software serial
  pinMode(9, OUTPUT);   pinMode(8, OUTPUT);   pinMode(7, OUTPUT);
}

void queryModbus(){
```

```
    //portOne.println(CRC8(datos,8));
  if (indiceIteracion > (numSlaves-1)) {
    indiceIteracion = 0;

//portOne.println("++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++");

Serial.println("++++++++++++++++++++++++++++++++++++++++++++++++++++++
");
    //delay(1000);
  }



  switch ( u8state ) {
    case 0:
      if (millis() > u32wait) u8state++; // wait state
      break;
    case 1:

      telegram.u8id = direccionEsclavo[indiceIteracion]; // slave address              ----
direccion de slave
      telegram.u8fct = 3; // function code (this one is registers read)           ----funcion
modbus

      telegram.u16RegAdd = 0; // start address in slave                        ----direccion
de registro
      telegram.u16CoilsNo = 15; // number of elements (coils or registers) to read    --
---cantidad de registros
      telegram.au16reg = au16data; // pointer to a memory array in the Arduino       -
----almacenamiento de datos
```

```
  master.query( telegram ); // send query (only once)

  u8state++;
  break;
case 2:


  if(master.poll()) // check incoming messages
  {
   if (master.getState() == COM_IDLE) {
     esclavoFallado[indiceIteracion]=0;
     /// conversion de ADC
     adc = (au16data[0]);
     temperatura1 = (au16data[1]);
     //float voltaje = ((adc *5.0) / 1023.0);
     float voltaje = adc;
     float temperatura2 = ((temperatura1 * 500.0) / 1023.0);


     /// impresion port software serial
     // portOne.println("  Holding Registers  Arduino ");
     /*
     portOne.print("slave: ");                        // portOne.print("Registro 05 =  ");
     portOne.print(direccionEsclavo[indiceIteracion]);
     portOne.print(" -> Voltage: ");
     portOne.print(voltaje);
     portOne.print(" Temperatura: ");
     portOne.println(temperatura1);
    */

     Serial.print("slave: ");                        // portOne.print("Registro 05 =  ");
```

```
    Serial.print(direccionEsclavo[indiceIteracion]);
    Serial.print(" -> Voltage: ");
    Serial.print(voltaje);
    Serial.print(" Temperatura: ");
    Serial.println(temperatura2);


    ///portOne.println(a);
    analogWrite(9, au16data[0] >> 2);
    u8state = 0;
    u32wait = millis() + 1500;
    indiceIteracion++;
  }

 }
 else
 {
  if(!(master.getState())){
    u8state = 0;
    u32wait = millis() + 1500;
    //portOne.print(" .. No hay Respuesta de slave -> : ");
    //portOne.println(direccionEsclavo[indiceIteracion]);
    Serial.print(" .. No hay Respuesta de slave -> : ");
    Serial.println(direccionEsclavo[indiceIteracion]);
    indiceIteracion++;
    break;
  }else{
    //portOne.println(" .. Esperando Respuesta ..  ");
    Serial.println(" .. Esperando Respuesta ..  ");
  }
}
```

```
    break;
  }
}
```

Código el esclavo que se le carga al Arduino para que este envíe datos al maestro.

```
#include <ModbusRtu.h>

uint16_t au16data[16];

//++++++++++Definicion de entradas analogicas
const int entradaAnalogica[16] = {A0,A1,A2,A3,A4,A5};
/*
 * Entrada Analogica A0 = voltaje1    = Voltaje de Baterias
 * Entrada Analogica A1 = corriente1  = Corriente de Banoo de Baterias
 * Entrada Analogica A2 = corriente2  = Corriente de Carga
 * Entrada Analogica A3 = temperatura = Temperatura del Site
 * Entrada Analogica A4 = Disponible
 * Entrada Analogica A5 = Disponible
 */
int voltaje1;          //Voltaje de Baterias
int corriente1;        //Corriente de Banoo de Baterias
int corriente2;        //Corriente de Carga
int temperatura;       //Temperatura del Site

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++

//++++++++++Definicion de entradas Digitales
/*Entradas digitales3 =  puerta abierta o cerrada
```

```
 *Entradas digitales4 =  sensor de presencia
 *Entradas digitales5 =  disponoble
 *Entradas digitales6 =  disponible
 *Entradas digitales7 =  disponible
 *Entradas digitales8 =  disponible
 *Entradas digitales9 =  disponoble
 *Entradas digitales10 = disponible
 *Entradas digitales11 = disponible
 *Entradas digitales12 = disponible
 *Entradas digitales13 = disponible
 */
const int entradaDigital[11]={3,4,5,6,7,8,9,10,11,12,13};
// disponible de au16data[16] de 6 - 17

int puertaAC;        // purta abierta o cerrada
int presencia;       // presencia de persona

/**
 *  Modbus object declaration
 *  u8id : node id = 0 for master, = 1..247 for slave
 *  u8serno : serial port (use 0 for Serial)
 *  u8txenpin : 0 for RS-232 and USB-FTDI
 *              or any pin number > 1 for RS-485
 */


int temp []={10,17,11,12,14,18,15} ;
int prom (int temp)
   {
   int y, suma;
  for(byte i=0;i<10;i++)
```

```
  {
    suma=0;

    y =((suma+temp[7])/7);
    return y;
  }
}


Modbus slave(1,0,0); // this is slave @1 and RS-232 or USB-FTDI
void setup() {
  slave.begin( 9600 ); // baud-rate at 19200
}
//int a=0;
void loop() {
  slave.poll( au16data, 16 );
voltaje1 =analogRead(entradaAnalogica[0]);
temperatura = analogRead(entradaAnalogica[1]);




//presencia= analogRead(entradaDigital[0]);

//au16data[0]=  voltaje1*1;
//au16data[1] =12 // temperatura*1;
au16data[0]= 24;
au16data[1]=36;

//au16data[2] = presencia*1;
}
```