UNIVERSIDAD NACIONAL DE LOJA

FACULTAD DE LA ENERGÍA, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES.

"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO EN TIEMPO REAL DE VARIABLES AMBIENTALES DE ÁREAS CRÍTICAS DEL EDIFICIO SIS ECU-911 LOJA."

TESIS DE GRADO PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES.

AUTOR

Carlos Eduardo Flores Gaona

DIRECTOR

Ing. Santiago Abraham Medina León, Mg. Sc.

LOJA-ECUADOR

2019

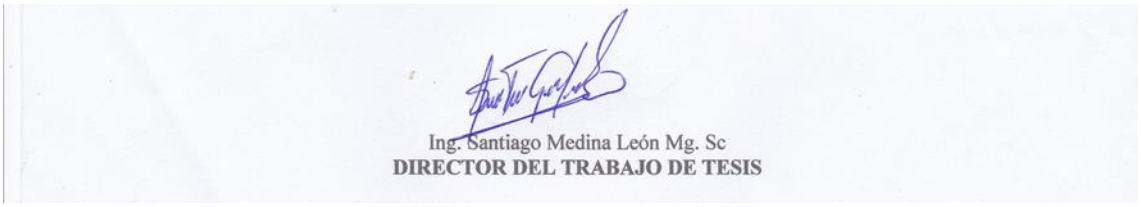
CERTIFICACIÓN

Ing. Santiago Abraham Medina León, Mg. Sc.
DIRECTOR DE TESIS

CERTIFICA:

Haber dirigido, asesorado, revisado y corregido el presente trabajo de tesis de grado, en su proceso de investigación, cuyo tema versa en **"Diseño e implementación de un sistema de monitoreo en tiempo real de variables ambientales de áreas críticas del edificio SIS ECU-911 Loja"**, previa a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones, realizado por el señor egresado: **Carlos Eduardo Flores Gaona**, la misma que cumple con la reglamentación y políticas de investigación, por lo que autorizo su presentación y posterior sustentación y defensa.

Loja, 22 de octubre de 2018



Ing. Santiago Medina León Mg. Sc
DIRECTOR DEL TRABAJO DE TESIS

AUTORÍA

Yo, **CARLOS EDUARDO FLORES GAONA**, declaro ser autor del presente trabajo de tesis y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos o acciones legales, por el contenido de la misma.

Adicionalmente acepto y autorizo a la Universidad Nacional de Loja, la publicación de mi tesis en el Repositorio Institucional – Biblioteca Virtual.

Firma: …  …….

Cédula: 1103753065

Fecha: 29/07/2019

CARTA DE AUTORIZACIÓN DE TESIS POR PARTE DEL AUTOR, PARA LA CONSULTA, REPRODUCCIÓN PARCIAL O TOTAL Y PUBLICACIÓN ELECTRÓNICA DEL TEXTO COMPLETO.

Yo, **CARLOS EDUARDO FLORES GAONA** declaro ser autor de la tesis titulada: **"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO EN TIEMPO REAL DE VARIABLES AMBIENTALES DE AREAS CRÍTICAS DEL EDIFICIO SIS ECU-911 LOJA"**, como requisito para obtener el grado de **INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES**; autorizo al Sistema Bibliotecario de la Universidad Nacional de Loja para que con fines académicos, muestre al mundo la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Digital Institucional:

Los usuarios pueden consultar el contenido de este trabajo en RDI, en redes de información del país y del exterior, con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia de la tesis que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja a los veintinueve días del mes de julio del dos mil diecinueve.

Firma: … 

Autor: Carlos Eduardo Flores Gaona.

Cédula: 1103753065

Dirección: Loja (Diego de Rojas 717-93 y Porfirio Díaz)

Correo electrónico: cefloresg@unl.edu.ec

Teléfono: 072109395 **Celular:** 0990847116

Director de Tesis: Ing. Santiago Abraham Medina León, Mg. Sc.

Tribunal de Grado: Ing. Manuel Pesántes. Mg. Sc.

Ing. Marco Suing. Mg. Sc.

Ing. Ángel Ordoñez. Mg. Sc.

DEDICATORIA

Este proyecto de fin de carrera lo he dedicado a mi familia quienes hacen mi mundo más significativo.

Siento que todo lo puedo lograr con su apoyo.

AGRADECIMIENTO

Agradezco a Dios por permitirme llegar a la culminación de esta carrera profesional, a mi madre Luz Marina, mis hermanos Gabriel, Miguel a mi hermana María Eugenia y en especial a mi esposa Mayra por el apoyo incondicional, siempre gracias.

A la Universidad Nacional de Loja y a toda la planta docente de la carrera de Ingeniería en Electrónica y Telecomunicaciones por los conocimientos compartidos. Y a los compañeros del aula que ahora serán compañeros de la vida. Gracias totales.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

TÍTULO.

“DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO EN TIEMPO REAL DE VARIABLES AMBIENTALES DE ÁREAS CRÍTICAS DEL EDIFICIO SIS ECU-911 LOJA.”

1. RESUMEN.

El monitoreo de variables ambientales al interior de un edificio proporciona información importante para la protección de las personas, equipos e instrumentos dentro del mismo. Además, permite tomar decisiones preventivas o correctivas para extender las condiciones del entorno dentro del edificio. El presente proyecto de tesis tiene como objetivo el diseñar e implementar un sistema de monitoreo de variables ambientales para las áreas críticas del edificio SIS ECU-911 Loja, que brinde información adecuada para proporcionar confort a las personas y la buena operación de los equipos.

El estudio se centra en el diseño y creación de una red alámbrica de sensores basada en el protocolo MODBUS RS-485, que funcionan en un entorno industrial y posibilitan el transporte de datos ambientales de manera confiable. La estructura básica del sistema utiliza el sensor de temperatura y humedad DHT22, sensor de luminosidad BH1750, tarjetas Arduino nano para capturar las medidas de las variables ambientales y el módulo MAX485 para el envío de los datos obtenidos a través de la red diseñada. Una tarjeta Arduino Mega realiza la función de sincronizar las peticiones de datos mediante el módulo MAX485. Los datos recogidos por la red de sensores son enviados por el puerto serial a un servidor local para ser almacenada en una base de datos de PostgreSQL. Estos datos son analizados en la pila Elastic para monitoreo a través de una interfaz web.

El sistema de monitoreo de variables ambientales de áreas críticas fue implementado en las áreas de UPS, *Data Center* y Sala Operativa del edificio SIS ECU-911 Loja. El sistema se encuentra en completo funcionamiento y proporciona información ambiental para tomar decisiones relacionadas con el funcionamiento y ocupación de estas salas. La información obtenida también puede ser útil para otras aplicaciones y líneas de investigación que dependen de datos de temperatura, humedad relativa e iluminación.

ABSTRACT

The monitoring of environmental variables inside a building provides important information for the protection of people, equipment and instruments inside it. It also allows taking preventive or corrective decisions to extend the environmental conditions inside the building. The objective of this thesis project is to design and implement a monitoring system of environmental variables for the critical areas of the SIS ECU-911 Loja building, which provides adequate information to ensure the proper comfort to the people and the good operation of the equipment.

The study focuses on the design and creation of a wired network of sensors based on the MODBUS RS-485 protocol, which operate in an industrial environment and enable a reliable transport of environmental data. The basic structure of the system uses the temperature and humidity sensor DHT22, luminosity sensor BH1750, Arduino Nano cards to capture the measurements of environmental variables and the MAX485 module to send the data obtained through the designed network. A Mega Arduino card synchronizes the data requests through the MAX485 module. The data collected by the sensor network is sent by the serial port to a local server to be stored in a PostgreSQL database. This data is analyzed in the Elastic stack for monitoring through a web interface

The monitoring system of environmental variables of critical areas was implemented in the areas of UPS, Data Center and Operating Room of the SIS ECU-911 Loja building. The system is fully operational and provides environmental information to make decisions related to the operation and occupation of these rooms. The information obtained can also be useful for other applications and lines of research that depend on temperature, relative humidity and lighting data.

2. INTRODUCCIÓN

El edificio del SIS ECU-911 de la ciudad de Loja, es uno de los centros permanentes de respuesta continua *"con base a políticas, normativas y procesos, articula servicios de video vigilancia, botones de auxilio, alarmas, recepción y despachos de atención de emergencias"*. Cuenta con varias áreas de trabajo, de las cuales 3 son consideradas como críticas por el propio personal de la institución. Esto se refiere a que dichas áreas prestan servicios o contienen equipos de mucha importancia para cumplir con el propósito principal de la institución. Si una de estas áreas se ve afectada por condiciones ambientales inadecuadas, se vería afectado el rendimiento y eficiencia de personas y equipos. Por lo tanto, el monitoreo de las condiciones ambientales en dichas áreas críticas es imprescindible y urgente. Con la intervención de este trabajo de tesis se ha implementado un sistema que permite dar seguimiento continuo a las variables ambientales de las áreas críticas del edificio del SIS ECU-911 Loja.

Las condiciones ambientales del entorno tienen una influencia directa en el comportamiento y rendimiento del personal y la eficiencia de los sistemas dentro de una edificación (Forgiarini Rupp, Giraldo Vásquez, & Lamberts, 2015). El monitoreo de las variables que representan esas condiciones ambientales proporciona oportunidades para extender la seguridad y permanencia de personas y equipos, frente a los posibles riesgos que se pueden detectar con la vigilancia continua.

Según Forgiarini Rupp, Giraldo Vásquez, & Lamberts (2015), las variables ambientales que están relacionadas con el confort de las personas y la eficiencia de los equipos dentro de un edificio son: temperatura, humedad, iluminación, ruido y vibración. De estas variables la temperatura, humedad e iluminación fueron consideradas como variables de estudio debido a su influencia directa tanto en equipos como en las personas.

Los aumentos bruscos de la variable de temperatura demandan de atención inmediata para evitar daños considerables en los equipos de la red y posiblemente a las personas. En ambientes de operación como los que presenta el edificio del SIS ECU-911 Loja, el monitoreo de esta variable permite disponer de una alarma en tiempo real que indica realizar una acción correctiva si ésta variable sobrepasa un umbral predefinido.

Las diferentes opciones tecnológicas disponibles en el mercado actual, brindan una amplia gama de posibilidades para realizar monitoreo ambiental en interiores de edificios. Entre todas estas posibilidades se encuentran la tecnología de hardware y software libre Arduino (Academy, 2018). Esta tecnología incluye dispositivos versátiles para control, adquisición y procesamiento de señales ambientales.

Los sistemas de monitoreo de variables ambientales al interior de un edificio requieren al menos de cinco partes (Cabarcas, Montoya, Reyes Betancourt, & Arrieta, 2017) que se definirán como: 1) Adquisición de datos, 2) Transporte de datos, 3) Almacenamiento de datos e información, 4) Gestión de información, y 5) Monitoreo y sistema de alerta.

La adquisición de datos usando la tecnología Arduino requiere al menos de dos componentes, el sensor y el dispositivo de control/procesamiento. Debido a las características específicas de las variables ambientales a monitorear, en esta etapa es imprescindible un conocimiento específico del tipo de sensor y resultado deseado. Así, se pueden encontrar en el mercado unos sensores más sofisticados que otros en cuanto a precisión y resolución (velocidad de respuesta).

El trasporte de los datos desde los dispositivos de registro de las señales ambientales hasta el puerto serial de donde empezará su almacenamiento y gestión depende de la estructura y configuración de la red. La transmisión de los datos se puede realizar de forma inalámbrica, como por ejemplo (Suárez Barón & Suárez Barón, 2014), o alámbrica, como el caso de este trabajo. Estas dos vías de transmisión de datos están enmarcadas dentro de protocolos y normas estandarizadas que garantizan su correcto funcionamiento. Algunas de estos estándares permiten más prestaciones que otros, pero a su vez requieren condiciones más específicas para su funcionamiento. El Protocolo de comunicación MODBUS RS-485 (Texas Instruments, 2008) usado en este trabajo ofrece amplias posibilidades para la red alámbrica implementada para el sistema de monitoreo de variables ambientales propuesto en este estudio.

El almacenamiento de los datos e información, la gestión de la información y las interfaces gráficas o web del sistema no dependen directamente de la configuración de la red alámbrica de adquisición de datos. Sin embargo, son un componente importante en la toma de decisiones. Estas etapas proveen al personal encargado las herramientas para dar seguimiento a las condiciones ambientales registradas y actuar cuando se requiera

atención a alguna de ellas. De la misma manera que sucede con los dispositivos electrónicos, existen muchas opciones tecnológicas para suplir las necesidades de estas etapas (Camargo Vega, 2015). Así, por ejemplo, la Pila Eastic, presenta opciones de sincronización, indizado y visualización mediante sus herramientas Logstash, Elasticsearch y Kibana, respectivamente.

El desarrollo del proyecto contempló cuatro etapas en las que incluyó un análisis de los dispositivos electrónicos disponibles para selección de los más adecuados para utilizar. Entre ellos, se analizaron sensores y tarjetas electrónicas para la captura de las variables ambientales de temperatura, humedad e iluminación.

El diseño del sistema considera los requerimientos mínimos definidos por (Cabarcas, Montoya, Reyes Betancourt, & Arrieta, 2017). El transporte de los datos se realiza mediante una configuración de red alámbrica basada en el protocolo MODBUS RS-485. A partir de ahí se gestiona los datos e información con tecnologías de software libre, como: PostgreSQL 10.0, Pila Elastic.

El presente trabajo de tesis fue orientado mediante el siguiente Objetivo General: "Diseñar e implementar un sistema de monitoreo en tiempo real de variables ambientales en áreas críticas del edificio SIS ECU-911 Loja". Las áreas definidas por el personal técnico de la institución son: área de UPS, área de *Data Center* y Sala operativa. Los objetivos específicos que permitieron el desarrollo del proyecto, fueron:

1. Investigar y determinar los sensores y tipos de tarjetas electrónicas a emplear para la medición de las variables ambientales y diseño del sistema.
2. Diseñar la red alámbrica a implementar para la comunicación de sensores y transmisión de datos.
3. Diseñar el interfaz Web para gestión y presentación de datos obtenidos.
4. Implementar y comprobar el correcto funcionamiento del sistema de vigilancia y monitoreo de Humedad, Temperatura y Luminancia en áreas críticas del edificio SIS ECU-911 Loja.

3. REVISIÓN DE LITERATURA

3.1 Dispositivos para la adquisición de datos.

3.1.1 Sensores.

Los sensores son dispositivos que tienen la capacidad de adquirir señales físicas presentes en un entorno y transformarlas en señales eléctricas (Cárdenas Esparza, 2013). Una vez adquiridas las señales eléctricas pueden ser procesadas mediante ordenadores para representarlas como variables medibles de ese entorno.

Existe un elevado número de sensores por lo que clasificarlos es una tarea compleja. Sin embargo, se enlistan algunos tipos de sensores de acuerdo a criterios de investigación generalizados (Mora, 2011) (Pallás Areny, 2004):

- Según aporte de energía: moduladores y generadores.
- Según naturaleza de la señal de salida: analógicos o digitales.
- Modo de operación: de deflexión y de comparación

Los sensores digitales serán descritos con mayor énfasis en los siguientes apartados debido a que representan una parte importe en el diseño y funcionamiento del sistema de monitoreo.

3.1.1.1 Sensores Digitales.

Los sensores digitales, son sensores que ofrecen directamente a su salida una señal en forma discreta o en forma de saltos. Por la simplificación que suponen de señales digitales y la alta inmunidad a interferencias electromagnéticas son ampliamente utilizadas en sistemas de medida y control. (Mora, 2011) (Pallás Areny, 2004)

En una forma simplificada, los sensores están compuestos por dos bloques funcionales como el acondicionador de señal y el procesamiento de datos. La agrupación de estos dos bloques funcionales conforma el sistema embebido de un sensor digital (Mora, 2011). En la imagen de la Figura 1 se presenta el esquema de este sistema.

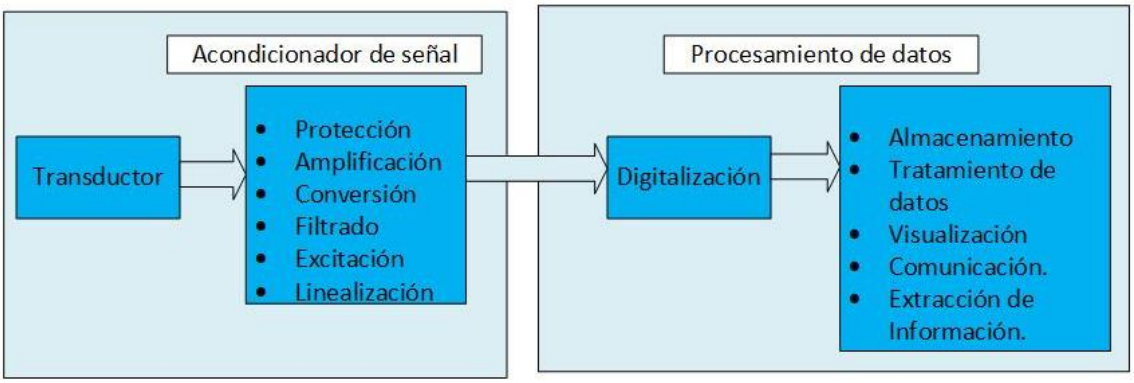


Figura 1. Sistema embebido. Fuente [23]

3.1.1.2 Acondicionador de señal en sensores digitales.

La señal captada por el sensor para proporcionar datos, se trata en diferentes etapas. El conjunto de subprocesos para acondicionar la señal tiene el propósito de generar una variable óptima para el procesamiento de la información captada.

De acuerdo con (Pallás Areny, 2004) y (Mora, 2011) en la etapa de acondicionamiento de señal, el conjunto de subprocesos que intervienen, son los que se enlistan a continuación:

- **Protección:** aplicación de una diferencia de potencial proporcional a la medida obtenida.
- **Amplificación:** subproceso de aumento de nivel de la señal obtenida.
- **Conversión por medio de opto-acopladores:** fase para proveer aislamiento de los sistemas eléctricos que componen los sensores.
- **Filtrado:** tratamiento de la señal para eliminar ruidos de alta frecuencia y evitar pérdidas de exactitud.
- **Excitación:** aplicación de corriente o tensión para producir la variación proporcional a la magnitud a medir.
- **Linealización:** métodos para convertir la respuesta del sensor en términos lineales.

Dentro de las fases de la etapa de acondicionador de señal en los sensores digitales, es importante ampliar la información de las técnicas de linealización.

- **Técnicas de Linealización.**

La linealización hace referencia a una serie de métodos que pretenden incrementar la linealidad de un sistema, esto es, que la señal procesada se presente idealmente sin alteraciones. (Cárdenas Esparza, 2013)

La linealidad de un sensor está definida por su curva de calibración estática. Describe cuanto se aleja esta curva de la curva de calibración ideal (Pallás Areny, 2004). Entonces el alejamiento de la curva real de linealidad depende de la no linealidad del sistema producto de los elementos que lo componen como el propio sensor, los circuitos analógicos de acondicionamiento, amplificadores, filtros, entre otros. (Cárdenas Esparza, 2013)

Existen diferentes técnicas digitales de linealización utilizadas para incrementar la linealidad de un sistema sin reducir la eficiencia. Alguna de estas técnicas se describe a continuación (Marta, 2015):

- **Conversión AD no lineal:** proporcionan un código lineal con el parámetro de medida sensado a través de un transductor no lineal.
- **LUT (tabla de búsqueda):** tabla que almacena valores digitales correspondientes con los datos que proporciona el sensor, aunque la resolución no es constante.
- **Linealización a trozos:** divide el rango del parámetro de entrada en *N* segmentos iguales o distintos y se aproxima de manera lineal cada tramo. En la Figura 2 se muestra el tipo de linealización a trozos.
- **Cálculo o estimación de la función característica:** parte de la característica del sensor si es conocida, sino calcula a través de técnicas de interpolación, estadísticas o redes neuronales.

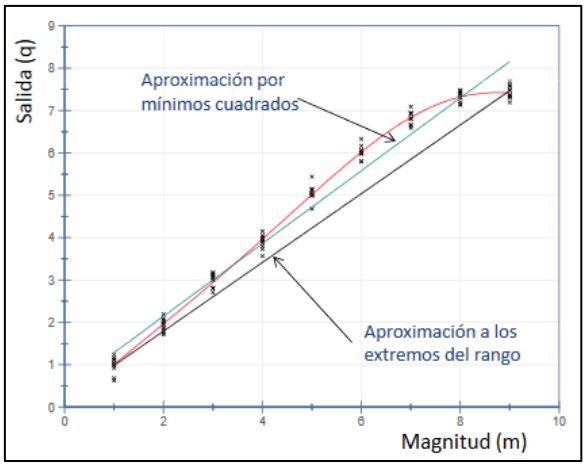


Figura 2. Curva real e ideal de Linealidad de un sensor. Fuente [23]

Luego de acondicionar la señal capturada por el sensor, el siguiente conjunto de subprocesos realiza el tratamiento de la señal para entregar un dato de salida.

- **Procesamiento de datos en sensores digitales.**

Los microprocesadores son los encargados de realizar la transformación de datos, por lo cual requieren de entradas digitales de señal. Una vez acondicionada la señal el siguiente proceso es digitalizarla. Con esto se logra que la señal sea más inmune al ruido, así como a otras interferencias. (Cárdenas Esparza, 2013)

La función de digitalización se compone de tres procesos importantes (Pallás Areny, 2004) como:

- **Muestreo:** toma de muestras periódicas aplicando el teorema de Nyquist para evitar pérdida de datos por valores no muestreados.
- **Cuantificación:** asignación de valor en nivel de voltaje a cada una de las muestras, cuantificadas con bits.
- **Codificación:** trasformación de los valores obtenidos durante la cuantificación en códigos binarios o cualquier otro tipo de codificación preestablecida.

La combinación entre un sensor y un microprocesador se denomina a menudo sensor inteligente, el mismo que está basado casi en su totalidad en elementos miniaturizados dentro de un mismo encapsulado (Cárdenas Esparza, 2013).

La agrupación de todas estas funciones, además de la capacidad de comunicación de los sensores digitales los vuelven muy útiles para los sistemas de medida y control existentes hasta la actualidad.

3.1.1.3 Selección del sensor digital para medir variables ambientales de humedad relativa y temperatura para el diseño y funcionamiento del sistema de monitoreo.

Existen en la actualidad gran variedad de sensores para medir señales de variables ambientales de temperatura y humedad relativa. Algunos de los sensores consultados pueden ser configurados para funcionar con las tarjetas Arduino o Raspberry pi. Las características técnicas de funcionamiento, el ambiente en el que puede ser utilizado el sensor y el costo, se muestran en la Tabla 1.

Tabla 1. Características de sensores de humedad relativa y temperatura consultados.

	Rango de Medición		**Ambiente de instalación**	**Compatibilidad con tarjetas Arduino o Raspberry**	**Costo en dólares USA**
Sensor	**Humedad Relativa %**	**Temperatura °C**			
Sensor digital HC2S3	-50% +100%	-50°C +100°C	Interiores, industriales	✗	150,00
Sensor digital SHT10	0% - 100%	-10°C +80°C	Exteriores, industriales, interiores	✓	34,00
Sensor digital DHT11	20% 80%	0°C +50°C	Interiores, industriales	✓	5,00
Sensor digital DHT22	0% +100%	-40°C +80°C	Interiores, industriales	✓	10,00

Fuente [El Autor]

El sensor HC2S3 (ver Tabla 1), es un dispositivo de alta precisión que tiene funcionamiento autónomo (no requiere ningún dispositivo de control) para la adquisición de las señales de temperatura y humedad relativa. Los datos de temperatura y humedad relativa del ambiente que captura el sensor se pueden consultar en un interfaz, instalado previamente en una computadora, cuyo archivo ejecutable se adquiere junto con el sensor. El alto costo que tiene el sensor representa un inconveniente importante, para utilizarlo en el diseño del sistema de monitoreo.

El sensor SHT10 tiene amplio rango de medición (ver Tabla 1) para capturar las señales de las variables de humedad relativa y temperatura. El sensor SHT10 puede instalarse en lugares internos como externos, en ambientes industriales lo cual representa un beneficio importante para el diseño del sistema de monitoreo. Aunque el costo del sensor SHT10 no es muy elevado (ver Tabla 1) se debe considerar seriamente esta condición, puesto que, se necesitan algunos elementos electrónicos adicionales (tarjetas para el control, baterías, entre otros) para el completo funcionamiento del sensor, lo que encarecerá, sin lugar a duda, la inversión total del proyecto.

El diseño del sistema de monitoreo puede considerar utilizar cualesquiera de los sensores DHT11 o DHT22 debido a que muestran parámetros de operación muy similares (ver Tabla 1). Sin embargo, el sensor DHT22 ofrece un mejor rango de medición de las variables de temperatura y humedad relativa en comparación con el sensor DHT11. El sensor DHT22 puede ser instalado en un entorno industrial dentro de un edificio. Según la referencia mostrada en la Tabla 1 el costo del sensor DHT22 es accesible y el control del sensor se lo puede realizar desde una tarjeta Arduino o Raspberry para la adquisición de datos.

A demás de las características mostradas en la Tabla 1 sobre el sensor DHT22, en el mercado local como nacional existe alta oferta del dispositivo por lo que puede ser adquirido fácilmente y a un precio cómodo, por lo que el sensor DHT22 es el dispositivo que se utilizará en el diseño del sistema propuesto. Para mayores especificaciones técnicas acerca del sensor DHT22 se puede consultar el apartado 3.1.2 del presente informe.

3.1.1.4 Selección del sensor digital para la captura de las señales de iluminación para el diseño y funcionamiento del sistema de monitoreo.

Los datos que se adquieran de la variable de iluminación de un espacio limitado ofrecen oportunidades para garantizar el confort de las personas y el ahorro energético en cuanto a cantidad conveniente de iluminación (lux/m^2). Las señales de la iluminación del ambiente pueden ser capturadas y los resultados entregados como datos por cualesquiera de los sensores mostrados en la Tabla 2.

Tabla 2. Características técnicas y parámetros generales de operación de sensor de iluminación.

	Rango de Medición	**Ambiente de instalación**	**Compatibilidad con tarjetas Arduino o Raspberry**	**Costo en dólares USA**
Sensor	**lux**			
Sensor digital BH1750	1 - 65535	Interiores, exteriores (luz artificial)	✓	6,00
Sensor digital TSL2561	0.1 – 40000	Exteriores, Interiores (luz artificial)	✓	16,00
Sensor analógico KY -018	0 – 1023 (utilizar ADC)	Interiores, Exteriores (luz natural)	✓	12,00

Fuente [El Autor]

El diseño del sistema de monitoreo requiere de un sensor que capture con alta resolución las señales de iluminación de un ambiente limitado y entregue los datos en forma digital para posterior almacenamiento. El análisis comparativo de características técnicas y operativas de los sensores consultados (ver Tabla 2) permite elegir el sensor BH1750 por su precio en el mercado, compatibilidad con las tarjetas Arduino o Raspberry y su amplio rango de medición en luxes de la luz artificial en un área interior.

En los siguientes apartados se describen con mayor detalle las características técnicas de los sensores digitales DHT22 y BH1750 que se utilizan en el diseño del sistema de monitoreo para la captura de las señales de las variables ambientales como humedad relativa, temperatura e iluminación respectivamente.

3.1.2 Sensor de Temperatura y Humedad Relativa DTH22

El sensor DTH22 es un dispositivo que entrega datos digitales de temperatura y humedad relativa de alta precisión, compuesto por tres elementos principales como un sensor de humedad capacitivo, un termistor para medir el aire circundante y un convertidor analógico digital (ADC) que realiza la conversión de las señales analógicas a digitales. En la imagen de la Figura 3 se puede observar el sensor DHT22, así como los pines de conexión. (ELECTRONICLAB, 2018)

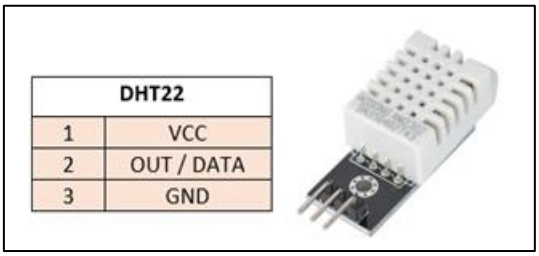


Figura 3. Sensor de temperatura y humedad DHT22. Fuente [16]

El sensor DTH22 tiene un microprocesador de 8 bits de tamaño pequeño y bajo consumo de energía. La distancia de transmisión de la señal es de hasta 20 metros en dependencia del medio físico utilizado para el envío de datos, conectado por un solo interfaz de bus (pin 2 del sensor). En la Tabla 3 se presenta adicionalmente algunas especificaciones técnicas del sensor DHT22. (ROHM Co., Ltd, 2011)

Tabla 3 Datos técnicos de módulo sensor DHT22

Módulo Sensor DHT22	
Descripción	**Unidades/Valores**
Comunicación	Un solo Bus
Rango de medición	-40°C hasta 80°C Temperatura 0% hasta 100% Humedad Relativa
Voltaje de operación	3.3V – 5V
Tiempo de muestreo	2 segundos/ 0.5Hz
Consumo de energía	30 micros Watts.
Precisión de mediciones	±0.5°C Temperatura; ±2% de Humedad Relativa
Sensibilidad	0.1 °C Temperatura; 0.1% Humedad Relativa.

Fuente [El Autor]

- **Protocolo de comunicación del sensor DHT22.**

El sensor DHT22 tiene un protocolo de comunicación propietario de 1-wire (1-cable). Cada sensor dispone de su propio bus de datos (pin 2 del sensor) y solo puede conectarse

un único sensor dentro del bus de datos. (Gomez Blazquez, 2016) (López Esquembri, 2017)

Cuando el microcontrolador empieza la comunicación configura el pin de datos como salida, poniendo el voltaje (VCC) en alto, después envía la señal de *"Inicio"* y establece un nivel (*GND*) bajo durante un intervalo de tiempo de 18 milisegundos y a continuación regresa a un nivel (*VCC*) alto entre 20-40 microsegundos. (Gomez Blazquez, 2016) (López Esquembri, 2017)

A continuación, el microcontrolador coloca el pin de datos como entrada, para esperar la respuesta del sensor, establece un nivel bajo durante los 80 microsegundos siguientes e inmediatamente en alto por 80 microsegundos más.

Después de este procedimiento el sensor se encuentra listo para transmitir los datos de humedad y temperatura, enviando 40 bits de información seguidos. Cuando finaliza la transmisión de los 40 bits, el sensor sitúa el bus a nivel bajo durante 50 microsegundos y enseguida a nivel alto, con lo cual libera el bus y se establece en modo de bajo consumo hasta que el microcontrolador envíe nuevamente una señal de *"Inicio"*, que tardará 2 segundos y repetirá todo el proceso (López Esquembri, 2017) (Gomez Blazquez, 2016). En la Figura 4 se grafica el procedimiento.

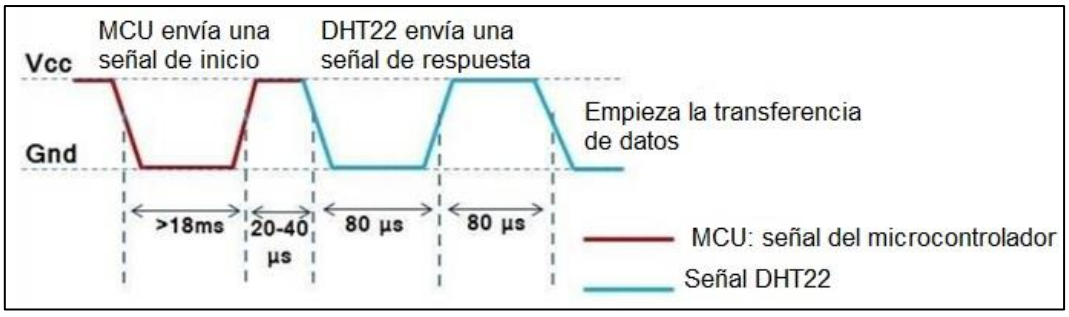


Figura 4. Señal de Inicio y respuesta del sensor. Fuente [17]

- **Estructura de la trama de información.**

La trama de información de los datos de temperatura y humedad relativa que envía el sensor DHT22 tiene un tamaño de 40 bits (5bytes) estructurada de la manera como se muestra en la Figura 5:

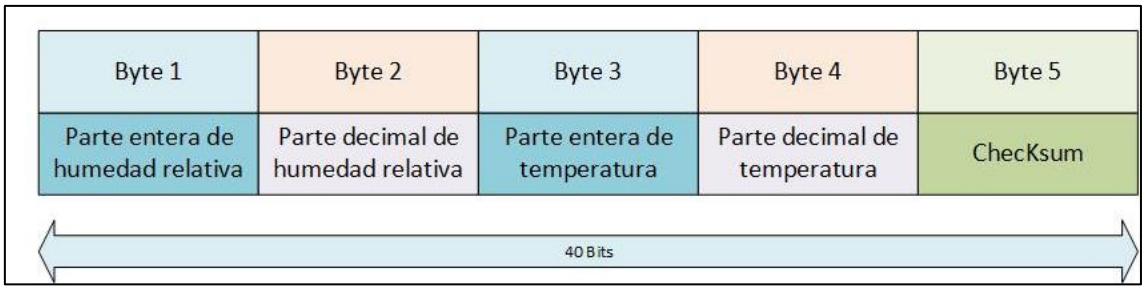


Figura 5. Estructura de trama de datos sensor DHT22. Fuente [El Autor]

El byte de *Checksum* (comprobación) es importante para verificar que la trama es la correcta y no hay errores de transmisión. Se calcula sumando los 4 primeros bytes de temperatura y humedad relativa, el resultado de esta suma debe ser igual al byte de *Checksum*, (Gomez Blazquez, 2016).

3.1.3 Módulo Sensor de Luminosidad BH1750

El módulo BH1750, es un sensor digital que permite captar con una alta sensibilidad y resolución la intensidad de luz ambiente. Cuando se conecta con un procesador autómata (como Arduino) forma una unidad de medición de iluminación en unidades luxes (unidad del sistema internacional para la iluminancia), denominada luxómetro. (Liu, s.f.)

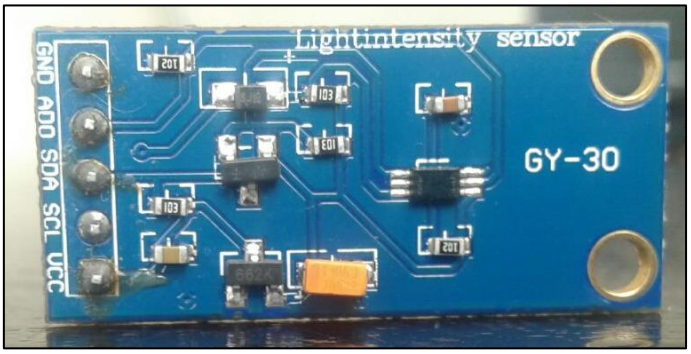


Figura 6. Módulo BH1750. Fuente [El Autor]

La iluminancia se determina por la relación entre el flujo luminoso (cantidad de luz emitida por una fuente de luz) y un área determinada (superficie en metros). El módulo BH1750 se presenta en la imagen de la Figura 6.

El módulo BH1750 tiene un conversor interno ADC de 16 bits que le permite entregar una salida digital en comunicación *I2C*. Proporciona mediciones de intensidad de iluminación directamente en unidades Lux equivalente a un $lumen/m^2$ (ROHM Co.,

Ltd, 2011). En la Tabla 4 se describen las especificaciones técnicas más relevantes del sensor BH1570.

Tabla 4. Datos Técnicos de Módulo BH1750

Módulo Sensor de luminosidad BH1750	
Descripción	**Unidades/Valores**
Comunicación	Bus I2C
Rango de medición	1-6535 luxes
Voltaje de operación	3.3V-5V
Consumo de corriente promedio	Bajo: 0.1 uA
Inmunidad al ruido	Rechazo de ruido 50Hz-60Hz
Direcciones de operación I2C	ADDR= HIGH(5V): 0X5C ADDR= LOW (GND): 0X23C
Modo de Resolución	HIGH MODE 2: 0.5 lx/120ms. HIGH MODE :1 lx/120ms. LOW MODE: 4 lx/16ms

Fuente: [El Autor]

A continuación, se describen los componentes principales del módulo BH1750:

- PD: Foto-diodo con respuesta similar a la percepción de los ojos humanos.
- AMP: Integración OPAMP para la conversión de corriente a voltaje.
- ADC: Convertidor AD para obtener datos digitales de 16 bits.
- *Logic + I2C Interface*: Cálculo de luz ambiente e interfaz de bus $I2C$.
- OSC: Oscilador interno típico 320 KHz. Es un reloj interno lógico. (ROHM Co., Ltd, 2011)

En la imagen de la Figura 7 se presenta el diagrama de bloques de composición del módulo BH1750.

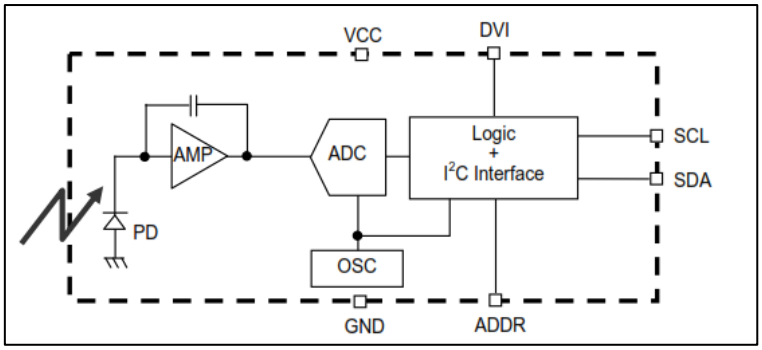


Figura 7. Diagrama de bloques de BH1750. Fuente [30]

- **Protocolo de comunicación del sensor BH1750.**

El sensor BH1750 tiene un protocolo de comunicación mediante bus *I2C* (Bus de circuito Inter integrado), que es un protocolo serial con velocidad de transmisión en versión estándar de 100 kbps y 5 Mbps en versión ultra rápida. Soporta en el mismo bus de comunicación múltiples maestros y esclavos; básicamente emplea dos líneas interdependientes para establecer la comunicación como SDA (*Serial Data /* dato serial) y SCL (*Serial Clock /* reloj serial). (Tutoriales, 2018)

Cuando las líneas SDA y SCL se encuentran en estado alto, se establece que el bus de comunicación está libre o en condición de recibir/enviar algún dato. El cambio de estado de 1 a 0 en la línea SDA mientras la línea SCL se mantiene en alto determina la condición de inicio o *"Start"*, cuando se da este proceso se considera que el bus de comunicación está ocupado. La condición de parada o *"Stop"* se determina cuando existe un cambio de estado de 0 a 1 en la línea SDA mientras la línea SCL se mantiene en alto. El siguiente proceso determinará la condición de bus libre con los parámetros explicados al inicio de este párrafo (ROHM Co., Ltd, 2011). Este procedimiento se grafica en la imagen de la Figura 8.

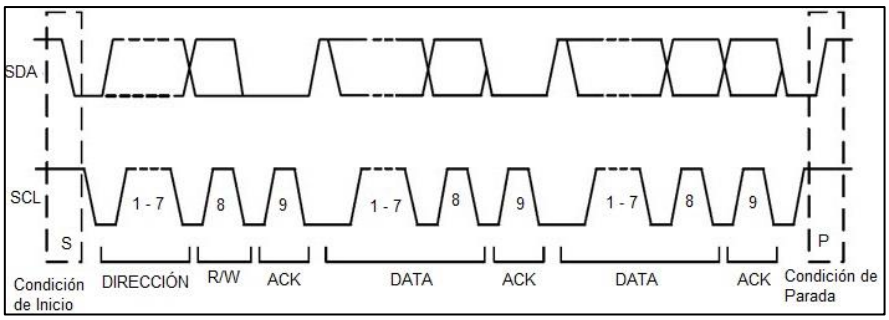


Figura 8. Protocolo de comunicación I2C de sensor BH1750. Fuente [30]

- **Estructura de la trama de información de protocolo I2C.**

Luego de establecerse la condición de *"Start"* los bits de información se transmiten empezando por el más significativo *(MSB)*, manteniendo constante el valor del bit durante todo el período de la señal de la línea SCL. El cambio de bit se realiza por el cambio de estado de la línea SCL (de alto a bajo) durante el flanco descendente. (Aprendiendo Arduino, 2016) (ROHM Co., Ltd, 2011)

El bit R/W no pertenece a los datos de información, sino que permite direccionar el dispositivo sobre el cuál se establece la comunicación y la condición que se va a realizar

en el mismo como leer *(Read)* establecido por un nivel alto o 1 y escribir *(Write)* establecido por nivel bajo o 0. (Aprendiendo Arduino, 2016) (ROHM Co., Ltd, 2011). El bit de ACK (bit de comprobación) es enviado por quien recibe la información y se establece como verdadero en estado bajo o 0 (ROHM Co., Ltd, 2011). En la gráfica de la Figura 9 se puede ver la estructura de la trama.

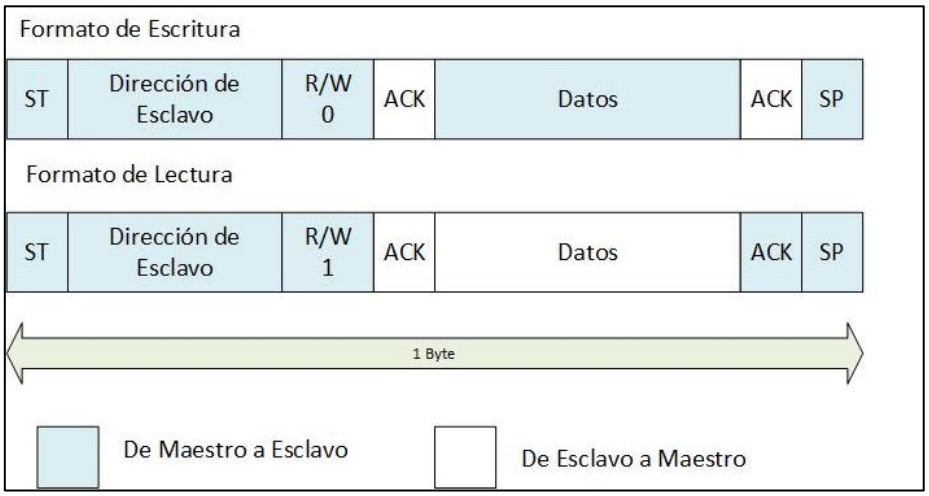


Figura 9. Estructura de trama de información de BH1750 por Bus I2C. Fuente [30]

3.1.4 Entorno de Desarrollo Integrado Arduino y tarjetas electrónicas.

El Entorno de Desarrollo Integrado (IDE) Arduino es una plataforma para la programación de microcontroladores, compuesta de una serie de herramientas de software que soporta lenguajes de programación basados en programación C++ (ARDUINO , 2018). El entorno Arduino fue escrito en java y está basado en Processing.

El IDE de Arduino, en términos generales, se compone de un editor de texto (donde se escribe el programa), compilador (traducción del programa a lenguaje máquina), depurador (corrector de errores) y una GUI (constructor de interfaz gráfica) en el mismo entorno de programación. (ARDUINO , 2018)

El software de Arduino tiene características de software libre (ver apartado 3.3.1) de código abierto lo que ha permitido amplio uso y expansión. Se aplica en gran variedad de ideas, que son compartidas en internet por distintos usuarios, para aplicaciones sencillas o estructuras que se utilizan en la creación de hardware compatible con la tecnología Arduino.

3.1.4.1 Tarjetas Arduino.

Las características de software libre (ver apartado 3.3.1) de Arduino han permitido la creación de una serie de tarjetas hardware para la programación de microcontroladores. Un resumen general se muestra en Tabla 5 como cuadro comparativo de las características técnicas de las tarjetas electrónicas Arduino que fueron consultadas, así como el costo de cada una de las tarjetas.

Tabla 5. Comparación de características de las tarjetas Arduino.

Tarjeta Arduino	**Memoria Flash KB**	**Pines I/O digital**	**Pines I/O analógicos**	**Costo en dólares USA**
UNO	32	14/6	6/0	15,00
MEGA	256	54/15	16/0	20,00
NANO	16	14/6	8/0	10,00
DUE	512	54/12	12/2	18,00
LEONARDO	32	20/7	12/0	12,00

Fuente: [El Autor]

Las tarjetas Arduino seleccionadas para el desarrollo del diseño y funcionamiento del sistema, son las siguientes:

- Tarjeta Arduino Mega.
- Tarjeta Arduino Nano.

Las tarjetas Arduino son utilizadas para el desarrollo del diseño del sistema de monitoreo como componentes físicos y para configurar mediante programación el funcionamiento lógico de la red alámbrica que se necesita implementar.

Las tarjetas Arduino brindan prestaciones para el sistema de monitoreo que pueden ser aprovechadas de la mejor manera. La diferencia más notable se observa en cuanto a la capacidad y velocidad de procesamiento de los microcontroladores integrados en cada tarjeta (nano y mega), como también la cantidad de pines de entrada/salida digitales o analógicos que poseen (Arduino & Geniuno Products, 2018).

El módico costo de las tarjetas Arduino (nano y mega) y la oferta en el mercado representan un motivo notable para el uso de estos dos dispositivos. Además, el funcionamiento de las dos tarjetas puede ser configurado en el mismo entorno de

programación para operar conjuntamente y formar una unidad de procesamiento mucho más grande, así como eficiente.

A continuación, se muestran las características más importantes de los dos tipos de tarjetas Arduino nombradas anteriormente.

- **Tarjeta Arduino Mega.**

La tarjeta Arduino Mega es una tarjeta electrónica de desarrollo de código abierto construida con un microcontrolador ATmega2560. En la Figura 10 se muestra este tipo de tarjeta, las características principales se describen a continuación:

- Memoria Flash: 256 KB.
- Pines (entrada/salida) I/O digitales: 54 I/O, se pueden usar 14 I/O como salida PWM.
- Entradas Analógicas: 16
- Reloj de sincronización (velocidad): 16 MHz.
- Corriente DC: 40-50 miliamperios. (Arduino & Geniuno Products, 2018)

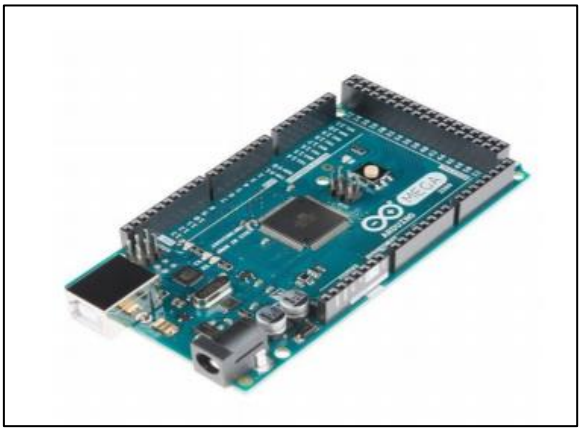


Figura 10. Tarjeta Arduino Mega. Fuente [5]

- **Tarjeta Arduino Nano.**

La tarjeta Arduino Nano es una tarjeta electrónica compacta y muy completa basada en el microcontrolador ATmega328. Cumple con casi todas las funciones de una tarjeta Arduino uno, con la desventaja que tiene menos memoria (Arduino & Geniuno Products, 2018). En la Figura 11 se puede observar este tipo de tarjeta, las características principales se describen a continuación:

- Memoria Flash: 16 KB.
- Pines (entrada/salida) I/O digitales: 14 I/O, se pueden usar 6 I/O como salida PWM.
- Entradas Analógicas: 8
- Reloj de sincronización (velocidad): 16 MHz.
- Corriente DC: 40 miliamperios. (Arduino & Geniuno Products, 2018)

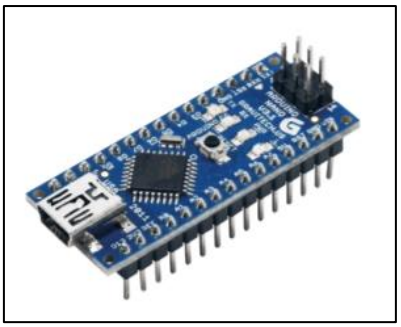


Figura 11. Tarjeta Arduino Nano. Fuente [5]

3.2 Estándares de comunicación serial para el transporte de datos.

Los estándares seriales que describen especificaciones técnicas de funcionamiento y se utilizan para el intercambio de datos entre máquinas, son regulados por organismos internacionales de normalización como la Asociación de Industrias Electrónicas (EIA) y la Asociación de la Industria de las Telecomunicaciones TIA. (Martinez, 2014)

De acuerdo con los organismos EIA/TIA, existen diferentes normas para aplicaciones que impliquen comunicación de dispositivos en un entorno industrial. Los estándares seriales para comunicación son: RS-232, RS-422 y RS-485 (Martinez, 2014) (Samboya, 2012). Un breve resumen de las características de funcionamiento de los estándares seriales consultados se muestran en la imagen de la Figura 12.

Características EIA/TIA-232, EIA/TIA-422, EIA/TIA-485			
NORMAS	**EIA/TIA-232**	**EIA/TIA-422**	**EIA/TIA-485**
Modo de Comunicación	Full dúplex	Full dúplex Hall dúplex	Full dúplex Hall dúplex Semi dúplex
Señal	Desbalanceada	Balanceada Diferencial	Balanceada Diferencial
Cantidad Máx. Controladores - Receptores	1 Driver- 1 Receptor	5 Drivers- 10 Receptores	32 Drivers – 32 Rceceptores
Máx. Distancia/ Máx. Tasa de Datos	15 metros/ 19.2Kbps	1200 metros/100 Kbps 12 metros/10Mbps	1200 metros/100 Kbps 12 metros/10Mbps
Cableado	Single ended	Single-ended Multi-drop	Multi-drop
Corriente de salida	500 mA	150 mA	250 mA

Figura 12. Comparación de estándares seriales. Fuente [27]

El estándar serial EIA/TIA-485 tiene características de funcionamiento destacables (ver Figura 12) como el modo de comunicación, la cantidad de dispositivos transmisores como receptores y la distancia máxima de transmisión de datos que pueden aprovecharse de la mejor manera. El estándar de comunicación serial EIA/TIA-485 es ampliamente utilizado y muchas de las tecnologías desarrolladas basadas en este estándar, se han enfocado en aplicaciones de monitoreo y control. Por lo expuesto, el estándar EIA/TIA-485 se utilizará para el diseño y funcionamiento del sistema de monitoreo en estudio para la comunicación de dispositivos.

3.2.1 Estándar de comunicación serial EIA/TIA -485

El estándar TIA/EIA-485 es también llamado RS-485 (Estándar Recomendado, por sus siglas en inglés RS), desarrollado conjuntamente por dos asociaciones comerciales, como la Asociación de Industria Electrónicas (EIA) y la Asociación de Industria de las Telecomunicaciones (TIA). EIA-485. Es un estándar de comunicaciones muy utilizado

en aplicaciones donde se requiere la adquisición y el control de datos. (Texas Instruments, 2008)

EIA/TIA -485-A permite establecer la comunicación entre dispositivos en modo *simplex*, *half dúplex*, *full dúplex* en transmisión punto a punto o punto a multipunto, en otras palabras, un nodo central puede conectarse con varias unidades transmisoras remotas. La transmisión punto a multipunto es de tipo diferencial que permite velocidades de hasta 10 Mbps en distancias de hasta 50 metros y 200 Kbps en distancias de hasta 1200 metros. (Samboya, 2012). El modo diferencial que se utiliza en la transmisión punto a multipunto se refiere al hecho de que se emplean dos señales para transmitir y dos para recibir, con su respectiva referencia a tierra como se puede ver en la Figura 13.

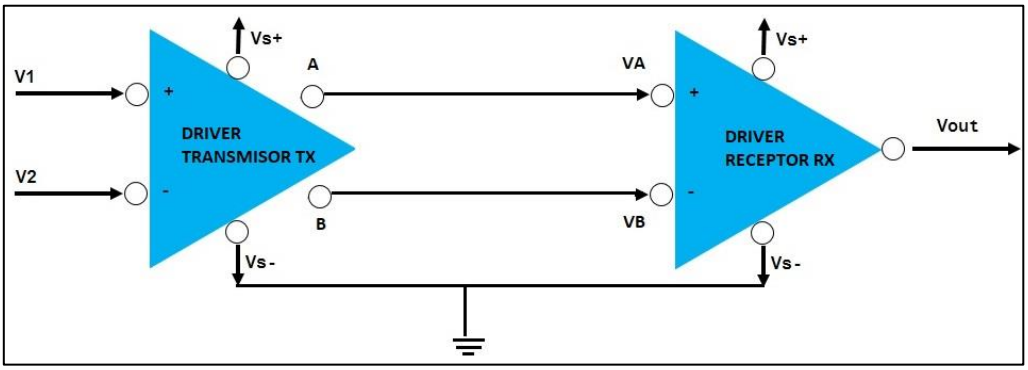


Figura 13. Transmisión punto a multipunto diferencial. Fuente [6]

La imagen de la Figura 13 muestra, de acuerdo a la norma EIA/TIA-485, dos señales como **A** y su complemento **B.** En el driver transmisor, una entrada alta TTL (Lógica Transistor-Transistor) ocasiona que la señal **A** sea más positiva que **B**, mientras que una entrada baja TTL provoca que la señal **B** sea más positiva que la señal **A**. En el driver receptor, si la señal **A** es más positiva que **B**, la salida TTL será un 1 lógico y si la señal **B** es más positiva que la señal **A**, la salida TTL será un 0 lógico. (Texas Instruments, 2008) (Samboya, 2012)

En términos de voltaje, el driver de transmisión RS-485 tiene una salida válida de +1.5 voltios. En los drivers de recepción la diferencia de voltaje entre las señales **A** y **B** requiere de 0.2 voltios. Esto indica que si **A** es al menos 0.2 voltios más positiva que **B**, el receptor tendrá a la salida un 1 lógico y si **B** es 0.2 voltios más positivo que **A**, tendrá a la salida un 0 lógico. Cuando la diferencia entre las señales **A** y **B** es menor a 0.2 voltios existirá un nivel lógico indefinido lo que indica un error en la transmisión/recepción de la

información. En la Tabla 6 se presenta un resumen de las especificaciones técnicas de la norma EIA/TIA – 485. (Texas Instruments, 2008) (Soriano Miras, 2012)

Tabla 6. Resumen de Datos Técnicos EIA-485.

EIA/TIA - 485 RESUMEN DE ESPECIFICACIONES				
Parámetros	**Condiciones**	**Min.**	**Max.**	**Unidad**
Voltaje Conductor Circuito Abierto.		1.5 -1.5	6 -6	V V
Voltaje de salida del conductor Cargado	$R_{LOAD} = 54\Omega$	1.5 -1.5	5 -5	V V
Corriente de cortocircuito de salida del conductor	Por salida a +12V o -7V		± 250	mA
Tiempo de Rise de salida del Driver	$R_{LOAD} = 54\Omega$ $C_{LOAD} = 50pF$		30	% Ancho de Banda Bit.
Voltaje de modo común del Drive	$R_{LOAD} = 54\Omega$	-1	3	V
Sensibilidad del Receptor	$-7 \leq Vcm \leq +12$		± 200	mV
Receptor de modo común Rango de voltaje		-7	+12	V
Resistencia de entrada del receptor		12K		Ω

Fuente [25]

El estándar EIA/TIA-485 es técnicamente una especificación para controladores, receptores y transceptores conectados a una misma red, que describen parámetros de funcionamiento como las cargas unitarias, el accionamiento de salida, la corriente de cortocircuito y el voltaje de modo común. El controlador de la red tener capacidad de generar al menos 1,5 voltios diferencialmente con 60 ohmios de carga (dos terminadores de 120 ohmios en paralelo junto con 32 cargas de unidades) en un rango de voltaje de modo común de -7 a +12 V CC. (Candelas-Heredias, 2011).

3.2.2 Protocolos de comunicación.

Los protocolos de comunicación son un conjunto de reglas sobre los cuales se basa la conexión de dispositivos para la transmisión de datos (Candelas-Heredias, 2011). Definen

parámetros como la longitud máxima del canal de comunicación, la velocidad de transmisión de datos, longitud de la trama, entre otros. (Martinez, 2014)

La comunicación de dispositivos en un entorno industrial puede realizarse mediante el uso de protocolos. Los protocolos industriales consultados se muestran en la en la Tabla 7.

Tabla 7. Protocolos industriales de comunicación de dispositivos

Protocolo	**Arquitectura**	**Tasa de transmisión**	**Cantidad de dispositivos en la red**	**Aplicaciones**
PROFIBUS	Maestro - Esclavo	12 Mbps	32 maestros – 127 esclavos	Sistemas SCADA
ETHERNET	LAN 802.3	10 Mbps y 10 Gbps	100 a 1000 usuarios	Internet
MODBUS (serial)	Maestro - Esclavo	1200 bps a 256 Kbps	1 maestro y 32 esclavos	Sistemas de Monitoreo y control

Fuente [El Autor]

El protocolo de comunicación industrial MODBUS de acuerdo a la información consultada (ver Tabla 7) permite el control de una determinada cantidad de dispositivos lo que puede aprovecharse para la escalabilidad de la red. El diseño del sistema de monitoreo no contempla la transmisión de grandes cantidades de datos por lo que la tasa de transmisión del protocolo MODBUS se acopla fácilmente a las necesidades del proyecto. Como información adicional MODBUS es un protocolo de uso público y gratuito, fácil de implementar.

A continuación, en los siguientes apartados, se define el protocolo de comunicación MODBUS que se utiliza para el diseño del sistema de monitoreo de variables ambientales.

3.2.2.1 Protocolo MODBUS.

El protocolo de comunicación MODBUS basa su funcionamiento en la arquitectura maestro-esclavo a modo de solicitud-respuesta. Para la comunicación de dispositivos utiliza un bus central controlado por el dispositivo maestro que envía peticiones y los

dispositivos esclavos responden por el mismo bus de datos, dependiendo del modo de comunicación (*simplex, half –dúplex, full-dúplex*) que esté configurada la red. (Candelas-Heredias, 2011) (Soriano Miras, 2012)

Otra característica principal del protocolo MODBUS es, que en la red establecida los equipos esclavos no se comunican ni envían información entre sí y no transmiten información sin previa orden del equipo maestro. Así mismo los esclavos poseen una dirección única dentro de la red y el maestro no tiene ninguna. (Martinez, 2014)

- **Representación de datos en el Protocolo MODBUS.**

EL protocolo de comunicación MODBUS presenta tres formas de representación de datos para la transmisión dentro de la red, como son el MODBUS ASCII, el MODBUS RTU, y el MODBUS TCP (Candelas-Heredias, 2011). La decodificación de la información dependerá del formato empleado para la transmisión.

- **MODBUS RTU.**

En la representación de datos mediante *MODBUS RTU* (Unidad Terminal Remota) los bytes se envían en codificación binaria plana debido a que no se emplea ningún tipo de conversión (Martinez, 2014). El mayor beneficio de la representación *MODBUS RTU* es que aprovecha mejor el ancho de banda del canal, al aumentar la velocidad de transmisión de datos. Un inconveniente significativo se encuentra en la dificultad de identificar cuándo inicia o termina una trama de datos, aunque utiliza un tiempo de espera entre bytes recibidos para poder detectar las distintas tramas.

La conformación de la trama *MODBUS RTU* está delimitada por intervalos de tiempo de caracteres de silencio. Un caracter de silencio tiene una duración de un byte de datos enviado por el canal de comunicación, no transporta datos y su duración T (tiempo) depende de la velocidad V_t y del número de bits que se usen para su codificación N. De acuerdo con el protocolo *MODBUS*, para velocidades de hasta 19200 bps (baudios/segundo), el tiempo entre tramas debe ser como mínimo 3.5 veces la duración de un caracter, y para velocidades superiores se recomienda un tiempo fijo de 1.75 milisegundos. (Martinez, 2014)

El algoritmo definido en el protocolo *MODBUS RTU* incorpora un código *CRC* (Control de Redundancia Cíclica) de 16 bits (2 bytes) para detección de errores. El algoritmo es

aplicado por el emisor, calculado a partir de todos los bytes de la trama, antes de enviar. En el receptor se aplica el mismo algoritmo, calcula y comprueba el valor obtenido de la trama *(CRC)* para poder validar los datos, en caso contrario se ignora el mensaje. (Candelas-Heredias, 2011) (Martinez, 2014). En la Figura 14 podemos verificar la conformación de la trama *MODBUS RTU.*

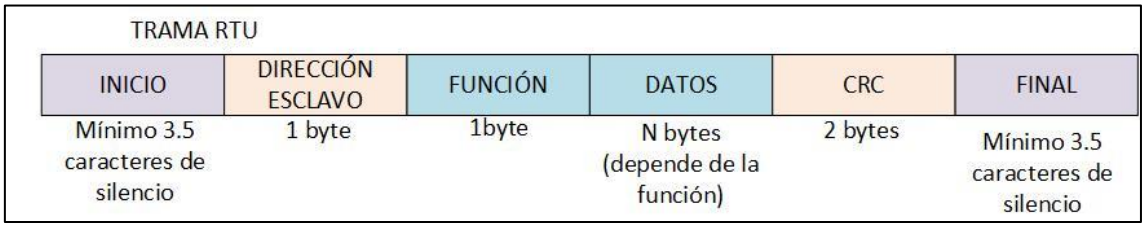


Figura 14. Estructura de Trama RTUI de MODBUS Serie. Fuente [6]

- **Capas de Aplicación en el modelo OSI del protocolo MODBUS.**

El protocolo *MODBUS* referido al modelo OSI interviene a nivel de aplicación y requiere de una pila de protocolos para el correcto funcionamiento en la red que se utiliza, dependiendo de la representación de datos. De esta manera en *MODBUS ASCII* y *MODBUS RTU* fueron desarrollados para aplicarse sobre un medio físico en serie asíncrono como el EIA/TIA-232, EIA/TIA-422, EIA/TIA-485. El *MODBUS TCP* se desarrolló para funcionar en redes que emplean arquitectura TCP/IP como Ethernet o Wi-Fi. (Gomez Blazquez, 2016) (Candelas-Heredias, 2011). En la imagen de la Figura 15 se puede observar las capas del modelo OSI en las que interviene *MODBUS* en sus diferentes modos de representación de datos.

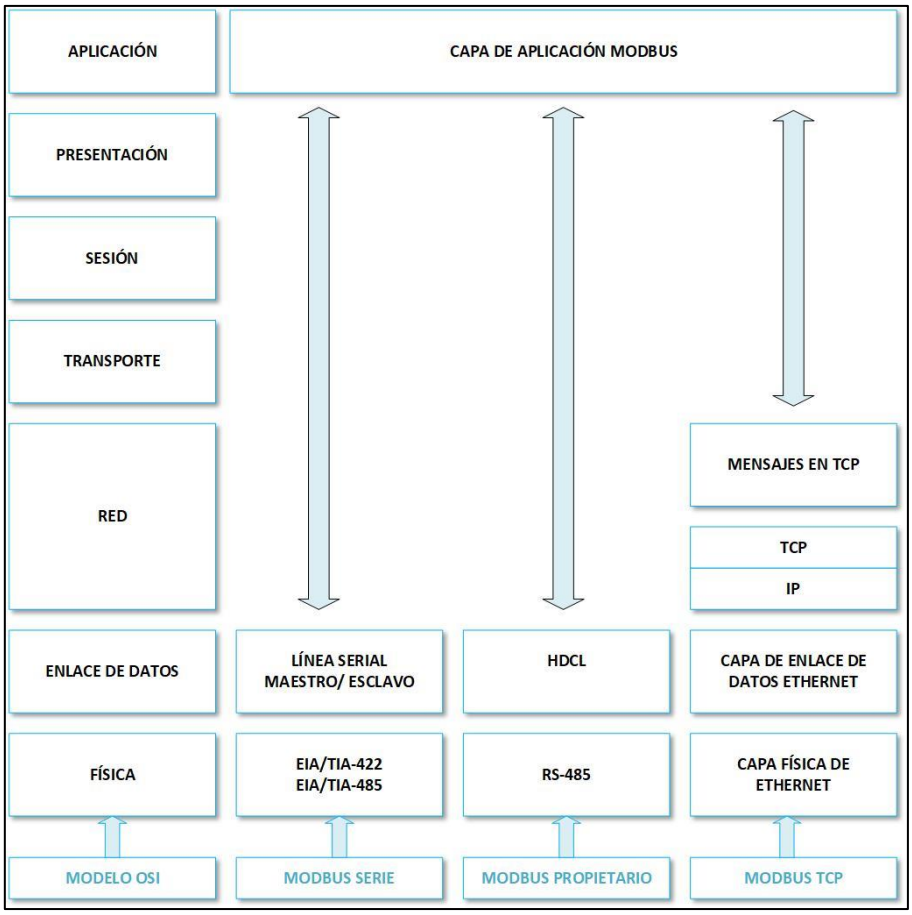


Figura 15. Capas de aplicación de MODBUS en modelo OSI. Fuente [23]

- **Esquema de conexión eléctrica de topología MODBUS con RS485.**

La aplicación del protocolo *MODBUS* sobre una línea de comunicación serial consiste en implementar físicamente dos cables con una interfaz eléctrica siguiendo las recomendaciones del estándar EIA/TIA-485. En esta estructura de red se requiere un dispositivo maestro para la sincronización de la transmisión de datos dentro del bus, los dispositivos remotos conectados a las mismas líneas seriales y además un tercer cable que actúe de conexión común con todos los nodos de la red (Martinez, 2014) (Samboya, 2012). En la Figura 16 se muestra el esquema de conexión de una red *MODBUS RTU* mediante RS485.

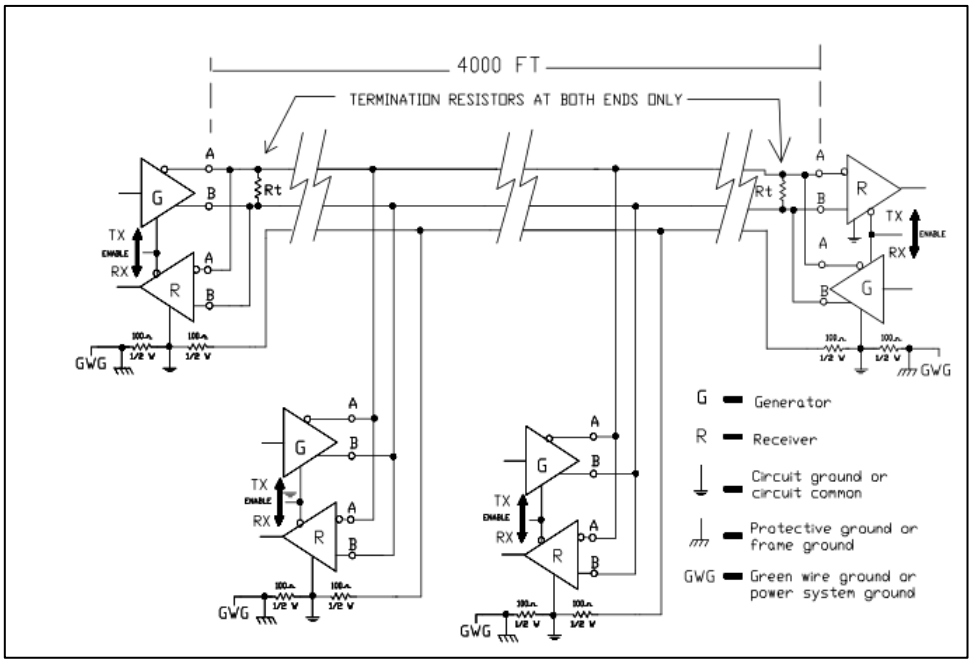


Figura 16. Protocolo MODBUS con RS485. Fuente [33]

- **Módulo MAX485 para Arduino**

El módulo MAX485 es un dispositivo desarrollado para operar con tarjetas Arduino mediante protocolo EIA/TIA-485-A (Liu, s.f.). Este módulo se utilizará para el diseño de la red alámbrica tipo *MODBUS* para facilitar el intercambio de datos de los dispositivos electrónicos que conforman la red.

Está compuesto principalmente por un microcontrolador MAX485 con todos los elementos electrónicos adicionales para su correcto funcionamiento de acuerdo con la norma EIA/TIA-485-A. En la imagen de la Figura 17 se presenta este dispositivo, mientras que la Tabla 8 se detalla las características principales del módulo MAX485.

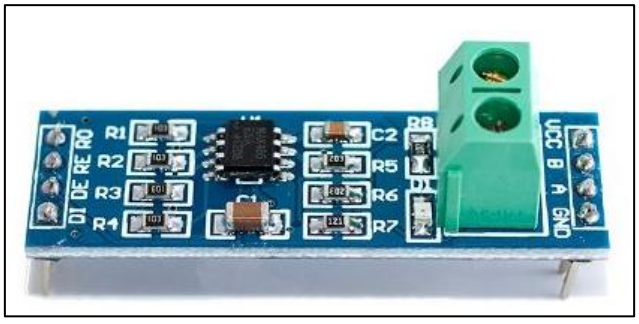


Figura 17. Módulo TTL a RS485 para Arduino. Fuente [30]

Tabla 8. Características principales de Modulo TTL 485.

Módulo MAX485 o Módulo TTL a RS485	
Descripción	**Unidades/Valores**
Voltaje de Operación	5 V $\pm$ 5%
Retardo de propagación	32 nanosegundos
Tasa de transmisión	100Kbps a 2.5 Mbps
Consumo de corriente promedio	Bajo300 uA.
Dispositivos por Bus	MODBUS hasta 32 dispositivos Transceiver
Estados de operación	Tres estados
Rango de entrada de voltaje en modo común.	-7V a +12V
Comunicación entre dispositivos	Simplex, Half-Dúplex, Full-Dúplex

Fuente [El Autor]

- **Funcionamiento del módulo MAX485.**

El módulo tiene pines que facilitan la conexión con el MAX485, así como con dispositivos externos. Los pines denominados como A y B (ver Figura 17) son las líneas diferenciales que permiten la transmisión de datos. La línea B es la negación en voltaje de A y ambas líneas tienen el mismo nivel de voltaje. El modo diferencial presente en las líneas A y B permite que las señales al ser transmitidas se "resten" y de esta forma se consigue eliminar al máximo el ruido que se adhiere en las líneas de transmisión. La comunicación con el microcontrolador se ejecuta de forma serial a través de los pines RX /TX de la tarjeta Arduino y los pines RO/DI del módulo MAX485. Como herramienta útil se puede emplear un pin digital de la tarjeta Arduino para especificar el tipo de operación que realice el módulo como enviar o recibir datos. (Texas Instruments, 2008). La imagen de la Figura 18 se describe los pines del Módulo MAX485 o TTL a RS485.

Se puede emplear el módulo MAX485 solo como transmisor o como receptor. La configuración como transmisor requiere que los pines RE y DE del módulo se conecten a 5V y para el intercambio de datos con la tarjeta Arduino se utiliza el pin TX conectado con el pin DI del módulo, el pin RO queda sin conexión. Para configurarlo como receptor los pines RE y DE se conectan a GND y los datos se intercambian por medio del pin RX de la tarjeta Arduino al pin RO del módulo, en esta configuración el pin DI queda desconectado. (Soriano Miras, 2012) (Liu, s.f.)

Módulo TTL a RS485			
No.	**Nombre**	**Funcion**	**Descripción**
1	RO	RECEPTOR DE SALIDA	Si A>B por 200mV, RO debe estar en alto; si A<B por 200 mV RO debe estar en bajo
2	RE	RECEPTOR DE SALIDA HABILITADO.	RO está habilitado cuando RE esta en bajo; RO tiene alta impedancia cuando RE esta en alto.
3	DE	DRIVER DE SALIDA HABILITADO.	Esta habilitado cuando DE esta en alto. Tiene una alta impedancia cuando DE está en bajo. Si las salidas del Driver están habilitadas, las partes funcionan como controladores de línea. Si bien son de alta impedancia funcionan como receptores de línea si RE está en bajo.
4	DI	DRIVER DE ENTRADA	Un estado en bajo en DI fuerza la salida A en bajo y la salida B en alto. Igualmente, un estado alto en DI fuerza una salida A en alto y salida de B en bajo.
5	GND	GROUND	Pin de conexión a tierra.
6	A	LÍNEA DIFERENCIAL DE DATOS A	Driver de salida y receptor de entrada diferencial
7	B	LÍNEA DIFERENCIAL DE DATOS B	Driver de salida y receptor de entrada diferencial
8	VCC	Voltaje	Conexión positiva de fuente entre 4.75 V y 5.25V

Figura 18. Descripción de funciones de pines del Módulo TTL a RS485. Fuente [El Autor]

Una tercera configuración se realiza con las salidas RE y DE del módulo conectado a un pin I/O de la tarjeta Arduino (cualquier pin I/O) además los pines RX y TX de la tarjeta Arduino se conectan a RO y DI del módulo respectivamente. De este modo el funcionamiento del módulo como transmisor o receptor dependerá del estado del pin digital de la tarjeta, esto es, que se controla RX y TX del módulo. Este modo de configuración se puede utilizar como tipo de comunicación Half Dúplex. (Soriano Miras, 2012)

3.3 Herramientas de Software para almacenamiento y gestión de la información.

3.3.1 Software libre.

El software libre hace referencia a cualquier herramienta informática puede ser usada, copiada, estudiada, modificada y distribuida libremente a través de Internet. (García, 2007)

El software es considerado como libre si dicha herramienta informática aplica tres libertades: libertad de ejecutar el programa con cualquier propósito, libertad de modificar el programa (acceso al código fuente), libertad de copiar el programa y libertad de realizar mejoras a dicho programa para el bien público. (García, 2007) (Celaya, 2007). A continuación, se describen algunos beneficios para el diseño del proyecto utilizando software libre:

- Se basa en el principio de colaboración comunitaria.
- No se requiere costos por licencias ni actualizaciones
- El soporte al software se encuentra en una amplia comunidad en internet.
- Son adaptables y configurables a las necesidades del usuario.

Los beneficios que ofrece el uso de software libre por sí mismo y la utilidad más notable para el desarrollo del diseño del proyecto en estudio, es el bajo costo en referencia al pago por licencia de uso y por lo tanto el presupuesto ahorrado puede ser invertido en otros gastos que genere el proyecto.

Por las razonas expuestas anteriormente, se pretende que el desarrollo el diseño del sistema de monitoreo considere emplear software libre como herramientas de apoyo para aprovechar de forma más eficiente los recursos de hardware, potenciar el aprendizaje y adaptar las herramientas informáticas para el entorno en el que se va a utilizar.

3.3.2 Software Python 3.

Python es un lenguaje de programación "interpretado", de alto nivel y multiplataforma que posee legibilidad de código de fácil entendimiento. Lenguaje interpretado indica que no es necesario compilar el código para su ejecución ya que existe un intérprete que lee el fichero fuente y lo ejecuta (Python Software Fundation, 2018). De igual manera la característica de lenguaje interpretado de Python, permite ejecutar el mismo código en distintos sistemas operativos, sin necesidad de cambiar el código fuente, para lo cual requiere de un intérprete, siendo por esta razón multiplataforma.

Python también es un software libre de código abierto, potente, flexible y con sintaxis clara y concisa. Es utilizado para el desarrollo de códigos para aplicaciones científicas, para creación de GUI (interfaz gráfica de usuario), para desarrollo de juegos y páginas web. (Python Software Fundation, 2018).

Python tiene una serie de complementos y librerías (Python Software Fundation, 2018) para potenciar el rendimiento de los sistemas y puede incluso llegar a ser un buen sustituto del software propietario como lo es MATLAB (*MATrix Laboratory* o "Laboratorio de Matrices"). La equivalencia de funcionamiento entre Python y MATLAB no es total y tiene brechas claramente definidas. Mientras en Python los complementos son desarrollados con carácter genérico, MATLAB desarrolla herramientas complementarias (*"toolbox"*) en función de aplicaciones específicas.

MATLAB es capaz de realizar un amplio abanico de procesos, sin embrago no se podría aprovechar todo el potencial de esta herramienta en el diseño del sistema de monitoreo y la licencia de uso tiene un costo elevado. Como principal opción entonces se puede utilizar Python para el desarrollo del diseño del sistema de monitoreo por las razones mostradas y además porque es una herramienta informática de software libre (ver apartado 3.3.1). (Python Software Fundation, 2018) (García, 2007)

El entorno de la programación de Python utiliza en la actualidad algún IDE (entorno desarrollado integrado) que facilitan la escritura del código fuente, así como la ejecución del script y detección de errores. En el siguiente apartado se describe el uso del *IDE PYCHARM*.

- **IDE PYCHARM.**

IDE o entorno desarrollado integrado, es un software que permite generar código de forma más entendible. *PyCharm* es un entorno desarrollado para algunas aplicaciones incluido el software Python, que cuenta con versión profesional de pago y versión *community* de acceso gratuito. La primera versión se enfoca en el desarrollo web y la segunda está orientada al desarrollo científico. Como principal novedad es que los programas desarrollados en extensión .py se ejecutan en un entorno virtual como proyecto. (JET BRAINS, 2018)

El uso del *IDE PyCharm* ofrece algunos beneficios importantes como identificar y corregir errores de sintaxis, depurador de código (*debugging*) y ejecución de script en un entorno virtual. Un inconveniente de *PyCharm* es debido a que muchas de las funciones avanzadas no están disponibles bajo la versión *community* (gratis), lo que significa que en algún momento se deberá pagar la licencia.

El desarrollo del diseño del sistema de monitoreo, se utiliza el *IDE PyCharm* para escribir el código fuente de Python que permite la lectura de datos por puerto serial y también configurar el uso de una librería que realiza la conexión entre Python y el gestor de base de datos PostgreSQL (explicada más detalladamente en el apartado 4.2.4).

3.3.3 Gestor de Base de datos

- **Definición de Base de datos.**

Un sistema de base de datos es básicamente, un sistema computarizado para llevar registros (Date, 2001). El objetivo general de los sistemas de bases de datos es almacenar información y permitir a los usuarios recuperar y actualizar esa información bajo demanda. (Capote & Ruiz, 2008) (Novella Latorre, 2012)

Los datos que almacena una base de datos son persistentes u operacionales lo cual indica que cuando se registran en el sistema no pueden ser removidos fácilmente sino sólo con permisos del administrador y operacionales porque se utilizan para múltiples procesos a petición del usuario (Abraham Silberschatz, 2002).

Los beneficios que proporciona una base de datos son importantes para el control y almacenamiento de grandes volúmenes de información, así como por la rapidez con la que se puede acceder a un registro determinado. Según (PostgreSQL, 2001-2018) (Capote & Ruiz, 2008), de una manera general se enlistan algunos beneficios:

- **Los datos pueden compartirse:** a través de aplicaciones existentes o mediante el desarrollo de aplicaciones que operen sobre los mismos datos.
- **Se puede reducir la redundancia:** por medio del control de varias copias de la misma información y eliminar estos datos si fuera necesario.
- **Evitar inconsistencia de información:** a través del control de redundancia y la propagación de actualizaciones.
- **Mantiene integridad de los datos y seguridad:** por medio de las restricciones de integridad aplicadas por el administrador de datos.
- **Cumple con estandarización:** al representar los datos de acuerdo a las normativas para lograr posibilitar el intercambio de datos o para el desarrollo entre los distintos sistemas.

- **Sistemas en modelo relacional.**

El modelo de datos relacional parte de la forma en cómo se presentan los datos al usuario y se interrelacionan entre sí aportando información adicional al dato almacenado. Los datos son percibidos por el usuario como tablas y la razón implícita en la definición es que se denominan relacionales por cuanto el término relación es básicamente el término matemático para tabla. (Capote & Ruiz, 2008)

- **Lenguajes de base de datos**

La gestión de un sistema de base de datos requiere de un lenguaje de definición de datos que permita especificar, entre otras características, el esquema de base de datos y a la vez posibilite la manipulación de los datos (consultas y modificaciones), siendo estas dos características partes de un único lenguaje de base de datos conocido como SQL. (Capote & Ruiz, 2008) (Abraham Silberschatz, 2002)

- **Estructura de un sistema de base de datos.**

En términos generales un sistema de base de datos se compone por dos grandes partes funcionales como el gestor de almacenamiento y el procesador de consultas. (Abraham Silberschatz, 2002) (Camargo Vega, 2015)

El gestor de almacenamiento es un bloque de software que permite una interfaz entre los datos de bajo nivel dentro de la base de datos y los programas de aplicación o consultas enviadas al sistema (Capote & Ruiz, 2008). En este bloque se almacenan, recuperan y actualizan los datos. Los componentes principales se denominan:

- Gestor de autorización de integridad.
- Gestor de transacciones.
- Gestor de archivos.
- Gestor de memoria intermedia. (Capote & Ruiz, 2008)

Así mismo el gestor de almacenamiento incorpora algunas otras estructuras de datos como parte de la implementación física del sistema, tales como: archivos de datos, diccionario de datos e índices.

El procesador de consultas permite implementar el intérprete del lenguaje de definición de datos y el compilador de manipulación de datos, esto se refiere a la interacción del sistema de base de datos con el usuario. (Date, 2001)

- **Arquitectura de aplicaciones.**

Se puede determinar la arquitectura de una base de datos de acuerdo a las capas que las componen. Así en el conjunto del sistema, se puede distinguir, entre máquinas, un cliente que es donde operan los usuarios remotos de la base de datos y un servidor donde se ejecuta el sistema de base de datos. (Abraham Silberschatz, 2002)

La imagen de la Figura 19 describe el modelo cliente-servidor, nombrado en el párrafo anterior.

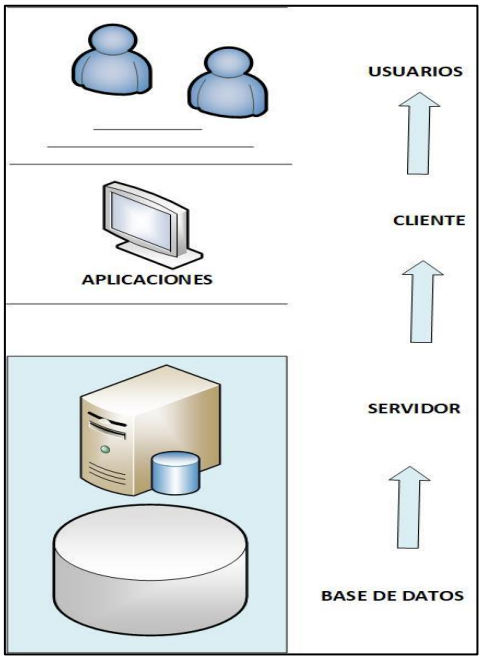


Figura 19. Arquitectura cliente-servidor. Fuente [1]

3.3.3.1 Criterios de selección del gestor de base de datos para el sistema de monitoreo.

El almacenamiento de datos para el sistema de monitoreo contempla el uso de un Gestor de Base de Datos de código abierto. Dado esta orientación se limita la búsqueda entre dos gestores de base de datos como PostgreSQL y MySQL. Las características generales, las facultades para almacenar datos, lenguajes soportados y aplicaciones de PostgreSQL y MySQL se muestran en la Tabla 9.

Tabla 9. Características de PostgreSQL y MySQL.

Criterio	**PostgreSQL**	**MySQL**
Características generales	• Software libre • Código abierto • Baja velocidad de respuesta y mayor consumo de recursos • Fácil instalación • Flexible y escalable	• Software Propietario • Código abierto. • Mayor velocidad de respuesta y bajo consumo de recursos • Fácil instalación • Flexible y escalable
Límite de almacenamiento de datos	Base de datos de hasta 50 millones de registros	Ilimitado
Lenguajes de programación soportados	Python, C, C++, Java, JavaScript, entre otros	Python, C, C++, Java, PHP, R, entre otros.
Aplicaciones	Manejo de grandes volúmenes de datos (Big Data), análisis de datos	Aplicaciones web, y procesamiento en línea

Fuente [El Autor]

Según la información consultada en la Tabla 9, aunque PostgreSQL consume mayor cantidad de recursos especialmente de memoria, para bases de datos grandes, la velocidad de respuesta es eficiente respecto a otro sistema de bases de datos como MySQL (de código abierto también) (PostgreSQL, 2001-2018)(Chambi, 2006). Es por esta razón que se utiliza PostgreSQL, debido a que el monitoreo en tiempo real incrementará constantemente datos, en la base de datos creada para tal propósito.

El gestor de base de datos que se utilizará en el desarrollo del diseño del sistema de monitoreo es PostgreSQL, por lo que en los apartados siguientes se procederá a describir el funcionamiento.

3.3.3.2 PostgreSQL.

PostgreSQL es un gestor de base de datos, objeto-relacional que utiliza una arquitectura cliente/servidor y emplea multiprocesos para garantizar la estabilidad del sistema. Es un

tipo de sistema orientado a objetos, libre, de código abierto y fácil instalación (PostgreSQL, 2001-2018) (Kasián, 2012).

Entre las características principales de PostgreSQL, se enumeran las siguientes:

- Sistema de gestor de base de datos 100% ACID (permite realizar transacciones con consistencia, aislamiento y durabilidad).
- Sistema distribuido bajo licencia BSD y código fuente disponible libremente.
- Soporta multiplataforma Linux, Unix, Windows entre otras.
- Acceso encriptado vía SSL.
- Múltiples métodos de autenticación. (Novella Latorre, 2012)

Las características de Programación/ Desarrollo de PostgreSQL son:

- Procedimientos almacenados en varios lenguajes de programación como: PL/pgSQL, PL/Perl, PL/Python.
- Soporta el almacenamiento de objetos binarios grandes (gráficos, video, sonido, entre otros).
- APIs para programar en C/C++, Java, .Net, Perl, Python, y algunos otros similares. (Novella Latorre, 2012)

En cuanto a las características SQL tenemos:

- Claves primarias (PK) y claves ajenas (FK).
- Índices
- Subconsultas.
- Triggers.
- Columnas auto-incrementales.
- Herencia de tablas. (Novella Latorre, 2012)

Limitaciones de almacenamiento del gestor de base datos PostgreSQL:

- Tamaño máximo de base de datos: ilimitado
- Tamaño máximo de tabla hasta 32 TB.
- Tamaño máximo de registro hasta 16 TB.
- Tamaño máximo de campo 1 GB.

- Máximos registros por tabla: ilimitado. (Capote & Ruiz, 2008)

La administración de PostgreSQL utiliza a menudo una herramienta denominada *PgAdmin* que funciona en la mayoría de los sistemas operativos (como Windows, Linux, Mac OS, entre otros). Es una interfaz gráfica compatible con todas las características de PostgreSQL y facilita la administración.

Los beneficios que presenta PostgreSQL para el desarrollo del sistema de monitoreo, son entre otras:

- Escalabilidad y gran cantidad de memoria del sistema que le permite soportar mayor cantidad de peticiones simultáneas.
- Fácil de administrar y flexibilidad para desarrollo de investigaciones sin generar costos adicionales.
- Capacidad de comprobar la integridad referencial y almacenar procedimientos en la propia base de datos.
- Tiene sintaxis SQL estándar y de fácil interpretación.
- Soporte empresarial disponible en modalidad de pago. (Espinoza, 2012)

Los inconvenientes principales de PostgreSQL son debido a:

- Características de procesamiento (más lento).
- Mayor consumo de recursos.
- En la modalidad de código abierto no existe soporte técnico obligatorio. (Kasián, 2012)

El estudio del diseño de monitoreo se estima que la información será almacenada en la base de datos PostgreSQL obtenida por puerto serial; para ello se establecerá un interfaz conector entre el puerto serial y el gestor de base de datos mediante el desarrollo de programación en Python.

3.3.4 Software de Big Data para procesar, almacenar y visualizar datos en la web.

Los datos que provienen de sensores o de cualquier otro dispositivo que se conecte a internet crecen de forma exponencial. Almacenarlos en un gestor de base datos no es suficiente puesto que los propios datos generan información adicional (metadatos). (Santacreu Grifol, 2016)

Las soluciones que se encargan del manejo de esta inmensa cantidad de información se denomina Big Data. Las soluciones Big Data fueron desarrolladas con el propósito de almacenar, procesar, obtener valor analítico y visualizar el conocimiento generado a partir de los datos capturados por sensores o cualquier otro dispositivo conectado a internet. (Academy, 2018) (Santacreu Grifol, 2016)

Muchos softwares han sido desarrollados para soluciones Big Data. Escoger la herramienta correcta que permita la indización de los datos (buscar información por índices) conservando el valor original del dato no es una tarea sencilla ya que de esta decisión dependerá el éxito o fracaso del proyecto.

Sin embargo, se toma como referencia el estudio realizado por (Santacreu Grifol, 2016) donde se analizan dos herramientas de Big Data: una de código abierto como la Pila Elastic y otra herramienta informática de pago como Splunk muy populares en la actualidad. Esta información se presenta en la Tabla 10.

Tabla 10. Comparación de herramientas de Big Data

Criterio	**Pila Elastic (ELK)**	**Splunk**	**Hadoop**
Arquitectura del Software	Código abierto	Arquitectura de Pago.	Código abierto
Componentes	Conjunto de herramientas: Logstash, Elasticsearch, Kibana.	Arquitecturas válidas de Splunk (SVA)	Sistema de archivos Distribuidos Hadoop HSDF
Recomendaciones de hardware	Especificaciones mínimas para instalación y funcionamiento: disco duro local, procesador, tarjeta gráfica.	Especificaciones mínimas para instalación y funcionamiento: disco duro local, procesador, tarjeta gráfica.	Especificaciones mínimas para instalación y funcionamiento: disco duro local, procesador, tarjeta gráfica.

Soporte	Foros y comunidades	Personalizadas y específicas.	Comunidades y Foros
Tamaño de indización de datos	Menos de 100 Mb de datos diarios	Mayor a 500 Mb de datos diarios	No especificado depende de otras herramientas
Conocimientos para manipulación de software	Conocimientos en java para adaptar a las necesidades del proyecto	Conocimiento asumido por el propietario del software debido a que se aplica a una solución específica.	Conocimientos en java para adaptar a las necesidades del proyecto

Fuente [El Autor]

Splunk es la herramienta de Big Data ideal para el funcionamiento del sistema de monitoreo. Las características de software de pago hacen que esta herramienta se pueda adaptar a las necesidades del proyecto en estudio. La robustez, rendimiento y la solución específica están garantizadas.

No obstante, el motivo principal para utilizar la pila Elastic para el diseño del sistema de monitoreo en lugar de Splunk, es el presupuesto del proyecto. La pila Elastic si bien es cierto requiere de alto costo de aprendizaje, no se realiza pago alguno por licencia de uso; además con la debida configuración de la pila Elasctic, se puede adaptar bien a las necesidades del sistema de monitoreo.

La gestión de los datos mediante la pila Elastic tendrán un fin simple: presentación y monitoreo en panel informativo en la web. En los siguientes apartados se describen las herramientas informáticas que componen la pila Eastic.

3.3.4.1 La pila Elastic.

- **Definición**

La pila Elastic, se conoce generalmente como *Elastic Stack* y es un conjunto de herramientas de código abierto, que permiten indizar volúmenes de datos de cualquier fuente (cualquier base de datos), cualquier formato (tabla relacional o no relacional) y

buscar, analizar y visualizar en tiempo real (Elasticsearch , 2018). Las herramientas que conforman el *Elastic Stack* se denominan *Logstash*, *Elasticsearch* y *Kibana*.

El desarrollo del diseño del sistema de monitoreo empleará tres de las herramientas del conjunto ELK como *Logstash*, *Elasticsearch*, y *Kibana* para gestionar, almacenar y representar los datos en tiempo real. En los siguientes apartados se describen estas herramientas. En la imagen de la Figura 20 se muestra un esquema de la integración de las herramientas de la pila Elastic.

Figura 20: Integración de las herramientas de la pila Elastic. Fuente [15]

- **Logstash.**

Logstash es una herramienta (de código abierto) para la canalización-recopilación de datos, rendimiento y transporte flexible. *Logstash* está diseñado para procesar con eficiencia una lista creciente de registros, eventos, etc., de fuentes de datos no estructurados para su distribución a una variedad de sistemas, incluyendo *Elasticsearch*. (Prado Ventura, 2016)(Elasticsearch , 2018).

Básicamente, *Logstash*, se puede considerar como un sistema de procesos encadenados que tiene un flujo de datos que implica que, la salida de una fase es una entrada de otra (del término *pipeline*). Su estructura está configurada con una entrada (por donde ingresa la información desde ficheros, puertos o base de datos), filtro (selección de información para definir, analizar o alterar) y salida. (Prado Ventura, 2016)

Se utilizará *Logstash* en el desarrollo del sistema de monitoreo, debido a que posibilita la obtención de la información de los procesos que se realizan en la pila Elastic, como la conversión de formatos de entrada en otro que acepte la herramienta *Elasticserach*, la conversión de los formatos de salida y los filtros que se emplean para procesar eventos. Esta información es también útil para la detección de fallas del sistema (pila Elastic). (Elasticsearch , 2018)

- **Elasticsearch.**

Elasticsearch es una herramienta que funciona como motor de búsqueda y análisis, de código abierto, diseñado para escalabilidad horizontal, confiabilidad y administración de la pila Elastic. Combina la velocidad de búsqueda con el poder de la analítica a través de un lenguaje de consulta sofisticado y amigable con el desarrollador, y además cubre datos estructurados y no estructurados, así como series de tiempo. (Elasticsearch , 2018)

El servidor de búsqueda *Elasticsearch* está basado en *Apache Lucene* (API de código abierto) que permite indexar datos y tiene la capacidad de comunicarse con otros sistemas por medio de su interfaz web denominado *RESTful*. (Prado Ventura, 2016)

La información que recopila Elasticsearch la guarda en documentos con formato. json que le dan características de una base de datos no-SQL (base de datos no relacional) lo cual es útil para la gestión y el diseño de grandes conjuntos de datos distribuidos, y el indexado es utilizado por los motores de búsqueda en internet, lo cual presenta también un beneficio para el desarrollo del diseño del sistema de monitoreo.

La herramienta *Elasticsearch* se utilizará en el sistema de monitoreo para el almacenamiento de los datos con las características descritas anteriormente y la información procesada será enviada a la herramienta *Kibana* de la pila Elastic.

- **Kibana.**

Kibana es una plataforma de visualización de información, de código abierto, que permite interactuar con los datos almacenados en la pila Elastic, a través de gráficos de alta definición. *Kibana* se puede modelar la forma de representación de los datos almacenados con elementos visuales combinados en paneles personalizados y permiten también compartir información (a partir de sus datos), en cualquier equipo terminal conectado a internet. (Elasticsearch , 2018)

La herramienta *Kibana* facilita la visualización y el manejo de toda la información que se encuentra almacenada en *Elasticsearch*, permite la configuración de uno o varios *dashboard* (representación gráfica de principales métricas) con la información útil para el usuario, aplicación de filtros para búsqueda, presentación de datos, así como facultar la exportación de resultados. (Elasticsearch , 2018)

La herramienta *Kibana* se aplicará en el sistema de monitoreo para el diseño del interfaz web. La información se detalla en el *dashboard* con modelamientos gráficos entendibles para el usuario y parámetros óptimos para la visualización, que facilitan el monitoreo de las métricas de las variables ambientales capturadas por los sensores y presentadas en la web, en tiempo real.

4. MATERIALES Y MÉTODOS.

4.1 Materiales

En el tiempo que se desarrolló el diseño del sistema de monitoreo se utilizaron los materiales necesarios con un costo monetario, como se describe en la Tabla 11:

Tabla 11: Materiales y Presupuesto

Material	Unidades	Precio de Unidad $	Costo Total $
Arduino Mega2560	1	20	20
Arduino Nano	3	10	30
Módulo Max485	4	5	20
Sensor DHT22	3	10	30
Sensor BH1750	3	6	18
Regulador de Voltaje 5V	4	1.2	4.8
Capacitores	8	0.10	0.8
Resistores	4	0.05	0.2
Led de alto brillo	4	0.15	0.6
Borneras para PCB	16	0.25	4
Baquelita 15x30 cm	1	2	2
Caja Negra	4	5	20
Cable UTP Cat. 5e x 76 metros	1	0.45	34.2
Conectores RJ-45	6	0.5	3
Conectores Jack RJ-45 PCB	6	1.5	9
Fuente Externa 110VAC/12VCC	4	15	60
Cables Varios	1	5	5
		Total	261.6

Fuente. [El Autor]

4.2 Metodología.

La metodología que se emplea para el desarrollo del sistema propuesto consta de diversos parámetros para los cuales se utilizan varios métodos como:

Método Investigativo: se emplea este método para obtener la información acerca del hardware y software adecuado para el diseño del sistema, que proporcionen los beneficios necesarios y posibiliten el cumplimiento de los objetivos planteados.

Método Deductivo: el empleo de este método se lo realiza con el fin de determinar los dispositivos electrónicos óptimos para la obtención de las variables a analizar; el software adecuado para procesar los datos y el medio de transporte de los mismos.

Método para el diseño Web: se emplea esta metodología para definir un diseño sistemático con un enfoque dirigido a la presentación de la información, que posibilita el monitoreo, la reacción frente a los datos obtenidos e interpretación espontánea por parte de los usuarios.

4.2.1 Desarrollo Eléctrico.

La información que se obtuvo a partir de la investigación sobre sensores y tarjetas útiles para el sistema, permitieron elegir una lista de dispositivos adecuados para la adquisición y transmisión de los datos en la red de comunicación MODBUS RS-485.

A continuación, se describen los diferentes componentes empleados en el diseño y construcción del sistema.

Arduino Mega: este componente se utilizó como nodo Maestro que sincroniza el funcionamiento de la red alámbrica MODBUS.

MAX485: se empleó para la transmisión de datos a largas distancias. Este dispositivo transforma los niveles de transmisión a niveles de TTL. La configuración de Modo de envío o recepción fue fijada para que sea controlada por un pin digital de la tarjeta Arduino. El esquema de conexión del MAX485 a la tarjeta Arduino Mega se muestra en la imagen de la Figura 21.

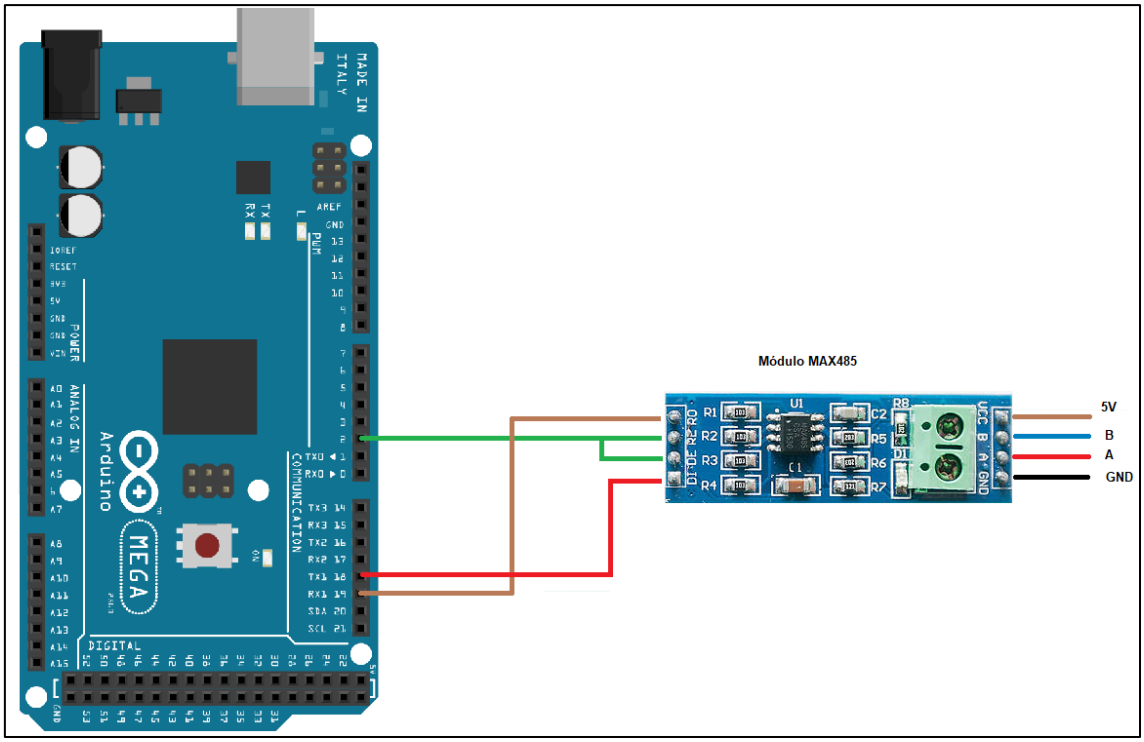


Figura 21. Conexión de Módulo TTL con tarjeta Arduino Mega. Fuente [El Autor]

La comunicación entre los dispositivos por *MODBUS RS485* requiere, que el envío y recepción de datos se realice uno a la vez, cuando el bus de datos se encuentre disponible. Los dispositivos deben ser configurados en modo *half-dúplex* para cumplir este requerimiento. La conexión para el modo *half-dúplex* consiste en unir los pines A de los módulos MAX485 entre sí, y de igual manera los pines B. Los pines DE y RE del módulo MAX485 se conectan al mismo pin digital de la tarjeta Arduino Mega (pin 2 para el sistema propuesto) y el modo de comunicación del bus de datos dependerá de la salida (HIG o LOW) a la que se encuentre el pin digital. En la imagen de la Figura 22 se muestra el esquema de conexión del bus de datos con MAX485.

Arduino Nano: se utilizó el Arduino Nano para la recepción de los datos captados por los sensores de las variables ambientales a medir. Además, se utilizó como parte central del Nodo Remoto (Dispositivo esclavo de la red).

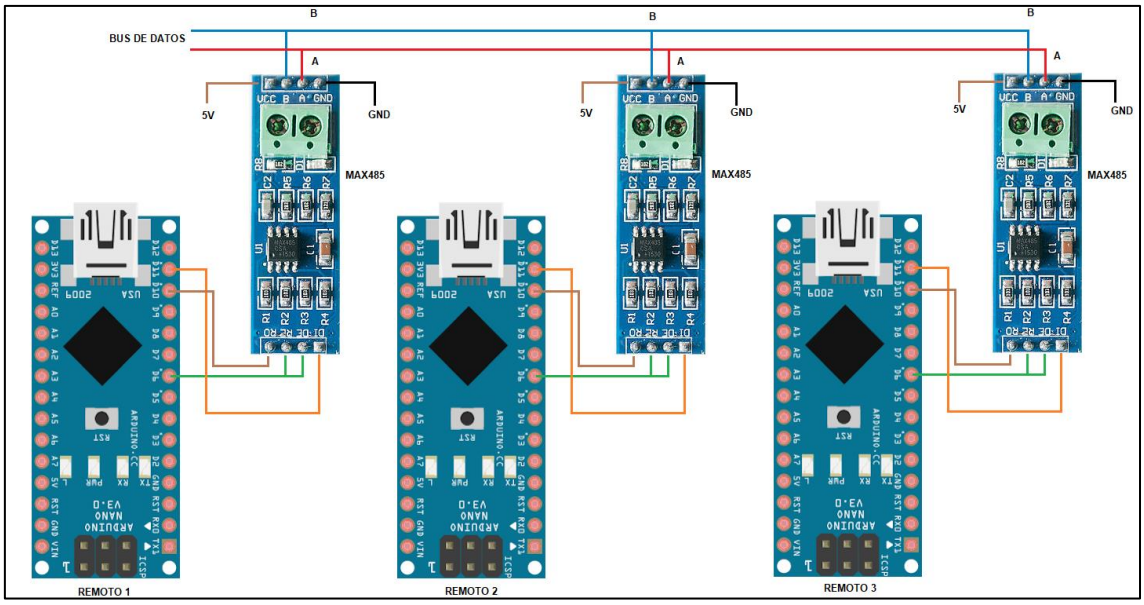


Figura 22. Conexión de Bus de datos con MAX485. Fuente [El Autor]

Sensor DHT22: se empleó para la obtención de la señal digital sobre condiciones de temperatura y humedad relativa en las áreas de interés. El dispositivo fue conectado al Arduino Nano y forma parte de los nodos remotos. En la imagen de la Figura 23 se puede ver la conexión de DHT22 con tarjeta Arduino Nano.

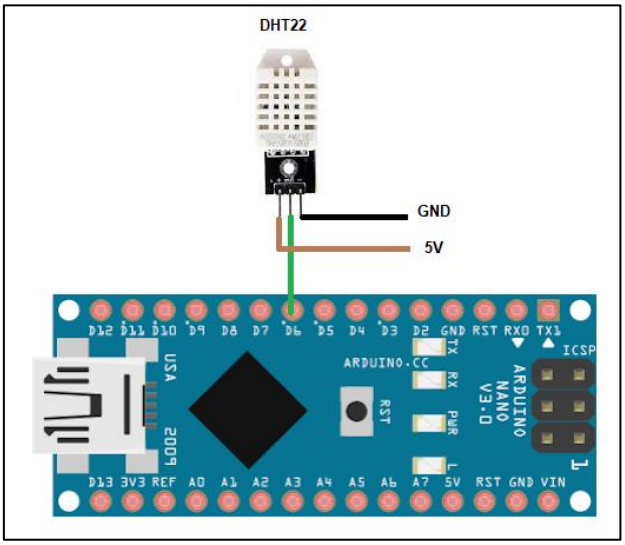


Figura 23. Conexión de DHT22 con tarjeta Arduino Nano. Fuente [El Autor]

Sensor BH1750: se utilizó para la obtención de la cantidad de iluminación en las áreas críticas del edificio SIS ECU-911 Loja, también forma parte de los nodos remotos. En la imagen de la Figura 24 se puede observar el esquema de conexión de BH1750 con la tarjeta Arduino Nano.

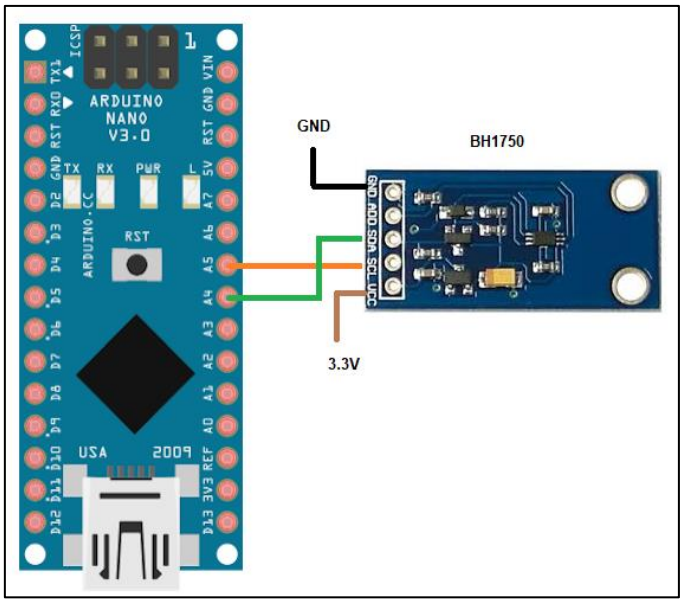


Figura 24. Conexión de BH1750 con tarjeta Arduino Nano. Fuente [El Autor]

Regulador de Voltaje 7805: se utilizó un regulador de voltaje 7805 para alimentar los dispositivos DHT22 y MAX485 que conforman los nodos remotos. El voltaje de entrada de 12V proporciona un cargador eléctrico conectado a la red de alimentación de energía del edificio. En la imagen de la Figura 25 se presenta el esquema de conexión del regulador.

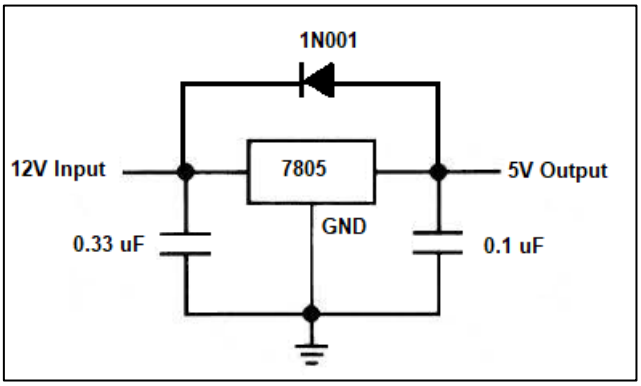


Figura 25. Regulador de voltaje 7805. Fuente [El Autor].

Los planos de conexión eléctrica de los dispositivos que forman parte de los módulos diseñados para el sistema de monitoreo, así como el esquema de conexión de la red alámbrica se detallan en el apartado 11.1. En la imagen de la Figura 26 se muestra un esquema de conexión de componentes de los dispositivos que conforman la red del sistema de monitoreo.

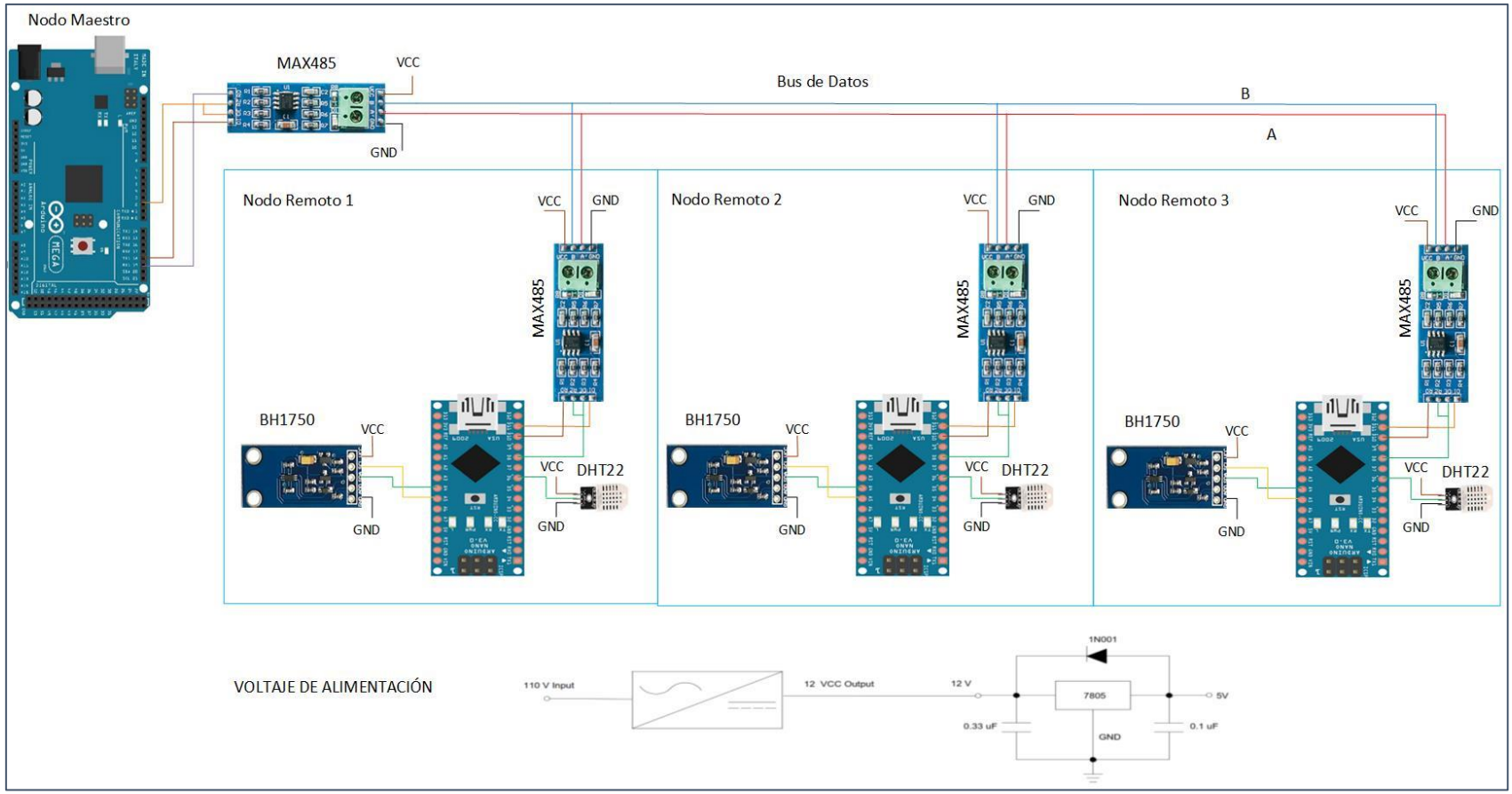


Figura 26. Esquema de conexión del Nodo Maestro y los Nodos Remotos. Fuente: [El Autor]

4.2.1.1 Diseño de PCB.

Se utilizó las herramientas que posee el entorno PROTEUS 8.6 para seleccionar los componentes electrónicos que forman los circuitos básicos de los módulos del sistema de monitoreo y realizar las conexiones eléctricas que resultan en un conjunto de pistas por dónde se transmiten señales. El ruteado de pistas y el diseño del esquema final del circuito se realizaron mediante el entorno ARES PROTEUS, del mismo software.

Los diseños esquemáticos finales se imprimen teniendo en cuenta todos los procesos a emplear tanto en la forma de imagen (por ejemplo, impresión en espejo) y el tipo de material utilizado (papel de transferencia) para facilitar la etapa de construcción de las PCB. Los pasos a seguir para la fabricación de PCB se describen a continuación.

- **Proceso de fabricación de las PCB.**

Para la fabricación de las PCB, se procedió de la siguiente manera:

- **Preparación de placa baquelita:** se recortó el tamaño de la paca de acuerdo con el diseño del circuito. Posteriormente se limpió la superficie con alcohol y acetona para eliminar residuos de tinta o grasa.
- **Transferencia térmica:** es el proceso que se utilizó para transferir los trazos diseñados por software hacia la baquelita. El método más simple se hizo a través del planchado por un lapso de 10 minutos. Posteriormente se procedió a poner la placa en agua fría para eliminar los restos de papel de la baquelita y otras impurezas.
- **Eliminación de cobre restante:** se sometió la placa a una combinación de ácido férrico y agua para eliminar el cobre restante. El proceso es puramente químico. Dependiendo de la concentración del ácido puede tardar entre 10 y 30 min.
- **Limpieza de placa y comprobación de pistas:** después de someter a la placa de baquelita al proceso químico, se lavó con abundante agua, secó y luego se realizó una prueba de campo de continuidad de pistas.
- **Perforado y soldado de componentes:** Se perforaron todos los puntos de conexión necesarios donde fueron soldados los componentes correspondientes.

Las gráficas de la Figura 27 y Figura 28 presentan el diseño de conexión de las PCB, mientras que en el apartado 11.4 se pueden ver los diseños esquemáticos para mejor comprensión.

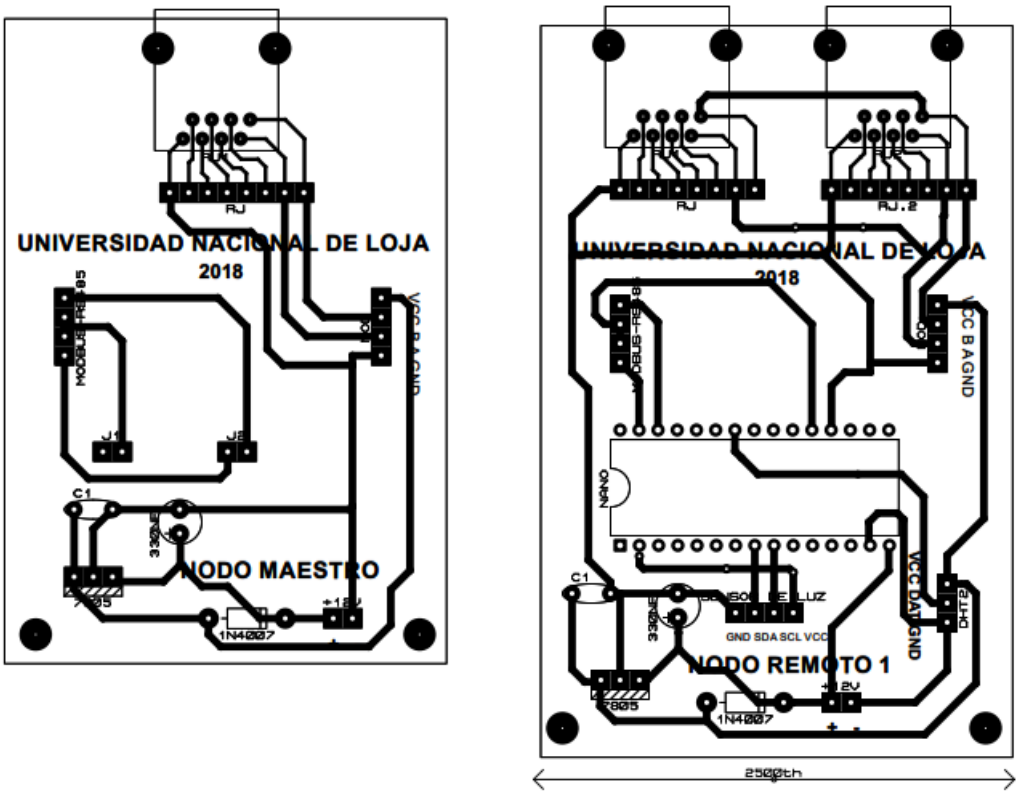


Figura 27. Diseño de circuito para PCB de Nodo Maestro y Nodo Remoto 1. Fuente [El Autor]

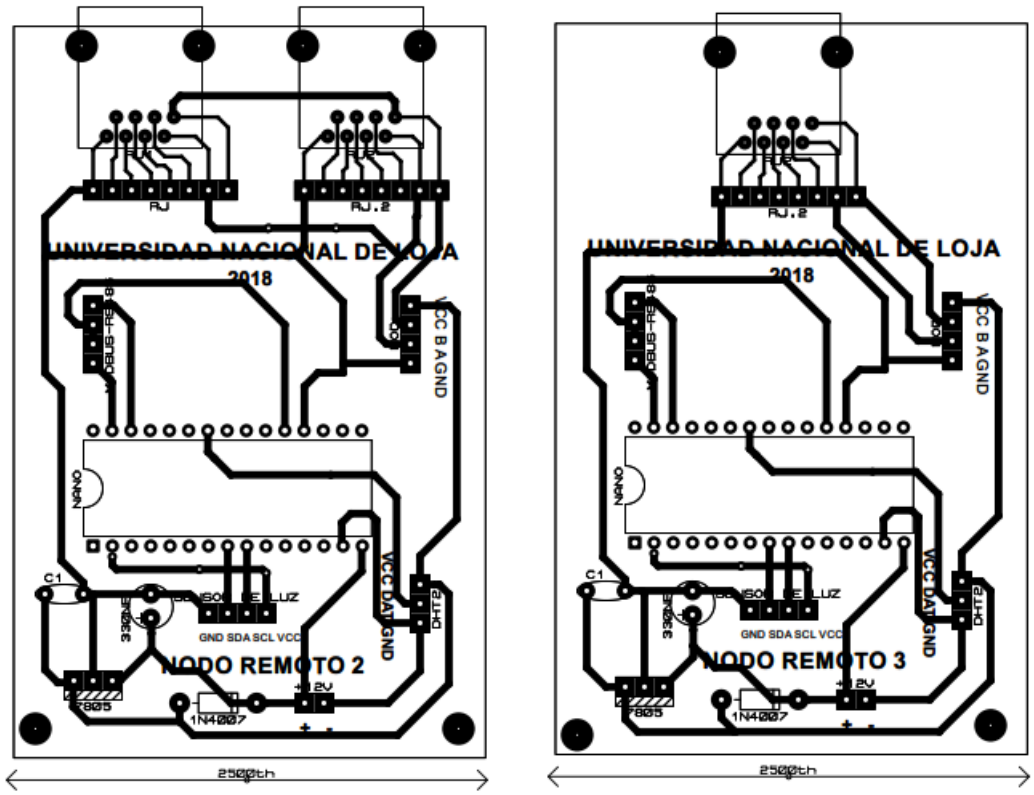


Figura 28. Diseño de circuito para PCB Nodo Remoto 2 y Nodo Remoto 3. Fuente [El Autor]

4.2.2 Adquisición de datos.

Las señales de las variables ambientales de las áreas de interés, respecto a la temperatura, humedad e iluminación que proporcionan información del entorno, fueron captadas por los sensores DHT22 y BH1750.

Los sensores fueron conectados a las tarjetas Arduino nano y forman parte de los nodos remotos. Los datos que los sensores proporcionan son enviados a las tarjetas Arduino donde se almacenan de forma temporal y se disponen para ser agrupados en tramas para el envío en el momento que se requiera.

4.2.2.1 Adquisición de los datos de temperatura y humedad mediante sensores DHT22.

La programación de los sensores de temperatura y humedad relativa DHT22 permite obtener las lecturas de las señales de las variables ambientales como datos digitales. El script desarrollado en el entorno de Arduino, se ejecuta de acuerdo al diagrama de flujo mostrado en la imagen de la Figura 29.

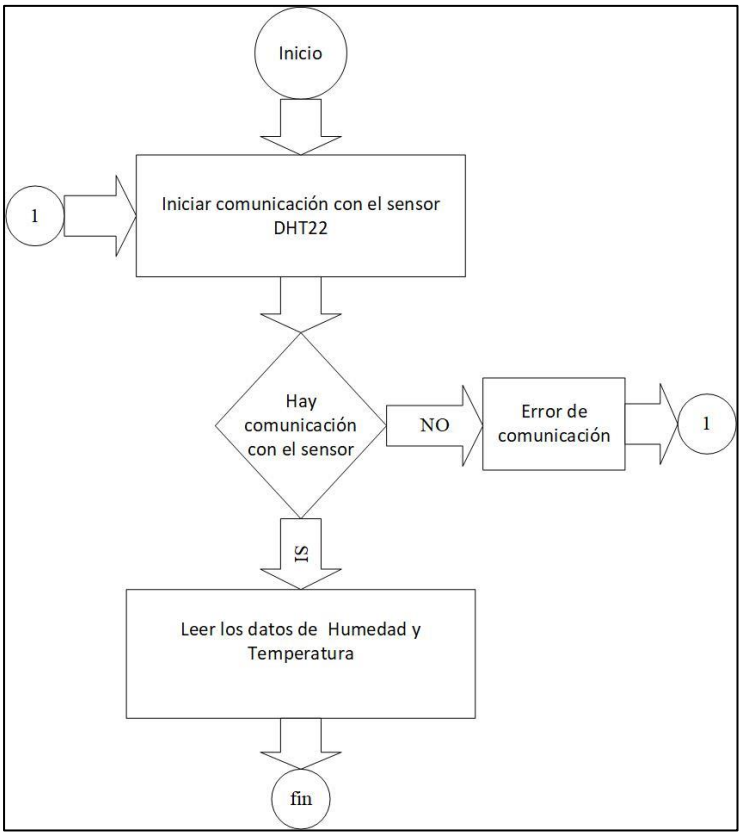


Figura 29. Diagrama de flujo de programación de sensor DHT22. Fuente [EL Autor]

El funcionamiento del sensor con la tarjeta Arduino nano utiliza la librería *DHT.h* para el envío de los datos capturados, la secuencia de comandos utilizados tiene la siguiente sintaxis:

```
#include "DHT.h" //librería para sensores humedad y temperatura
//Pines correspondientes para variables de humedad y temperatura
DHT dht1(6, DHT22);

//se inicializa los sensores de humedad y temperatura
void setup() {
dht1.begin();
}
//Leemos los sensores de humedad, temperatura
void loop() {
          float h1 = dht1.readHumidity(); //variable humedad
          float t1 = dht1.readTemperature();//variable temperatura
//("envío de error si existe pérdida de datos del SENSOR 1 DHT22");
           if (isnan(h1) || isnan(t1)) {
```

```
            h1=-1.0;//error obteniendo datos de humedad
            t1=-1.0;//error obteniendo datos de temperatura
            }
```

En esta secuencia de código muestra los comandos para el sensor DHT22 del Nodo Remoto 1 la programación completa se puede consultar en el apartado 11.2.2

4.2.2.2 Adquisición de los datos de iluminación mediante sensores BH1750.

Los sensores BHT1750 se utilizan para la obtención de la medida de la variable de iluminación de las áreas de interés. El proceso para obtener la señal por parte del sensor y el envío hacia la tarjeta Arduino nano sigue el flujo mostrado en la imagen de la Figura 30

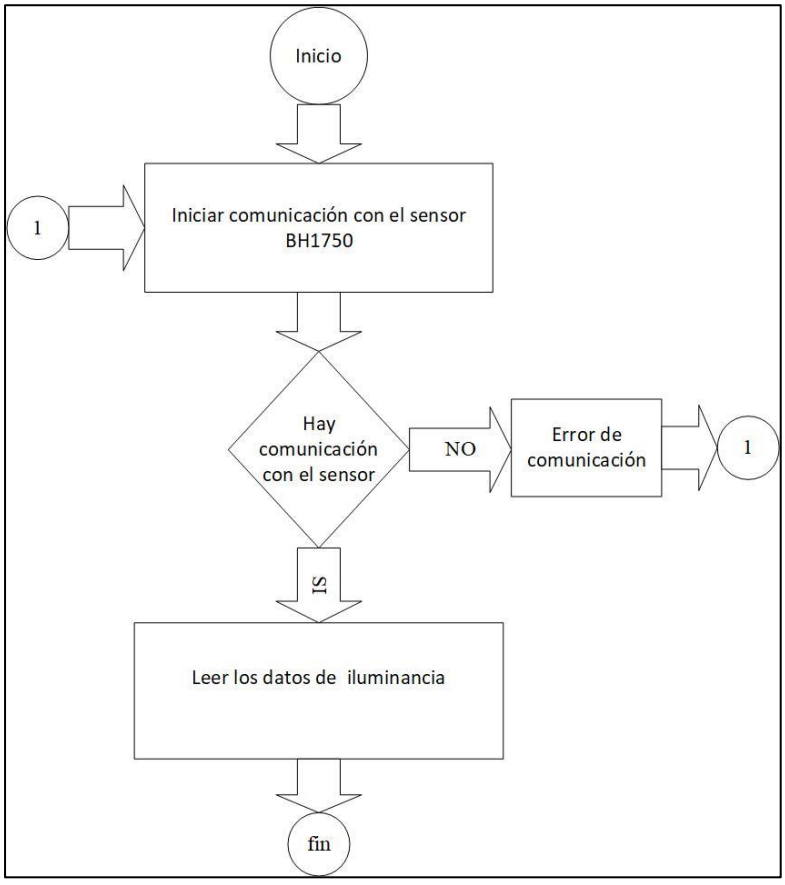


Figura 30. Diagrama de flujo de programación de sensor BH1750. Fuente [EL Autor]

Se utilizó la librería *BHT1750.h* para la programación del intercambio de información entre la tarjeta nano y el sensor BH1750. Los comandos escritos, en líneas generales, son los que se muestran a continuación:

```
#include <BH1750.h>//libreria para sensor de luz
#include <Wire.h>//bus de comunicacion de sensores de luz con la nano

void setup()
{
//Inicilizar el bus I2C, la biblioteca BH1750
Wire.begin();
}
void loop(){
          //Llamar a funcion de sensor de iluminación
           read_sensor1();
          //Leer los datos del sensor de  iluminación
          int lectura = sensor1_result;
          }
```

Las medidas de las señales de iluminación que captura el sensor, las envía a la tarjeta Arduino nano como datos digitales mediante el protocolo de comunicación I2C. El modo de comunicación I2C requiere de líneas de control para el envío de datos, razón por la cual el script tiene un bloque de instrucciones que manipula el intercambio de información desde BH1750 a la tarjeta Arduino nano. Las líneas de comando utilizadas son las siguientes:

```
 //Direccion de  sensor BH1750
byte u8_BH1750_address = 0x23; //Pin ADDR en nível lógico 0
//Instrucciones de modo lectura
byte mode_HighResolution_1 = 0x10;
//variables para sensores de luminancia
uint16_t sensor1_result = 0;

//Funciones para sensores de luminancia
void read_sensor1(void){
   //Inicia a comunicacion I2C en la direccion selecionada
```

```
  Wire.beginTransmission(u8_BH1750_address);
  //Configuracion del sensor para el modo de operacion
  Wire.write(mode_HighResolution_1);
  //Finaliza la comunicacion
  Wire.endTransmission();
  //Hace la peticion delectura del sensor, esperando 2 bytes de respuesta
  Wire.requestFrom(u8_BH1750_address, 2);

  //Espera la llegada de  dos 2 bytes
  if (2 <= Wire.available()) {
    sensor1_result = Wire.read();
    sensor1_result = sensor1_result << 8;
    sensor1_result |= Wire.read();

  }else{
      sensor1_result=-1;
  }
}
```

El desarrollo de la programación completa se encuentra en el apartado 11.2.2 del presente informe.

4.2.3 Transporte de datos.

El diseño del sistema de monitoreo requiere de al menos un dispositivo para el control y sincronización de la red (Nodo Maestro) y tres dispositivos esclavos (Nodos Remotos) para responder las solicitudes.

La programación desarrollada en el entorno Arduino permite configurar el protocolo MODBUS RTU para el transporte de datos y el modo de acceso al bus de datos (comunicación *half-dúplex*), de cada uno de los dispositivos (Nodo Maestro y Nodos Remotos), como se muestra en la imagen de la Figura 31.

El nodo maestro (Arduino Mega) sincroniza el bus central de datos utilizando el módulo de interfaz serial MAX485 como transmisor en un primer momento para enviar peticiones a los nodos remotos (Arduino nano). Los nodos remotos responden a esas peticiones accediendo al bus de datos por medio del módulo de interfaz serial MAX485.

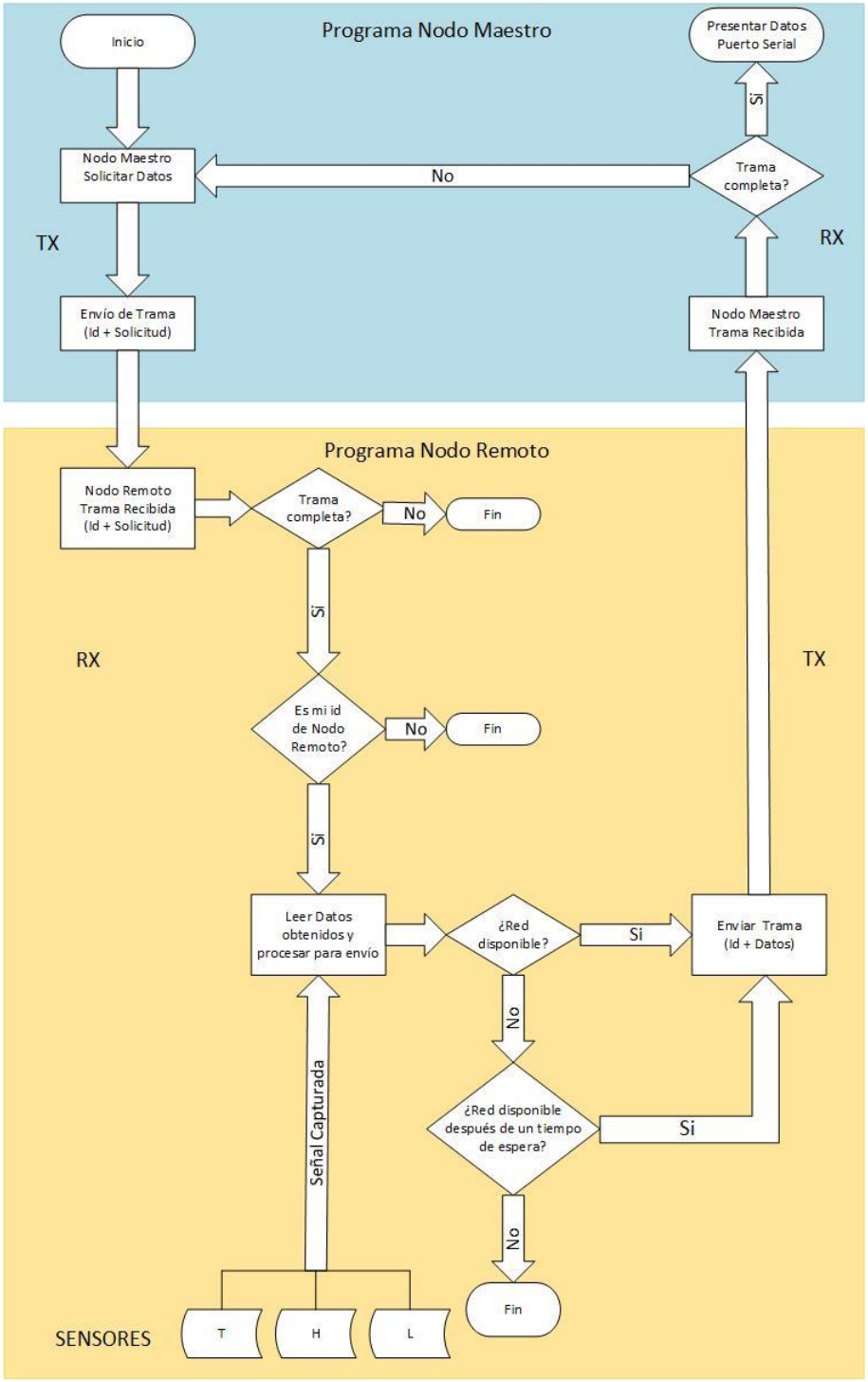


Figura 31. Diseño de programación en Entorno Arduino. Fuente [El Autor]

De acuerdo con el diseño del script (ver Figura 31), se crean sentencias estructuradas que permiten al Nodo Maestro enviar una trama de datos a modo de broadcast (mensaje enviado a todos los dispositivos remotos) para solicitar información. De igual forma, las líneas del código desarrollado, facilitan la respuesta por parte de los Nodos Remotos:

verifican la trama de datos recibida, agrupan los datos (capturados por los sensores) y envían una trama de respuesta utilizando el bus central de datos. Finalmente, la información recibida en el Nodo Maestro, de no existir errores, se presenta por puerto serial. Las sentencias de la programación que se ejecutan en el código de Arduino se describen en los siguientes apartados.

4.2.3.1 Programación del modo de transmisión de la red alámbrica y procesamiento de la trama de datos.

- **Configuración del modo de transmisión de datos en la red alámbrica.**

El modo de comunicación que se diseñó para el sistema de monitoreo en la transmisión de datos es *half-dúplex*. El intercambio de los datos se realiza sin tener pérdidas por colisiones debido a que el bus de datos lo utiliza un dispositivo a la vez. El control del canal de comunicación como transmisor o receptor depende del estado del pin digital (Arduino Mega) al que se conecta el módulo MAX485.

El Arduino Mega controla el bus de datos por medio del módulo MAX485, que se estableció como *transceiver* (TX y RX) y por defecto está siempre configurado en modo de transmisor, mediante la siguiente sintaxis:

```
const int EnTxPin =  2; // HIGH:TX y LOW:RX
digitalWrite(EnTxPin, HIGH); //RS485 como Transmisor
digitalWrite(EnTxPin, LOW); //RS485 como receptor
  delay(60);
```

En los nodos remotos el controlador del bus de datos lo realiza la tarjeta Arduino nano por medio del pin digital (2) al que está conectado el módulo MAX485, los comandos son los siguientes:

```
SoftwareSerial mySerial(10, 11); // RX, TX
//Constantes para comunicacion RS485
const int EnTxPin =  2;  // HIGH:TX y LOW:RX
```

4.2.3.2 Procesamiento de la Trama de datos para la transmisión/recepción.

- **Estructura de la trama de envío desde el Nodo Maestro.**

El Nodo Maestro de la red alámbrica envía una trama por puerto serial, compuesta por números y letras que especifican una cabecera, la dirección única del esclavo, función a realizar y tráiler de comprobación de recepción de trama:

```
('I'); //inicio de trama
(int(101),DEC);//dirección del esclavo
('H'); //H para indicarle que vamos a Leer el sensor
('F'); //fin de trama
```

- **Estructura de la trama de envío desde el Nodo Remoto.**

El Nodo Remoto envía una trama de respuesta que está conformada por una secuencia de números y letras que tiene la siguiente forma:

```
('i'); //inicio de trama
(mydireccion,DEC); //dirección única de esclavo
(h.byte_h, 4); //valor de humedad del sensor
(t.byte_t, 4); //valor de temperatura del sensor
(l.byte_l, 4); //valor de iluminacion del sensor
('f'); //fin de trama
```

- **Estructura de la trama de Recepción en el Nodo Maestro.**

Cuando finaliza el envío de la trama el nodo maestro se pone en modo de recepción en espera de la respuesta de los diferentes nodos remotos conectados a la red. Esto lo realiza mediante la siguiente línea de comando:

```
digitalWrite(EnTxPin, LOW); //RS485 como receptor
```

En los datos que llegan al puerto serial (nodo maestro), la trama recibida empieza por la cabecera enviada por el nodo remoto, para luego leer en el puerto la dirección única del esclavo, los bytes de datos y el tráiler de fin de trama, esto tiene la siguiente forma:

```
(cabecera == 'i')//se busca la cabecera enviada
direccion[0] = Serial1.read();// dirección de esclavo byte 0
```

```
 direccion[1] = Serial1.read();// dirección de esclavo byte 1
 direccion[2] = Serial1.read();// dirección de esclavo byte 2
h1 = Serial1.read();//datos de humedad byte 0
h2 = Serial1.read();//datos de humedad byte 1
h3 = Serial1.read();//datos de humedad byte 2
h4 = Serial1.read();//datos de humedad byte 3
 t1 = Serial1.read();//datos de temperatura byte 0
t2 = Serial1.read();//datos de temperatura byte 1
t3 = Serial1.read();//datos de temperatura byte 2
t4 = Serial1.read();//datos de temperatura byte 3
l1 = Serial1.read();//datos de iluminación byte 0
l2 = Serial1.read();//datos de iluminación byte 1
l3 = Serial1.read();//datos de iluminación byte 2
l4 = Serial1.read();//datos de iluminación byte 3
trailer = Serial1.read();// se lee fin de trama
```

- **Estructura de la trama de Recepción en el Nodo Remoto.**

Los puertos serie de los distintos nodos remotos reciben la trama enviada por el nodo maestro. La trama se compone de: una cabecera, la dirección única del esclavo, la función solicitada y el tráiler o fin de trama, con la siguiente sintaxis:

```
(cabecera == 'I')// Inicio de trama o cabecera enviada por el Nodo
 Maestro
direccion[0] = mySerial.read();// dirección de esclavo byte 0
direccion[1] = mySerial.read();// dirección de esclavo byte 0
direccion[2] = mySerial.read();// dirección de esclavo byte 0
funcion = mySerial.read();// función solicitada por el Nodo Maestro
trailer = mySerial.read(); // tráiler de fin de trama
```

- **Comprobación de trama recibida y respuesta del Nodo Remoto.**

Cuando un nodo remoto recibe una trama de datos por el puerto serie, comprueba si los datos recibidos tienen la estructura de una trama completa. A continuación, verifica si la trama contiene la identificación del nodo remoto para poder ejecutar alguna acción correspondiente. Esto se obtiene mediante las siguientes líneas:

```
(trailer == 'F') // Comprobación de trama completa o tráiler.
direccion_c = atol(direccion); // se verifica la dirección única de
 esclavo.
```

Si la identificación (serie de números recibidos) coincide con la del nodo remoto (serie de números asignados previamente), el nodo remoto se configura como transmisor para responder al nodo maestro, mediante el siguiente comando:

```
digitalWrite(EnTxPin, HIGH); //RS485 como transmisor
```

Para responder el nodo remoto, utiliza una trama de datos que contiene la información solicitada por el nodo maestro, fijando un formato apropiado para la transmisión de datos, de la siguiente forma:

- Definimos estructuras unión para dar a los datos de los sensores

```
//Para la humedad
union h_tag {
 float float_h;//
  byte byte_h[4];
} h;
//Para la temperatura
union t_tag {
  float float_t;
  byte byte_t[4];
} t;
//Para la luz
union l_tag {
  float float_l;
  byte byte_l[4];
} l;
```

- Posteriormente se pasa los bytes de información de los sensores a las siguientes variables:

```
h.float_h = h1; // formato de datos para humedad relativa
 t.float_t = t1; // formato de datos para temperatura
l.float_l = float(lectura); // formato de datos para iluminanción
```

- **Comprobación de trama recibida y presentación de datos por puerto Serial en Nodo Maestro.**

Cuando la trama de datos es recibida en el puerto serial del nodo maestro, el nodo maestro realiza una verificación de la trama. Si la trama no está completa se rechaza y se prepara el nodo maestro para enviar nuevamente la solicitud al nodo remoto. Esta comprobación se realiza mediante la sentencia:

```
(trailer == 'f') //se comprueba el tráiler o fin de trama.
```

Cuando el nodo maestro comprueba que la trama recibida es la correcta, procede a convertir los valores leídos por puerto serie, de lenguaje de máquina (bytes) a valores flotantes, que pueden ser entendidos por el usuario. Esto realiza de la siguiente manera:

```
//Para la humedad
h.byte_h[0] = h1;
h.byte_h[1] = h2;
h.byte_h[2] = h3;
h.byte_h[3] = h4;

//Para la temperatura
t.byte_t[0] = t1;
t.byte_t[1] = t2;
t.byte_t[2] = t3;
t.byte_t[3] = t4;

//Para la luminancia
l.byte_l[0] = l1;
l.byte_l[1] = l2;
I.byte_l[2] = l3;
l.byte_l[3] = l4;
```

La información que se presenta en el puerto serial del nodo maestro tiene una estructura que identifica además del origen de procedencia de la información (mediante la dirección única del nodo remoto) los datos de las variables capturadas, separadores de caracteres (comas entre datos) que facilitan la gestión al utilizar programación en Python. La identificación se realiza de la manera siguiente:

```
(atol(direccion) == 101) //dirección de esclavo, en este caso 101.
Serial.print("S1"); // Nombre para identificación de nodo
Serial.print(","); // separador de caracteres.
Serial.print(h.float_h,2); //datos de humedad en formato flotante
Serial.print(",");// separador de caracteres
Serial.print(t.float_t,2); //datos de temperatura en formato flotante
Serial.print(",");// separador de caracteres
Serial.print(l.float_l,2); //datos de luminancia en formato flotante
Serial.println(" ");// salto de línea
```

La programación completa del nodo maestro y los nodos remotos, desarrollada en el software Arduino, se presenta en el apartado11.2

4.2.4 Almacenamiento de la Información.

La programación desarrollada en Python 3.0 se utiliza en el diseño de sistema de monitoreo, con el propósito de leer los datos que se encuentran en el puerto serial de Arduino (Mega) y realizar la conexión e inserción de los datos, en una base de datos creada simultáneamente en PostgreSQL.

La herramienta informática Python permite también crear una alerta inmediata para informar al personal de turno acerca de un evento (peligro) registrado. El sistema de alerta diseñado se ejecuta cuando se registra un incremento de la variable de temperatura originado en cualesquiera de las tres áreas de intervención y que supere un umbral establecido. La alerta consiste en el envío de un mensaje vía correo electrónico. La programación del script se diseña de acuerdo al diagrama de flujo mostrado en la imagen de la Figura 32.

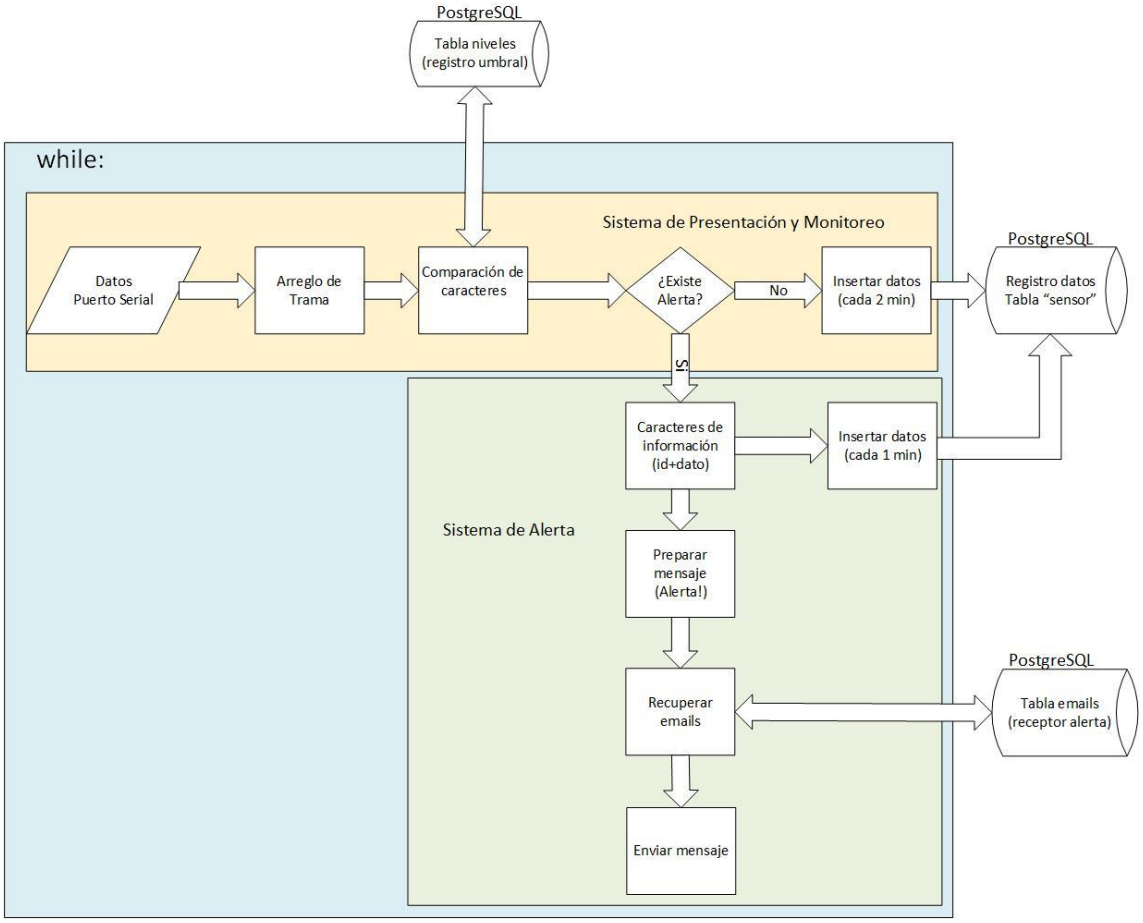


Figura 32. Diseño de la programación en Python para almacenamiento de datos. Fuente [El Autor]

4.2.4.1 Procesamiento de la trama información para almacenamiento mediante la programación de Python.

Python presenta una extensa gama de librerías para lectura de datos mediante serial y su posterior procesamiento o gestión usando sentencias ordenadas en un script. A continuación, se explica en los siguientes apartados la programación requerida para el sistema diseñado.

- **Lectura del Puerto Serial.**

La lectura del puerto serial se realizó usando la librería *"Pyserial"* que posibilitó la comunicación entre Python y Arduino. Los detalles técnicos de velocidad de conexión en baudios y tiempo de espera para abrir el puerto serial se describen a continuación:

```
import serial #librería para comunicación entre Python y Arduino.
ser = serial.Serial("/dev/ttyACM0", 9600, timeout=10) #abrir el puerto,
 velocidad y tiempo
```

- **Librería utilizada para la conexión con base de datos PostgreSQL.**

La conexión con la base de datos en el SGDB PostgreSQL versión 10.0 se realizó con ayuda de la librería psycopg2. La base de datos denominada *"datosarduino"*, se encuentra en un servidor local *("localhost")* dentro de la red del ECU911 (mayor detalle sobre el esquema de la base de datos denominado *"datosarduino"* se presenta en el apartado 4.2.4.3). A efectos de privacidad, en este documento se tratará el usuario de la base de datos como *"user"* con contraseña de *"password"*. Las sentencias de conexión a la base de datos son las siguientes:

```
import psycopg2 # librería para conexión con base de datos PostgreSQL.
conn=   psycopg2.connect(host="localhost",   database="datosarduino",
 user="user", password="password") # conexión con la base de datos.
```

- **Transferencia de datos a PostgreSQL mediante función *cursor()*.**

La inserción de los valores correspondientes a cada tabla y columna en la base de datos, se realizó con la función *"cursor ()"*. Las sentencias utilizadas para este propósito son:

```
cur = conn.cursor()# establecemos la conexión del cursor
cur.execute(posgres_insert)# transferir los datos a PostgreSQL
conn.commit()# empezar la transferencia con la base de datos.
cur.close()# cerrar el cursor
```

- **Cerrar conexión con la base de datos PostgreSQL.**

Luego de enviar los datos al servidor se procedió a cerrar la conexión para indicar la finalización de la transferencia de datos. Esto se realizó mediante la siguiente línea de comando:

```
#cerrar la conexión a PostgreSQL.
conn.close()
```

- **Inserción de trama de datos en PostgreSQL**

La trama de inserción de datos se realizó una vez identificado el Nodo Remoto desde el cual provienen, así como la asignación de la tabla en la cual se alojarán los datos. Las columnas correspondientes a los diferentes datos (humedad, temperatura, e iluminación respectivamente) y los datos o valores numéricos; así mismo la trama la completa la variable *"NOW"* referente a la fecha en la cual se insertan los datos.

A fin de reducir la duplicación de código, se creó una variable auxiliar denominada *"columnas_sensor"* que contiene los nombres de las columnas de la tabla sensor que van a recibir los datos. En la siguiente sintaxis se puede verificar lo explicado:

```
# Tabla y columnas para transferencia de datos.
columnas_sensor = "id, n_sensor ,dshum, dstem, dslux, fecha"
```

La inserción de los datos del nodo remoto se realiza con la sentencia:

```
//Para el Nodo Remoto 1:
if                  ("S1"                  in                  str(rx)):
 postgres_insert = "INSERT INTO sensor (" + columnas_sensor + ") VALUES
 ( nextval('serial'),1, " + rx_aux + ",NOW());"

//Para el Nodo Remoto 2:
elif                 ("S2"                 in                  str(rx)):
 postgres_insert = "INSERT INTO sensor (" + columnas_sensor + ") VALUES
 ( nextval('serial'),2, " + rx_aux + ",NOW());"

// Para el Nodo Remoto 3:
elif ("S3" in str(rx)):
  postgres_insert = "INSERT INTO sensor (" + columnas_sensor + ") VALUES
 ( nextval('serial'),3, " + rx_aux + ",NOW());"
```

Los extractos de código Python presentados en este apartado se ejecutan dentro de un bucle *"while"* que está escuchando permanentemente en el puerto serial del nodo maestro.

4.2.4.2 Programación de sistema de alerta vía email.

Como aporte adicional al sistema propuesto se incluyó la programación en Python de una alerta vía email dentro del bucle *"while"* que procesa la información para la base de datos. Esto permite tener una alerta casi instantánea de posibles incrementos de temperatura sobre el nivel deseado. El código implementado se compone de tres partes principales como: 1) comparar los datos recibidos por puerto serial con los umbrales máximos de temperatura registrados en la tabla niveles; 2) Caracterizar la alerta; y 3) envío de alerta. A continuación, se describe el programa desarrollado en Python para el envío de la alerta. Se utiliza la librería *"import smtp"* de Python para llevar a cabo el siguiente procedimiento.

- **Comparación de datos recibidos por puerto serial con los umbrales máximos de temperatura registrados en la tabla niveles y envío de alertas.**

Los datos que se reciben por puerto serial *("rx_auxx")* son los que se van comparando constantemente con la información de los umbrales máximos de temperatura *("rx_auxxi")* registrados en la tabla niveles dentro de la base de datos *"datosarduino"*.

La verificación de los datos de umbrales máximos registrados en la base de datos se consultó y almacenó en una variable *"rows"*, de donde se extrajeron los nombres de los sensores en variables separadas *("rx_auxxi"*). Esto se realizó con las siguientes líneas:

```
cur = conn.cursor()
niveles_sql = "SELECT nivmax, dispos FROM niveles where estado='A'"
cur.execute(niveles_sql)
rows = cur.fetchall()
for row in rows:
    if row[1].strip() == 's1':
        rx_auxx1 = row[0]
    if row[1].strip() == 's2':
        rx_auxx2 = row[0]
    if row[1].strip() == 's3':
        rx_auxx3 = row[0]
```

En cada ejecución del bucle, la cadena proveniente del puerto serial ser se almacena en un objeto denominado *"rx"* que contiene los valores de la trama explicados

anteriormente. Por lo tanto, debe ser manipulada mediante funciones de tratamiento de textos. Tal como se muestra en las siguientes líneas de Python:

```
            rx = ser.readline()
            rx_auxx = str(rx)
            rx_auxx = rx_auxx.replace('\\r\\n', ' ')
            rx_auxx = rx_auxx.replace('\\', '')
            rx_auxx = rx_auxx.rsplit(",")
```

El objeto *"rx_auxx"* es un arreglo, donde cada elemento es el valor correspondiente a: *"id de sensor"*, *"humedad relativa"*, *"temperatura"* y *"luminancia"*. A partir de este arreglo establece una comparación de valores de temperatura, de la siguiente forma: Si han pasado 60 segundos de espera, y *"rx"* contiene un valor de temperatura que supera el umbral declarado para el sensor correspondiente y si la comparación es verdadera, entonces se procede a enviar una alerta con la información (del área de origen y los valores superados) al personal de turno encargado. El envío de las alertas se realiza mediante protocolo *smtp*, con las siguientes instrucciones:

```
 # Definir el nombre del area de afectacion
 origen = " "
if rx_auxns == 's1':
  origen = "ups"
if rx_auxns == 's2':
  origen = "data center"
 if rx_auxns == 's3':
  origen = "sala de operaciones"

# consulta de los correos del personal de turno para enviar el mail
nivel_sql = "SELECT emailp, emails, emailt FROM emails where estado='A'"
cur.execute(nivel_sql)
rows = cur.fetchall()
for row in rows:
correo1 = row[0]
correo2 = row[1]
correo3 = row[2]
```

```
# especificar el servidor de iniciar TLS
s = smtplib.SMTP('smtp.gmail.com', 587)
s.starttls()

# autenticar la cuenta
s.login("ecu911.loja.tecnologia@gmail.com", "contraseña")

# mensaje a enviar
message = "Alerta! Temperatura más allá del nivel permitido para el
sensor " + origen

# enviar el mail  (de quien envía, quien recibe, mensaje a enviar)
s.sendmail("ecu911.loja.tecnologia@gmail.com", correo1, message)
s.sendmail("ecu911.loja.tecnologia@gmail.com", correo2, message)
s.sendmail("ecu911.loja.tecnologia@gmail.com", correo3, message)

# culminar la sesión
s.quit()
```

Cuando no existen condiciones para enviar una alerta, el sistema registra directamente los datos en la tabla *"sensor"* cada dos minutos. Para mayor detalle sobre el script completo desarrollado en Python 3.0 ver el apartado 11.3.1

4.2.4.3 Almacenamiento de información en el Gestor de base de datos PostgreSQL 10.0.

La configuración de la base de datos se realizó a través *PgAdmin 3*. La base de datos se denomina "datosarduino" y contiene tres tablas: 1) *sensor*, que contiene la información de monitoreo recogida desde los sensores instalados en las áreas de interés; 2) *niveles*, que contiene la información del umbral máximo de temperatura que debería registrar los sensores; 3) *emails*, que contiene la información de contacto para comunicar una alarma en el caso de que un registro de temperatura supere el umbral en la tabla de *niveles*. De estas tres tablas, las tablas *niveles* y *emails* fueron incluidas como un valor agregado del trabajo realizado, pero se puede prescindir de ellas en cualquier momento.

La información que se almacena en la tabla *sensor* se identifica con una clave única *"id"* para cada registro que contiene: el nombre del Nodo Remoto *"n_sensor"*, los datos de

humedad *"dshum*", temperatura *"dstem"* e iluminación *"dslux"* respectivamente, y la *"fecha"* de inserción en la base de datos (que equivale a la fecha de monitoreo). La Figura 33 muestra el esquema de la base de datos implementada en PostgreSQL versión 10.0

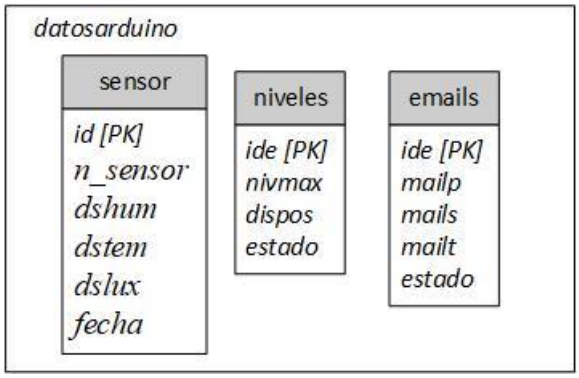


Figura 33.Diseño de la base de datos "datosarduino". Fuente [El Autor]

- **SQL para la creación del esquema de la base de datos *"datosarduino".***

Según el diseño de base de datos propuesto, se requiere crear las tres tablas mencionadas, con sus respectivos campos. Los scripts SQL utilizan los comandos CREATE TABLE y ALTER para especificar las condiciones deseadas en cada entidad.

- ***Creación del esquema para la tabla sensor.***

La primera tabla "sensor" (ver Tabla 12), fue creada con las siguientes condiciones:

Tabla 12. Diseño de tabla sensor.

Nombre	Descripción	Tipo de dato
Id	Identificador del registro. Clave primaria	Integer not null
n_sensor	identificador del sensor o nodo remoto	integer
Dshum	Texto de dato de humedad	character (8)
Dstem	Texto de dato de temperatura	character (8)
Dslux	Texto de dato de iluminación	character (8)
Fecha	Línea de tiempo	Timestamp without time zone

Fuente: [El Autor]

La cantidad de datos que se pueden asociar con el *"integer not null"* se encuentra en el rango de 2*10^8 de registros no nulos. En lenguaje SQL para PostgreSQL este esquema corresponde al siguiente script:

```
CREATE TABLE public.sensor (
 id integer NOT NULL,
 n_sensor integer,
 dshum character(8),
 dstem character(8),
 dslux character(8),
 fecha timestamp without time zone DEFAULT CURRENT_TIMESTAMP,
 CONSTRAINT sensor_pkey PRIMARY KEY (id)
) WITH (OIDS=FALSE);
```

Según la web oficial, se recomienda desactivar el parámetro OIDS cuando se usa la versión 10 de PostgreSQL, debido a que deja de ser útil en esta nueva versión. Para desactivar este parámetro se debe fijar como falso. La columna id es una secuencia autoincrementada que contiene un identificador único para cada registro. Esta secuencia se debe generar con la siguiente sentencia SQL:

```
 create sequence serial start 1
  select nextval('serial')
```

Después de crear la Tabla 12, se requiere fijar el propietario de la tabla de acuerdo a la necesidad del sistema de monitoreo. A efectos de este documento, el usuario del sistema se mostrará como “user”. La siguiente línea SQL permite fijar lo expuesto anteriormente:

```
ALTER TABLE public.sensor OWNER TO user;
```

- ***Creación del esquema de la tabla “niveles”.***

El diseño de la segunda de tabla “niveles” dispone la distribución de columnas indicando el nivel umbral máximo de temperatura, así como el nombre del Nodo Remoto que captura el evento, como se muestra en la Tabla 13:

Tabla 13. Diseño de tabla niveles.

Nombre	Descripción	Tipo de dato
ide	Identificador del registro. Clave primaria	Integer not null
nivmax	Texto nivel de umbral máximo de Temperatura	integer
dispos	Texto nombre del dispositivo o Nodo Remoto	character (4)
estado	Texto de estado del nivel “A” (activo)	character (1)

Fuente: [El Autor]

La secuencia de comandos SQL para crear la tabla niveles son los siguientes:

```
create sequence niveles_sec;
create table niveles(
ideniveles int default nextval ('niveles_sec') primary key,
id integer NOT NULL,
nivmax integer,
dispos char (4),
estado char (1)
)
```

En la tabla niveles, en la columna *"dispos"* se describe el nombre de los nodos remotos como: el área de UPS que es el Nodo Remoto 1 se denomina "s1", área de DATACENTER o Nodo Remoto 2 con "s2" y área de SALA OPERATIVA o Nodo Remoto 3 nombrado como "s3".

- ***Creación del esquema de la tabla "emails".***

La tercera tabla creada en la base de datos fue *"emails"* (ver Tabla 14), y contiene la siguiente estructura:

Tabla 14. Diseño de tabla emails

Nombre	Descripción	Tipo de dato
Ide	Identificador del registro. Clave primaria	Integer not null
Mailp	Texto de Email de primer contacto	character (60)
Mails	Texto de Email de segundo contacto	character (60)
Mailt	Texto de Email de tercer contacto	character (60)
Estado	Texto de estado de email "A" (activo)	character (1)

Fuente: [El Autor]

El script SQL utilizado para crear la tabla *"emails"* contiene los siguientes comandos:

```
create sequence emails_sec;
create table emails(
    idemils int default nextval ('emails_sec') primary key,
    emailp char (60),
    emails char (60),
    emailt char (60),
    estado char (1))
```

4.2.5 Gestión de información mediante aplicación web para presentación y monitoreo.

La gestión de información se realiza mediante herramientas informáticas de la pila Elastic versión 6.2.4. Estas herramientas permiten trabajar con los datos de PostgreSQL en diferentes etapas (Figura 34): Primero, se requiere *Logstash* para solicitar información al SGBD y transformarla para su análisis. Segundo, a partir de la información recopilada por *Logstash*, *Elasticsearch* permite establecer condiciones de búsqueda y análisis para finalmente presentarla en la tercera herramienta informática *Kibana*. *Kibana* permite configurar un *dashboard* que muestra la información de forma organizada y entendible para el personal de turno que hace el monitoreo.

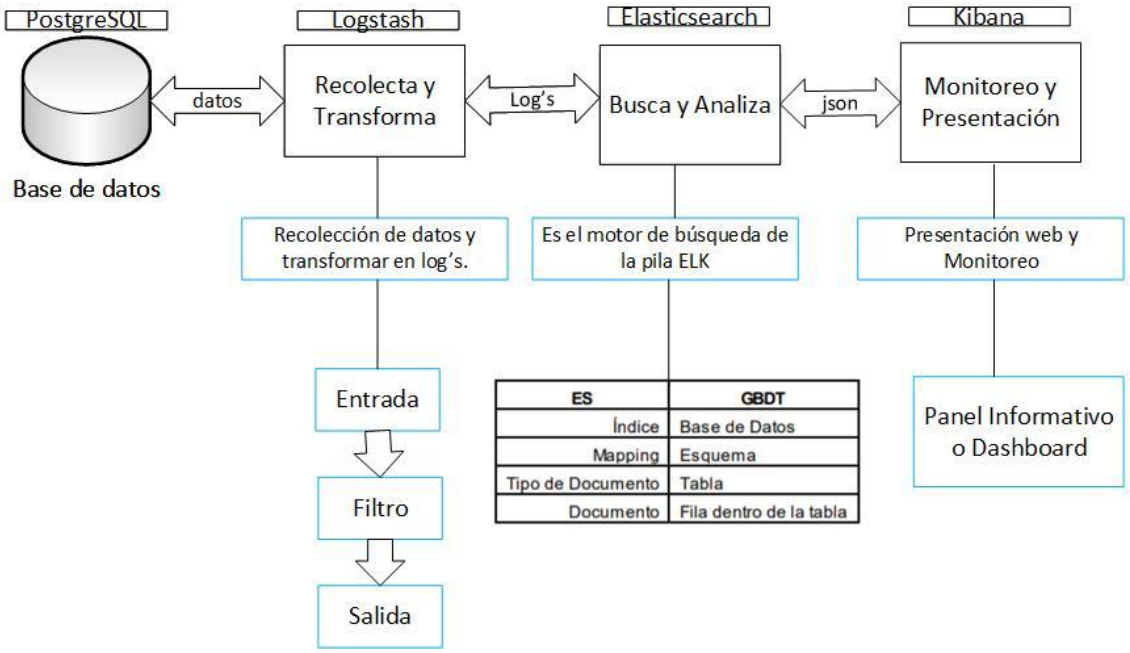


Figura 34. Esquema de gestión de información mediante pila ELK. Fuente [El Autor]

4.2.5.1 Instalación y Configuración de la pila ELK 6.2.4

La instalación de la pila ELK se realizó siguiendo los tutoriales existentes en la página oficial de la pila Elastic (Elasticsearch , 2018). Los archivos para la instalación de la pila Elastic se pueden descargar de la página oficial.

- **Gestión de la información almacenada en PostgreSQL mediante herramienta Logstash.**

La consulta de los datos, que se encuentran en la tabla *"sensor"* de PostgreSQL, desde *Logstash* requiere configurar un esquema y la forma en que se va a gestionar la información. El esquema básico necesita de una entrada (*input),* un filtro (*filter*) (sustituible por un vacío en este sistema) y una salida (*output).*

- *Input:* se realiza la conectividad con la base de datos por medio de los comandos *jdbc* (*java database connectivity*/conectividad de base de datos java). Seguidamente se toma los parámetros de usuario y contraseña configurados en la base de datos PostgreSQL y se procede a leer la información almacenada en la tabla. La línea de la script descrita como *"tracking_column => fecha"* permite la consulta continua de la información y permite la recuperación de nueva información.
- *Filter:* en el desarrollo del diseño del sistema de monitoreo no se utiliza ningún filtrado de la información debido a se requiere del ingreso constante de datos para el monitoreo en tiempo real.
- *Output:* luego de procesar la información que se conduce desde la base de datos de PostgreSQL mediante el bloque *"input"*, se generan *logs* en formato. json, adecuados para la indexación en el *Elasticsearch*, la dirección de host donde se almacenan ("https://127.0.0.1:9200"]) y el nombre de la indexación proporcionado por *Elasticsearch* ("elk-2018.09.21").

A continuación, se muestra el script de configuración de la herramienta *Logstash*:

```
input {
  jdbc {
    jdbc_driver_library => "/home/user/Data/postgresql-42.2.2.jar"
    jdbc_driver_class => "org.postgresql.Driver"
    jdbc_connection_string                                          =>
 "jdbc:postgresql://localhost:5432/datosarduino"
    jdbc_user => "user"
    jdbc_password => "password"
    schedule => "* * * * *"
    statement => "SELECT * from sensor
```

```
    use_column_value => true
    tracking_column => fecha
  }
}
filter {
}
output{
 stdout { codec => json }
 elasticsearch {
  hosts => ["https://127.0.0.1:9200"]
  index => "elk-2018.09.21"
 }
}
```

- **Indización y alojamiento de la información recolectada por Logstash mediante la herramienta Elasticsearch.**

Configuramos *Elasticsearch* de forma que se establezca un esquema del índice, esto se denomina como mapeo, y se refiere a establecer en *Elasticsearch* el tipo de atributos que se da a los *logs* que contienen la información. Los datos que ingresan de la tabla *"sensor"* a *Elasticsearch* son indexados como *logs* (línea de palabras) con formato. JSON que contiene los campos y valores de los datos originales, lo que permiten almacenarlos y recuperarlos (cuando se necesite realizar una consulta).

El esquema básico del índice se compone principalmente por el nombre de la indexación, tipo, id (proporcionado por *Elasticsearch*) y versión como se puede observar en las cuatro primeras líneas del script mostrado a continuación:

```
{
  "_index": "elk-2018.09.21",
  "_type": "log",
  "_id": "GkzOIWYBRSLaWxFzwBbo",
  "_version": 1,
  "_score": null,
  "_source": {
    "@version": "1",
    "n_sensor": 1,
```

```
    "dshum": 51,
    "dstem": 19.2,
    "id": 782485,
    "dslux": 0,
    "@timestamp": "2018-09-28T20:53:00.161Z",
    "fecha": "2018-09-28T20:52:14.925Z"
  },
  "fields": {
    "fecha": [
      "2018-09-28T20:52:14.925Z"
    ],
    "@timestamp": [
      "2018-09-28T20:53:00.161Z"
    ]
  },
  "sort": [
    1538167934925
  ]
}
```


Desde la sexta línea del script anterior, *source*, los siguientes parámetros se refieren básicamente al detalle de la base de datos desde donde se toma la información requerida, en nuestro caso de la tabla *"sensor"* que consta de las columnas: *"n_sensor"*, *"dshum"*, *"dstem"*, *"dslux"*, *"fecha"* (ver Figura 45). La línea *id* describe la identificación de la tabla sensor que proporciona PostgreSQL. Las líneas detalladas como *"timestamp"* muestran la fecha actual de indización de la información.

De no realizar este tipo de configuración *Elasticsearch* agrega por defecto un formato genérico para la indización de los datos la cual no satisface los requerimientos de consulta de este sistema.

- **Gestión de la información almacenada en Elasticsearch para presentación y monitoreo mediante herramienta Kibana.**

Kibana permite realizar la estructura de presentación de la información del sistema de monitoreo de forma gráfica, donde se selecciona el tipo de métrica más conveniente para

representar los datos. El panel informativo diseñado, tiene el tipo de visualización de datos, agrupados en el entorno gauge y seleccionados como se muestra en la Figura 35:

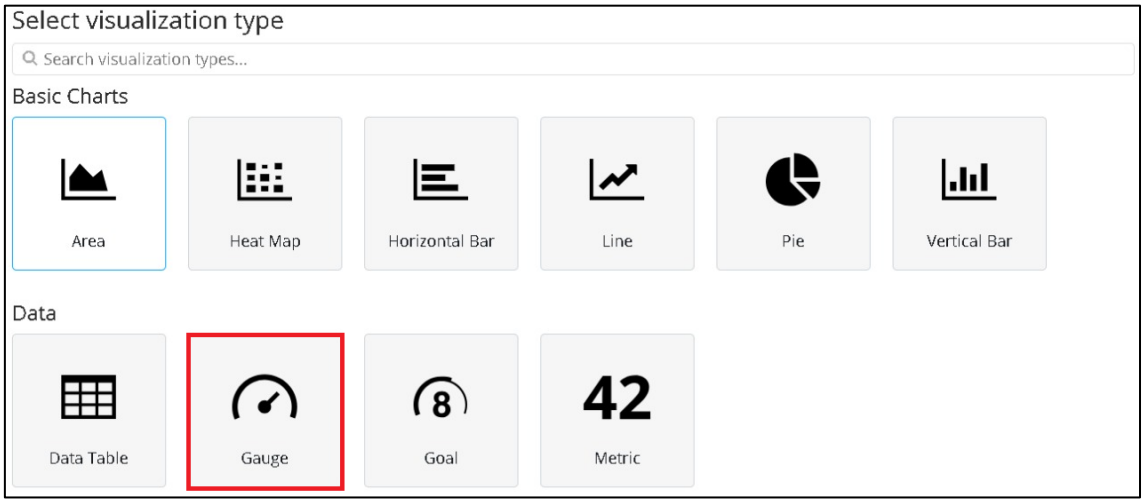


Figura 35. Configuración del Gauge para presentación de datos. Fuente [El Autor]

Luego de seleccionar el panel informativo o *dashboard* adecuado, se configura los niveles de las métricas que sirven para el monitoreo visual. Por ejemplo, los niveles aceptables de temperatura (línea de color verde del Gauge), nivel de riesgo de temperatura (línea de color amarillo del Gauge) y nivel de peligro de temperatura (línea de color rojo del Gauge). De la misma manera, se fija los parámetros de estilo para los datos obtenidos de la variable de humedad relativa, temperatura y los datos de iluminación. Se realiza este procedimiento para cada una de las áreas críticas desde donde se obtienen los datos de las variables ambientales. En la Figura 36 se verifica el detalle de la configuración.

Kibana también dispone de herramientas administrativas para verificar el funcionamiento en conjunto de los dos componentes (*Logstash* y *Elasticsearch*) utilizados para el desarrollo del diseño del sistema de monitoreo como se muestra en la Figura 37.

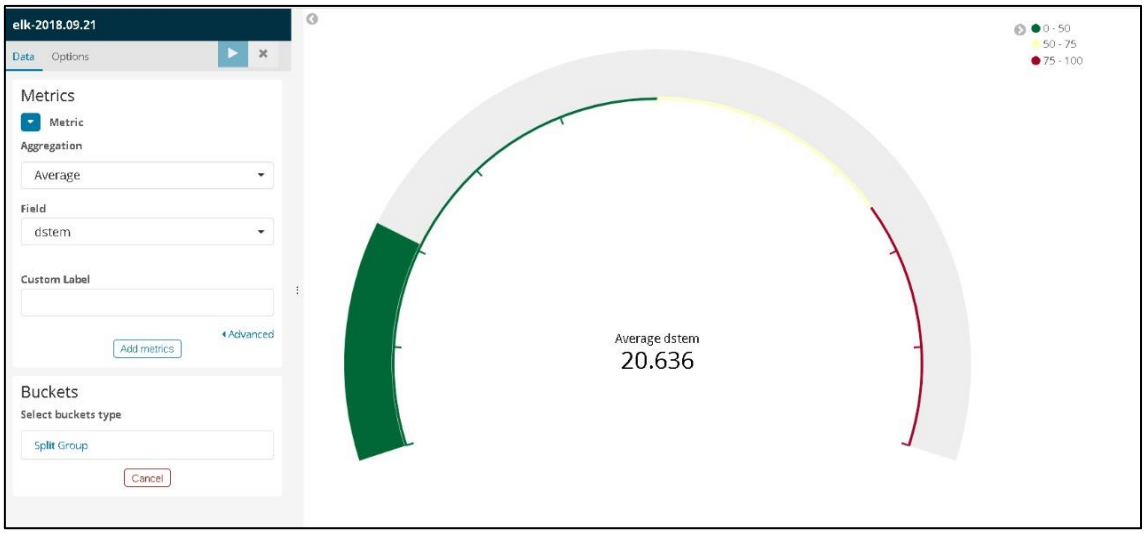


Figura 36. Configuración del Gauge para la presentación de los datos mediante Kibana. Fuente [El Autor]

En la Figura 37 se muestra la información de la indización los datos como *logs*, el esquema del índice de datos y se puede también configurar el formato de presentación de los datos (enteros, decimales, de coma flotante, etc.) aunque en el diseño del sistema de monitoreo se toma la información correspondiente como la gestiona *Logstash*.

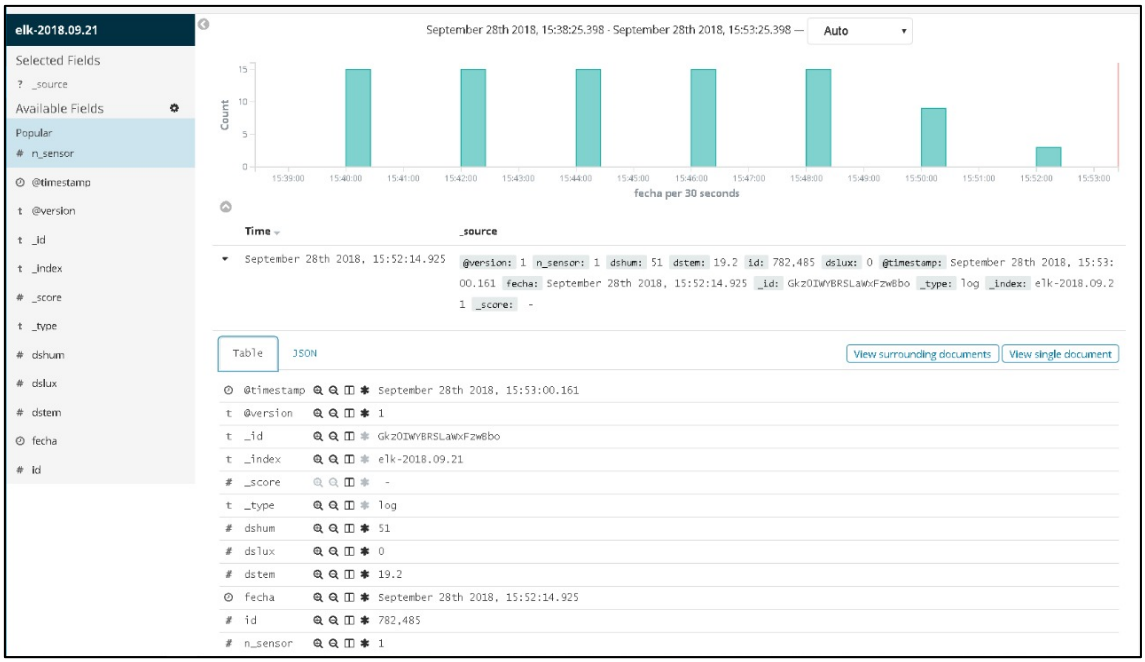


Figura 37. Visualización de indización de la información mediante herramienta Kibana. Fuente [El Autor]

5. Resultados.

5.1 Implementación del sistema de monitoreo en tiempo real de variables ambientales en las áreas críticas del edificio SIS ECU-911 Loja.

5.1.1 Módulos construidos para el sistema de monitoreo de variables ambientales.

El diseño del sistema está basado en la topología MODBUS (comunicación *half-dúplex*), consta de un nodo maestro que es el controlador de la red de comunicación y tres nodos remotos que funcionan como esclavos de la red.

El nodo Maestro está conformado básicamente por el Arduino Mega y el módulo MAX485. El Arduino Mega se encarga del control de envío de peticiones y la recepción de respuestas a través del módulo MAX485 que facilita la transmisión de esos datos. La estructura del módulo se lo puede ver en la Figura 38. Así mismo está configurado para presentar los datos recibidos por puerto serial desde donde se procederá a tomar y procesar esos datos en un nivel superior.

Figura 38. Componentes de Nodo Maestro. Fuente [El Autor]

Cada uno de los nodos remotos está compuesto principalmente por un Arduino Nano, un sensor de temperatura - humedad relativa (DHT22), un sensor de luminosidad (BH1750) y un módulo MAX485. Los sensores se utilizan para captar la información de las variables ambientales en las áreas donde fueron colocados y el MAX485 permite la comunicación con el nodo maestro, formando la red alámbrica MODBUS RS-485. En la Figura 39 se puede verificar los componentes del Nodo Remoto 2. Los componentes de los demás nodos remotos son similares a los mostrados en la Figura 39.

Figura 39. Componentes de Nodo Remoto. Fuente [El Autor]

5.1.2 Implementación de los módulos del sistema de monitoreo en las áreas críticas del edificio SIS Ecu-911 Loja y adquisición de datos

Los módulos construidos para el sistema de monitoreo de variables ambientales, se implementaron en las áreas críticas del edificio SIS ECU-911 Loja definidas por el personal técnico de la institución. Se utilizó cable UTP categoría 5 con conectores RJ45 para la transmisión de datos y que permiten redundancia de canal físico de la red de

comunicación alámbrica. En la Figura 40, se muestra el plano de las áreas críticas en donde se instalaron el Nodo Maestro y los tres Nodos Remotos.

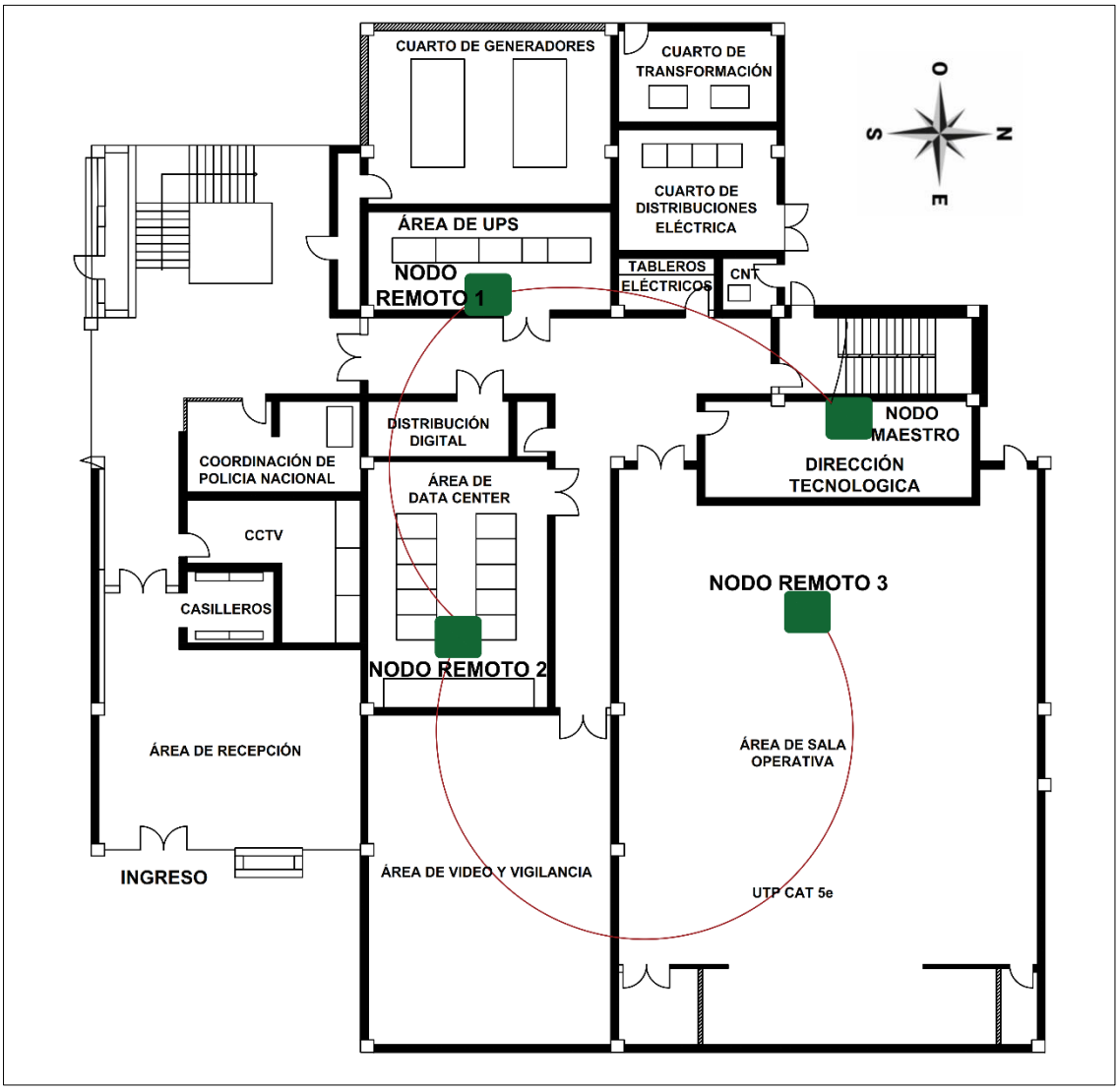


Figura 40. Plano de instalación de los Nodos Maestro y Remotos en el edificio SIS ECU-911 Loja. Fuente [El Autor]

El nodo maestro se procedió a ubicar en el área Técnica, desde donde se comunica con un servidor local por medio de puerto USB. En la Figura 41 se puede ver la ubicación del módulo del nodo maestro.

Figura 41. Nodo Maestro. Fuente [El Autor]

Los módulos de los nodos remotos se instalaron en lugares definidos por el personal técnico del SIS ECU-911 Loja como áreas críticas, así: El módulo del Nodo Remoto 1 se alojó en el área de UPS (ver Figura 42); el Nodo Remoto 2 en el área del *Data Center* (ver Figura 43); y el Nodo Remoto 3 en el área de Sala Operativa (ver Figura 44).

Figura 42. Nodo Remoto 1. Área de UPS. Fuente [El Autor]

Figura 43. Nodo Remoto 2. Área Data Center. Fuente [El Autor]

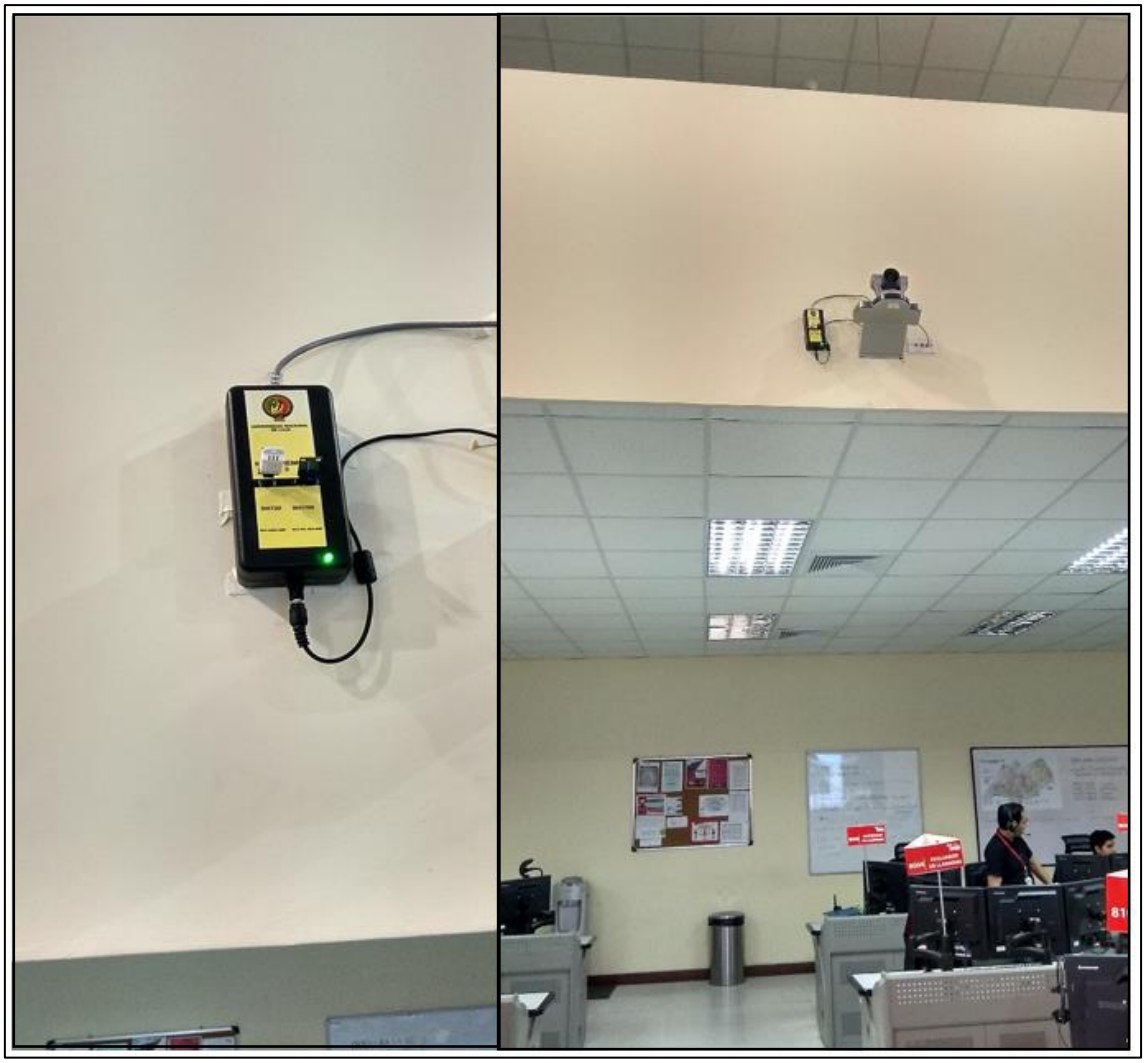


Figura 44. Nodo Remoto 3. Área Sala Operativa. Fuente [El Autor]

El Nodo Remoto 3 es el punto más alejado en la red alámbrica (76 metros) desde el nodo maestro. A pesar de ello, la información que se envía/recibe entre estos dos dispositivos no se ve afectada por interferencias, debido a la naturaleza de la topología de red que se empleó.

5.1.3 Transporte de datos

El sistema propuesto cuenta con tres etapas para el transporte de datos de las condiciones ambientes de las áreas críticas. La programación y configuración realizada para cada etapa, permite que:

- En el nodo maestro se sincronice la red MODBUS y se centralice la información solicitada mediante una secuencia predefinida. Los nodos remotos utilizan la red alámbrica uno a la vez previa identificación mediante un código único para enviar la trama de datos solicitada por el nodo maestro.
- En la red alámbrica y en el transporte de datos exista confiabilidad de los datos y la baja pérdida de la información.
- Se disponga la información centralizada en el puerto serial de un servidor local (designado por personal técnico del ECU-911 Loja), para su almacenamiento el servidor de base de datos PostgreSQL.

5.1.4 Almacenamiento de Información.

Los datos que son enviados por puerto serial, se almacenan en la base de datos de PostgreSQL en una tabla denominada "*sensor"* como se muestra en la Figura 45.

File Edit View Tools Help

No limit

	id [PK] integer	n_sensor integer	dshum numeric(5,2)	dstem numeric(5,2)	dslux numeric(6,2)	fecha timestamp without time zone
278670	491189	2	66.50	21.20	445.00	2018-07-18 17:36:08.343094
278671	491190	3	37.20	23.70	167.00	2018-07-18 17:36:08.343094
278672	491191	1	42.60	19.50	0.00	2018-07-18 17:37:09.055599
278673	491192	2	55.80	20.70	444.00	2018-07-18 17:37:09.055599
278674	491193	3	36.80	23.70	169.00	2018-07-18 17:37:09.055599
278675	491194	1	42.00	19.20	0.00	2018-07-18 17:38:10.894515
278676	491195	2	48.60	18.70	442.00	2018-07-18 17:38:10.894515
278677	491196	3	36.70	23.70	166.00	2018-07-18 17:38:10.894515
278678	491197	1	41.80	19.00	0.00	2018-07-18 17:39:12.172239
278679	491198	2	47.40	17.30	440.00	2018-07-18 17:39:12.172239
278680	491199	3	36.80	23.70	168.00	2018-07-18 17:39:12.172239
278681	491200	1	43.70	19.20	0.00	2018-07-18 17:40:13.433477
278682	491201	2	47.10	16.40	438.00	2018-07-18 17:40:13.433477
278683	491202	3	37.10	23.70	167.00	2018-07-18 17:40:13.433477
278684	491203	1	45.60	19.50	0.00	2018-07-18 17:41:15.812893
278685	491204	2	46.60	15.80	436.00	2018-07-18 17:41:15.812893
278686	491205	3	36.70	23.70	166.00	2018-07-18 17:41:15.812893
278687	491206	1	46.30	19.80	0.00	2018-07-18 17:42:17.078236
278688	491207	2	52.20	16.20	433.00	2018-07-18 17:42:17.078236
278689	491208	3	37.20	23.70	167.00	2018-07-18 17:42:17.078236
278690	491209	1	45.50	19.70	0.00	2018-07-18 17:43:19.457622
278691	491210	2	63.90	17.90	437.00	2018-07-18 17:43:19.457622
278692	491211	3	37.30	23.80	166.00	2018-07-18 17:43:19.457622
278693	491212	1	44.40	19.30	0.00	2018-07-18 17:44:21.275912
278694	491213	2	66.80	19.40	439.00	2018-07-18 17:44:21.275912
278695	491214	3	36.90	23.80	166.00	2018-07-18 17:44:21.275912

Figura 45. Base de datos en PostgreSQL. Fuente [El Autor]

5.1.5 Gestión, Presentación de Información en interfaz web y Monitoreo.

El interfaz web de las variables ambientales de temperatura, humedad relativa e iluminación se presenta en la Figura 46. Los datos representados son de fácil interpretación para el usuario con indicativos que detallan niveles de aceptación, prevención y riesgo, así como registro del tiempo actualizado de la información. El monitoreo de la data se realiza de forma autómata por medio de la herramienta *Kibana* como se muestra en el Figura 47. El personal técnico del edificio SIS ECU-911 Loja también puede monitorear los datos de manera remota, ya que el servidor donde se aloja la aplicación web tiene un enlace público.

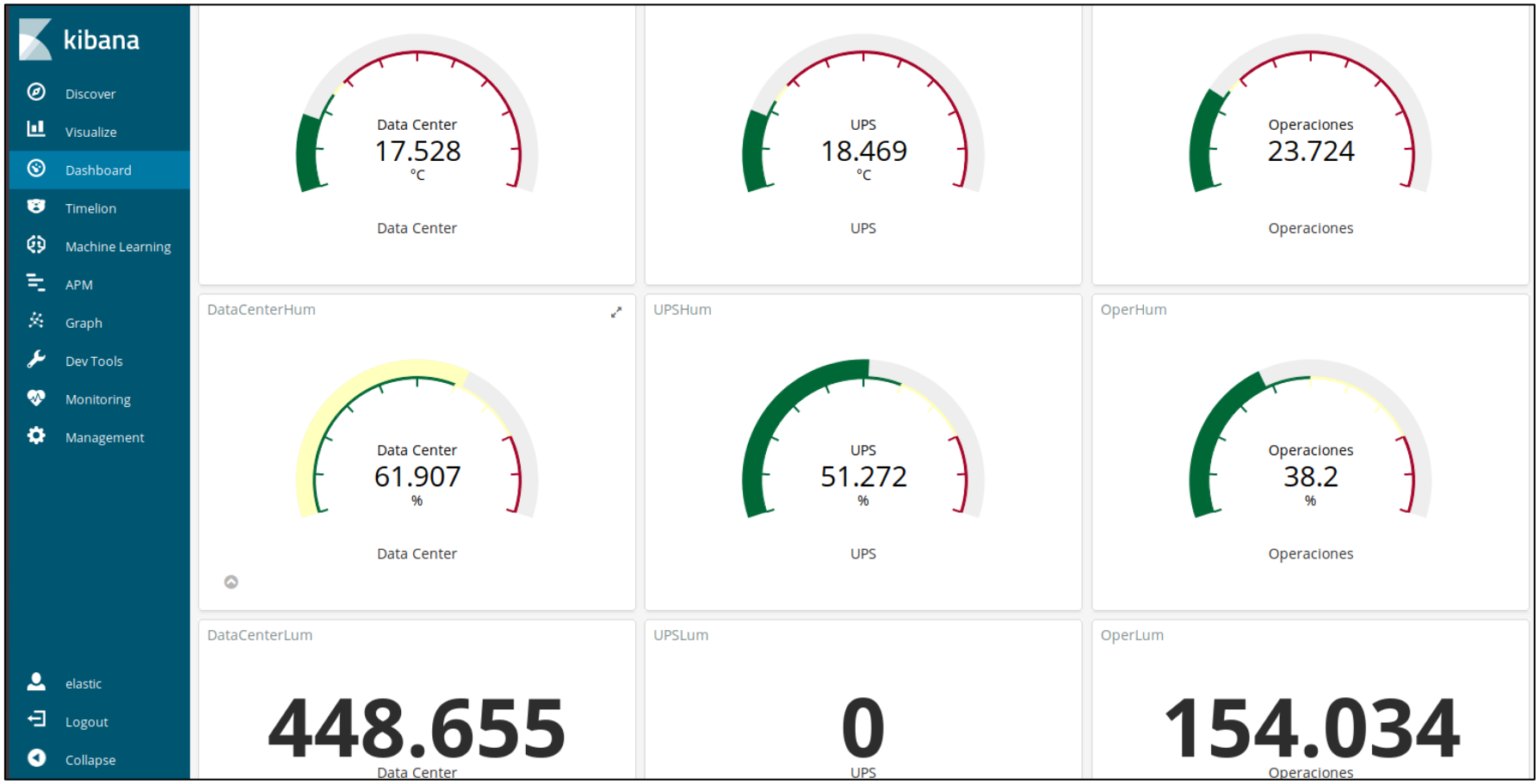


Figura 46. Presentación de información en tiempo real de temperatura, humedad relativa e iluminación. Fuente [El Autor]

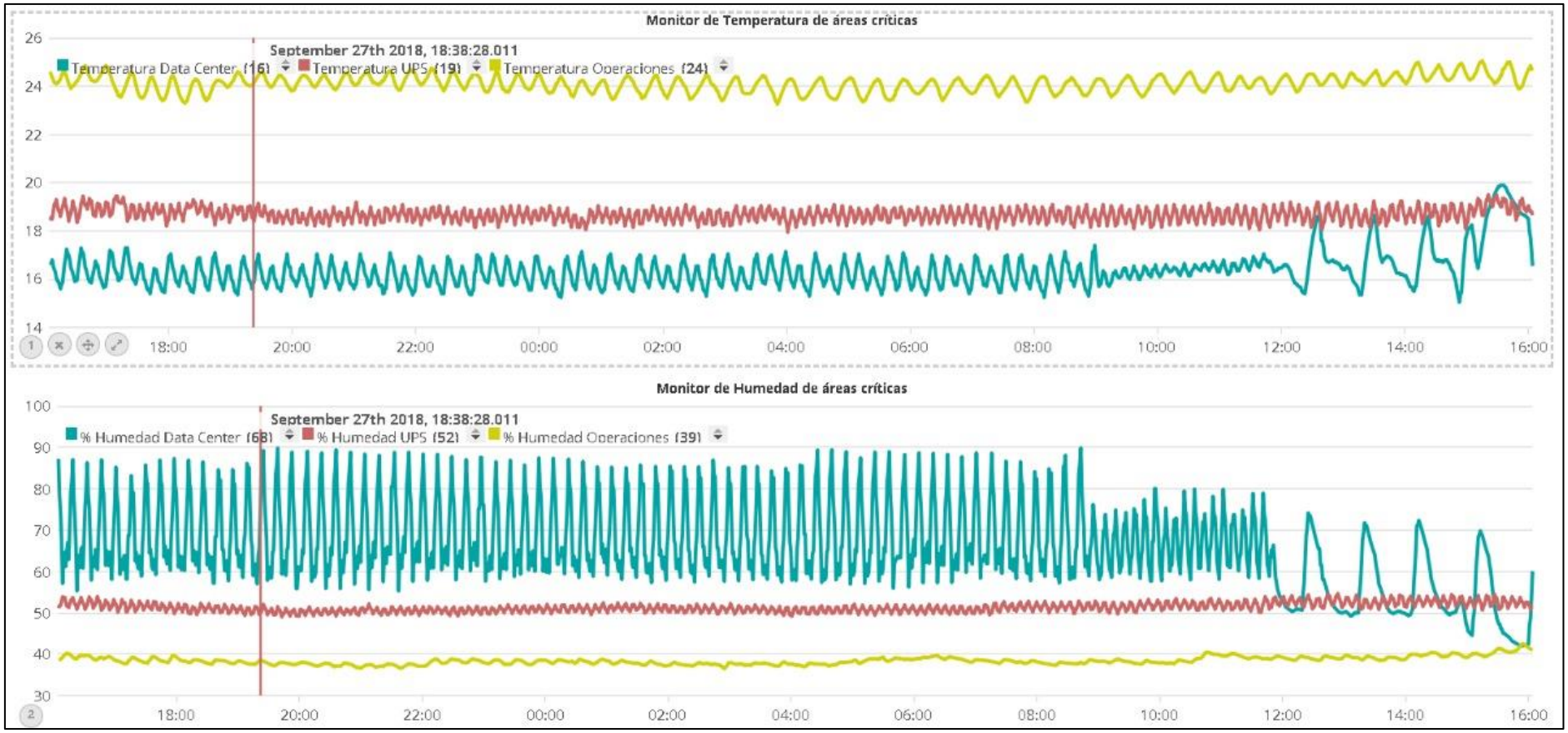


Figura 47. Monitoreo de la información mediante la herramienta Kibana. Fuente [El Autor]

5.1.6 Sistema de Alerta vía email

El formato de email que se envía al personal técnico del edificio SIS ECU-911 Loja se puede verificar en la Figura 48. Se detalla el nivel de temperatura superado y el área desde donde se origina la información. La alerta se envía en el menor tiempo posible, cercano al tiempo real, pero depende de la agilidad del personal técnico tomar las respectivas acciones.

Figura 48. Email de alerta de nivel de peligro de la variable de temperatura. Fuente [El Autor]

5.2 Diagrama de operación de Sistema de Monitoreo de variables ambientales.

En términos generales, el sistema de monitoreo desarrollado tiene los siguientes componentes operativos (Figura 49):

1) Adquisición de datos
2) Transporte de datos
3) Almacenamiento de información
4) Gestión de información
5) Presentación de información y monitoreo - Sistema de alerta.

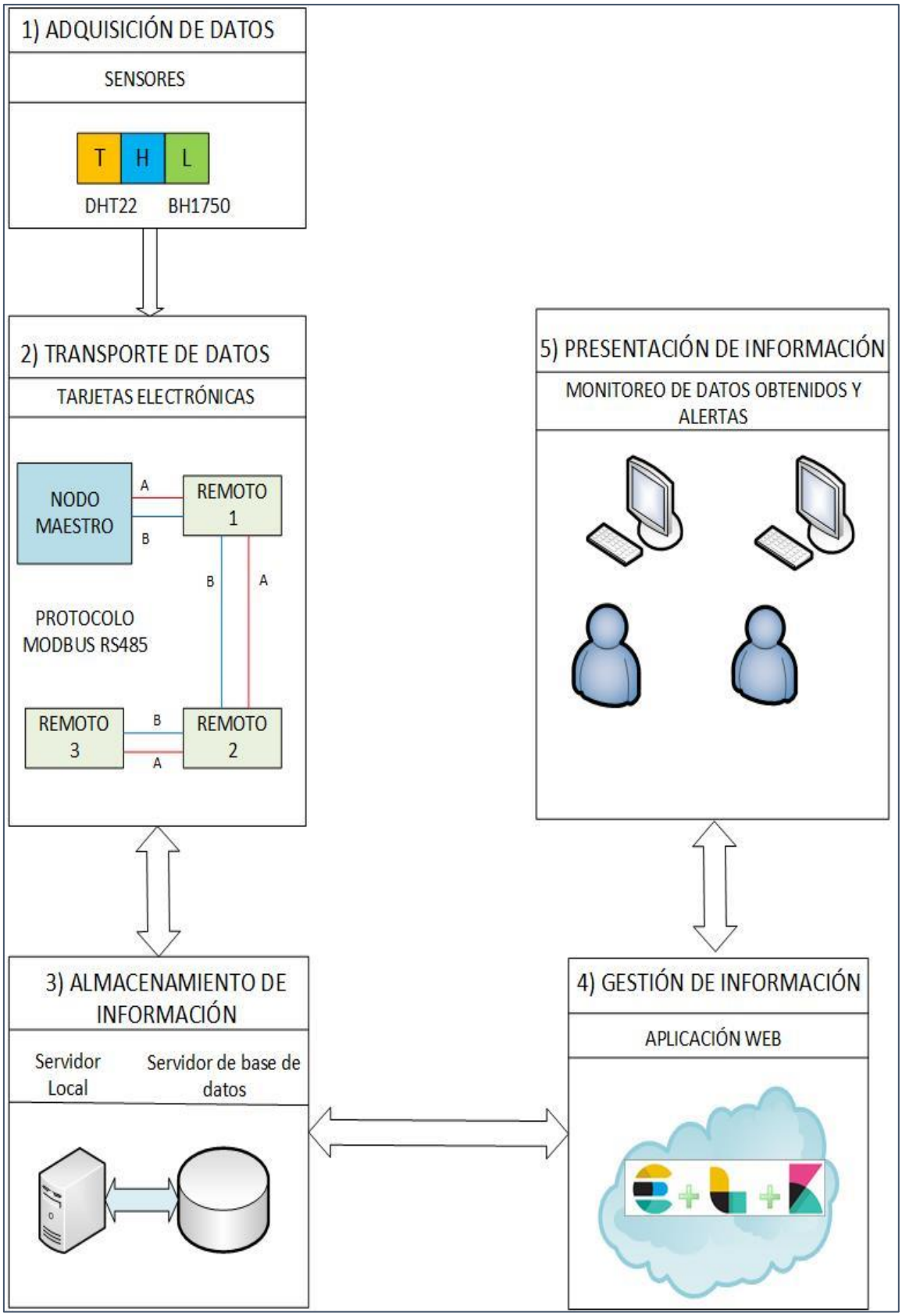


Figura 49. Diagrama de operación de sistema de monitoreo de variables ambientales. Fuente [El Autor]

6. Discusión.

El desarrollo de sistemas de monitoreo basados en hardware y software Arduino son fuentes generadoras de conocimiento, tanto a nivel básico como maestrante (Cabarcas, Montoya, Reyes Betancourt, & Arrieta, 2017). Es una de las razones por lo que se utilizó tarjetas electrónicas Arduino para el diseño de la estructura básica del sistema de monitoreo en tiempo real de variables ambientales de áreas críticas en el edificio SIS ECU-911 Loja. El sistema de adquisición y transporte de datos propuesto está en su mayoría desarrollado con estas tecnologías.

La tarjeta electrónica Arduino Mega 2560 para el nodo maestro fue seleccionada debido a que tiene el microcontrolador más potente de las tarjetas de la familia Arduino (ver 3.1.4.1) y además es de bajo costo, lo cual también representa un incentivo para su utilización. Por otro lado, el Arduino nano usado en los nodos remotos, tiene características (apartado 3.1.4.1) útiles y suficientes para las tareas de adquisición y envío de datos desde los sensores. El empleo de estos en lugar de otras alternativas estuvo condicionada específicamente a su bajo costo. Además, estas dos tarjetas mencionadas, posibilitan gran ahorro de tiempo en cuanto a desarrollo de la programación para el diseño y funcionamiento del sistema de monitoreo.

El entorno industrial para el cual fue diseñado el sistema, demanda seguridad en el transporte de datos e información para proteger el monitoreo de ruidos e interferencia que producen otros equipos activos en las mismas áreas donde se ubican los diferentes módulos que forman parte de la red propuesta. La red alámbrica implementada genera alto rendimiento y seguridad en lo relacionado a la comunicación de dispositivos y confiabilidad de los datos que se transmite.

El proyecto consideró el desarrollo de un sistema que sea escalable y adaptable, por lo cual se optó por el protocolo de red alámbrica MODBUS RS-485. Este protocolo es una técnica de centralización de datos, permite una determinada escalabilidad de nodos remotos sobre los cuales ejerce control un solo nodo maestro. La escalabilidad de este sistema requiere la modificación a nivel de programación tanto en Arduino como en Python para la adquisición y transporte de datos; y en cuanto a almacenamiento y presentación de información para monitoreo, se debe manipular PostgreSQL y la Pila

Elastic. La sincronización de los nuevos dispositivos que se agreguen en la red alámbrica con el Nodo Maestro y los Nodos Remotos debe contemplar modificaciones en los scripts que están operando actualmente. El aparatado 3.2 de este documento contiene información resumida y útil para entender el protocolo MODBUS para la escalabilidad del sistema desarrollado. Sin la correcta comprensión del protocolo, la sincronización de los nuevos dispositivos que conformen la red alámbrica podría ser más compleja.

Una vulnerabilidad que presenta este tipo de red centralizada, es que puede colapsar todo el sistema si el Nodo Maestro deja de funcionar. Los datos que se capturen en los nodos remotos se perderán debido a que no se procesan y almacenan en ninguna otra parte del sistema de monitoreo. El sistema está diseñado modularmente, por lo que se puede reemplazar los dispositivos averiados que son parte de la estructura (MAX485 o tarjeta Arduino Mega principalmente). Los nuevos dispositivos de reemplazo deben ser configurados de forma idéntica a los que están en funcionamiento. Esto se puede realizar mediante el código fuente desarrollado en Arduino y siguiendo los procesos detallados en el presente estudio en el apartado 4.2.3

Además de las prestaciones que ofrece por sí mismo el Sistema Gestor de Base de datos PostgreSQL (descritas en el apartado 3.3.3.2), fue usado en la configuración del sistema de monitoreo debido a que ya está instalado y en funcionamiento en el servidor local provisto por la institución. Más allá de eso, otro beneficio que ofrece PostgreSQL es que es perfectamente compatible con la pila Elastic. Al tratarse de un software código abierto no existe soporte oficial en Ecuador, dejando la sola posibilidad de encontrar soporte de terceros (por ejemplo, 2ndQuadrant) o desde la comunidad de usuarios (PostgreSQL, 2001-2018).

Las ventajas de emplear la pila Elastic, para la gestión de información del sistema propuesto, son su bajo costo y facilidad para desarrollo o adaptación de proyectos a las necesidades específicas de esta propuesta. El bajo costo está relacionado al tipo de licencia código abierto que tienen las herramientas que la conforman. Su versión básica tiene licencia cuya renovación anual se puede realizar sin costo, lo que deja a consideración solo el costo de aprendizaje. En referencia al fácil desarrollo con estas herramientas, Elastic posee plantillas con un frente de trabajo gráfico previamente

establecido lo cual reduce el tiempo de desarrollo de la interfaz web e interacción de la misma con PostgreSQL.

La versión básica de la pila Elastic solo permite vincular una base de datos de origen, por lo tanto, un solo proyecto puede ser monitoreado por Logstash (ver apartado 3.3.4.1). Para la escalabilidad horizontal (agregar proyectos, bases de datos, entre otros) se requiere asumir un costo debido a la versión de paga. Otro inconveniente de la versión básica es el alto costo en cuanto a recursos del servidor (máquina) que utiliza. Esto debe ser considerado seriamente en el momento del diseño del sistema de monitoreo.

La instalación y configuración de las herramientas de la pila Elastic requiere de un conocimiento intermedio o alto en gestión de sistemas informáticos, debido a que cada componente debe instalarse por separado. Para facilitar el desarrollo de este sistema, se recurrió al servidor proporcionado por personal del edificio ECU-911 Loja y bajo supervisión se instaló y configuró la pila Elastic donde se procesa la información para la presentación en el interfaz web.

Debido a que el número de sensores y variables ambientales son relativamente pocos, el dashboard de la aplicación web permite fácilmente dar seguimiento a los eventos en las diferentes áreas de interés. En un posible escenario de escalabilidad se debería considerar una sistematización de información más precisa y con parámetros visuales óptimos que faciliten la interpretación por parte del usuario.

El proyecto no contempló el desarrollo de sistemas automáticos de controladores y actuadores, por lo tanto, la respuesta ante una alerta debe ser realizada por el personal de turno. Como un valor agregado de este proyecto, se implementó un procedimiento de detección y comunicación de eventos de riesgo mediante alertas de temperatura. Las alertas se envían por correo electrónico cuando cualquiera de los sensores registra una temperatura superior a umbrales establecidos por el personal técnico del edificio SIS ECU-911 Loja (apartado 5.1.6). Este último componente, permite cerrar el ciclo de operaciones de un sistema de monitoreo en tiempo real: adquisición y transporte de datos, almacenamiento de información, gestión de información, presentación, monitoreo, y alerta temprana.

7. Conclusiones.

Los sensores digitales DHT22 y BH1750 utilizados para la captura de las variables ambientales de temperatura, humedad e iluminación de las áreas críticas en el edificio del SIS ECU-911 Loja, cumplen los requisitos mínimos para asegurar estabilidad y vigilancia continua. Para la selección de los sensores, se analizaron las características operativas y especificaciones técnicas de los dispositivos, además, se contrastó que las prestaciones de los sensores seleccionados cumplen con los requisitos para el sistema de monitoreo.

El diseño e implementación de red alámbrica centralizada de sensores utilizando el protocolo MODBUS RS-485, permite la comunicación del Nodo Maestro con los tres Nodos Remotos y el envío confiable de información. El uso del protocolo MODBUS posibilita el alcance de los propósitos planteados para este proyecto, y además deja abierta la posibilidad para que pueda ser ampliado fácilmente a otras áreas del edificio del SIS ECU-911 Loja.

El panel informativo creado para la presentación de datos vía web, expone las métricas capturadas por los sensores de las variables ambientales de humedad, temperatura e iluminación en un entorno gráfico, descriptivo y de fácil interpretación visual. La indización de los datos para consulta y análisis en la web, se logró utilizando las herramientas informáticas de la pila Elastic, encargadas de recopilar los datos almacenados en PostgreSQL.

Se implementó el sistema propuesto para el monitoreo de variables ambientales en el edificio SIS ECU-911 Loja. El sistema se encuentra en completo funcionamiento y presenta información en tiempo real sobre las condiciones ambientales de las áreas de interés. La información obtenida también puede ser útil para otras aplicaciones y líneas de investigación que dependen de datos de temperatura, humedad relativa e iluminación.

Con el sistema en funcionamiento se ha cumplido con los objetivos propuestos en la investigación y la aprobación emitida mediante certificación de validez otorgado por la institución en la que se intervino.

8. Recomendaciones.

- Los scripts desarrollados en los diferentes entornos de programación y utilizados en el desarrollo del sistema de monitoreo requieren de un nivel de conocimiento medio de programación en C++ y Python, por lo que se recomienda incrementar las habilidades en programación para un mejor entendimiento de esta propuesta.
- El conocimiento del lugar (interno o externo) donde se pretende implementar los componentes del sistema de monitoreo es importante, porque permite seleccionar con mayor precisión los dispositivos (sensores) para la captura de señales de las variables y el medio físico (alámbrico o inalámbrico) para el transporte de los datos.
- Se recomienda explorar estrategias de control automático como mejoras al sistema implementado para responder de manera inmediata frente a los riesgos detectados, para cuando el personal de turno no esté disponible inmediatamente. Además, se pueden implementar alertas locales de tipo visual, sonora o por pantalla que enfoque la atención del técnico para acelerar su actuación.
- Las herramientas utilizadas para el monitoreo en la web tienen un alto consumo de recursos del servidor donde está instalado. Por lo cual se recomienda usar un servidor que cumpla con los requerimientos mínimos de hardware y software para todos los procesos del sistema propuesto (ver apartado 11.6.4, Tabla 15).
- La alta disponibilidad que pueda tener el sistema de monitoreo depende en su mayoría del diseño de un esquema de redundancia 1:1 de los dispositivos de hardware (sensores, tarjetas Arduino, fuente reguladora, cable de par trenzado) (explicados en los apartados 3.1, 3.2 y 4.2), y que puedan conectarse vía USB al mismo servidor local. En cuanto a la intervención en la programación, el script desarrollado en Python deberá ejecutar sentencias que indiquen la lectura de datos del puerto serial de Arduino (puerto redundante) cuando no se encuentren datos en el puerto al que se conecta el sistema de monitoreo (principal).
- Se recomienda que el personal técnico que labora en el edificio SIS ECU-911 LOJA tenga bastos conocimientos acerca de manejo de bases de datos para solventar las limitaciones (particiones, campos operativos, entre otros) de PostgreSQL cuando la base de datos sea muy grande.

9. Líneas de investigación futura.

Algunas de las líneas de investigación que pueden desarrollarse en un futuro proyecto se exponen a continuación:

- Se podría estudiar una mejora al sistema de monitoreo en cuanto a la acción y control de dispositivos periféricos (sistema SCADA) para poder realizar una acción correctiva frente a un evento de peligro registrado. La mejora debería contemplar el análisis de los componentes electrónicos de la red y un posible cambio en el modo de acceso al bus de datos de los dispositivos que componen el sistema.

- Existe la posibilidad de utilizar la información de las condiciones ambientales de las áreas críticas como temperatura, humedad e iluminación para realizar un análisis estadístico y conocer cómo afectan a la salud y confort de las personas, así como al funcionamiento y permanencia de los equipos en las áreas críticas del edificio SIS ECU-911 Loja. Dentro de esta línea de investigación se debe considerar la escalabilidad del sistema de monitoreo para obtener información de las señales ambientales en distintos espacios dentro de las mismas áreas de intervención.

10.Bibliografía

[1] Abraham Silberschatz, H. F. (2002). *Fundamentos de bases de datos.* Madrid: Mc Graw Hill.

[2] Academy, C. N. (08 de 12 de 2018). *Introducción a Internet del Todo*. Obtenido de https://static-course-assets.s3.amazonaws.com/IoE11/ES/index.html

[3] Andrés, C. H. (10 de 11 de 2011). *Automatización Avanzada (37800) Máster en Automática y Robótica*. Obtenido de https://rua.ua.es/dspace/bitstream/10045/18990/1/AA-p3.pdf

[4] Aprendiendo Arduino. (2016). *Curso Arduino Teinnova*. Obtenido de http://teinnova.aprendiendoarduino.com/2016/07/24/modbus/

[5] ARDUINO . (2018). *Arduino Web Editor*. Obtenido de https://www.arduino.cc/en/Main/Software?

[6] Arduino & Geniuno Products. (2018). *Arduino & Geniuno Products*. Obtenido de https://www.arduino.cc/en/Main/arduinoBoardMega2560/

[6] Ayúz, J. M. (2018). *Aprendiendo Arduino*. Obtenido de https://aprendiendoarduino.wordpress.com/category/modbus/

[7] Cabarcas, A., Montoya, J. A., Reyes Betancourt, D. E., & Arrieta, C. (2017). *Sistema de medición automatizada de variables ambientales para agricultura de precisión con software libre.* Cartagena: Universidad de Cartagena.

[8] Camargo Vega, J. J. (2015). Conociendo big data. *Facultad de Ingeniería*, 63-77.

[9] Candelas-Heredias, F. (2011). *Comunicación con RS_485 y MODBUS.* Obtenido de https://rua.ua.es/dspace/bitstream/10045/18990/1/AA-p3.pdf

[10] Capote, O. P., & Ruiz, N. M. (2008). *Introducción a los sistemas de bases de datos.* Málaga: Paraninfo.

[11] Cárdenas Esparza, R. (2013). *Monitoreo de procesos industriales con sensores inteligentes.* Tijuana, BC., México.

[12] Celaya, C. L. (2007). Uso de software libre y de Internet como herramientas de apoyo para el aprendizaje. . *Revista de educacióna distancia*, 83 - 100.

[13] Chambi, J. &. (2006). *Comparación entre MySQL y PostgreSQL* . Cuenca.

conecta, W. (2018). *Interfaz RS232*. Obtenido de https://www.wut.de/e-8wwww-16-apes-000.php

[14] Date, C. (2001). *Introducción a los sistemas de bases de datos.* México: PEARSON EDUCACIÓN.

[15] Elasticsearch . (2018). *Elastic*. Obtenido de https://www.elastic.co/use-cases

[16] ELECTRONICLAB. (31 de 08 de 2018). *ELECTRONICLAB Ingeniería y Diseño Electrónico*. Obtenido de https://electronilab.co/tienda/sensor-de-temperatura-y-humedad-dht22/

[17] Espinoza, J. (2012). *POPENERP MEDICAL: MÓDULO DE FARMACIA PARA SISTEMA HOSPITALARIO.* Cuenca.

[18] Forgiarini Rupp, R., Giraldo Vásquez, N., & Lamberts, R. (2015). A review of human thermal comfort in the built environment. *Energy and Buildings*, 178-205.

[19] García, A. M. (2007). La promoción del uso de software libre por parte de las universidades. *Revista de educación a distancia*, 17-19.

[20] GitHub, Inc. (2018). *GitHub-postgres/postgres: Mirror of the official PostgreSQL* . Obtenido de https://github.com/postgres/postgres

[21] Gomez Blazquez, A. (2016). *Sistema sensor autònom sense fils de baix consum per a aplicacions industrials.* Catalunya.

[22] JET BRAINS. (2018). *The drive to develop Keep Evolving*. Obtenido de https://www.jetbrains.com/

[23] Kasián, F. &. (2012). *Búsquedas por similitud en PostgreSQL.* Obtenido de http://sedici.unlp.edu.ar/bitstream/handle/10915/23754/Documento_completo.pdf%3Fsequence%3D1

[24] Liu, T. (s.f.). *SparkFun Electronics*. Obtenido de https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

[25] López Esquembri, J. (2017). *Hypoglycemia ergonomic detection.* Barcelona.

[26] Marta, R. L. (2015). *Instrumentación Electrónica con Microprocesadores II: Procesos Avanzados*. Obtenido de http://ocw.uc3m.es/tecnologia-electronica/sistemas-embebidos-basados-en-fpgas-para-instrumentacion/material-de-clase-1/acondicionamiento-digital

[27] Martinez, M. C. (2014). *Protocolos de comunicación en automatización industrial*. Obtenido de http://repositorio.upct.es/bitstream/handle/10317/4338/pfc5902.pdf;jsessionid=E307D9DC24D3DD2E3A07E26FD3C9D76E?sequence=1

[28] Mora, H. (2011). Sistemas de Adquisición y procesamiento de datos. *Fundamentos de los Computadores*, 25-26.

[29] Novella Latorre, J. (2012). *Sistema de Gestión de Base de datos PostgreSQL.*

[30] Olaya, A. L. (2011). Implementación de una Red MODBUS/TCP. . *Ingeniería y Competitividad*, 35-44.

[31] Pallás Areny, R. (2004). *Sensores y Acondicionadores de señal.* Barcelona: Marcombo.

[32] PostgreSQL, 2. (2001-2018). *2ndQuadrant PostgreSQL*. Obtenido de https://www.2ndquadrant.com/es/

[33] Prado Ventura, C. (2016). *Interfaz de visualización de logs en un sistema de información para la gestión de datos en investigación clínica.* . Madrid.

[34] Python Software Fundation. (2018). Obtenido de https://www.python.org/psf/

[35] ROHM Co., Ltd. (2011). *Mouser Electronics*. Obtenido de http://www.mouser.com/ds/2/348/bh1750fvi-e-186247.pdf

[36] Samboya, N. (2012). Normas de comunicación en Serie: RS-232, RS-422 y RS-485. *Ingenio Libre*, 86-94.

[37] Santacreu Grifol, A. (2016). *Integración Big Data en el Mundo empresarial: ELK-Hadoop-Splunk.* Barcelona.

[38] Soriano Miras, J. (2012). *Diseño de un extensor de entradas y salidas analógicas por MODBUS RTU sobre RS-485.* Cataluya.

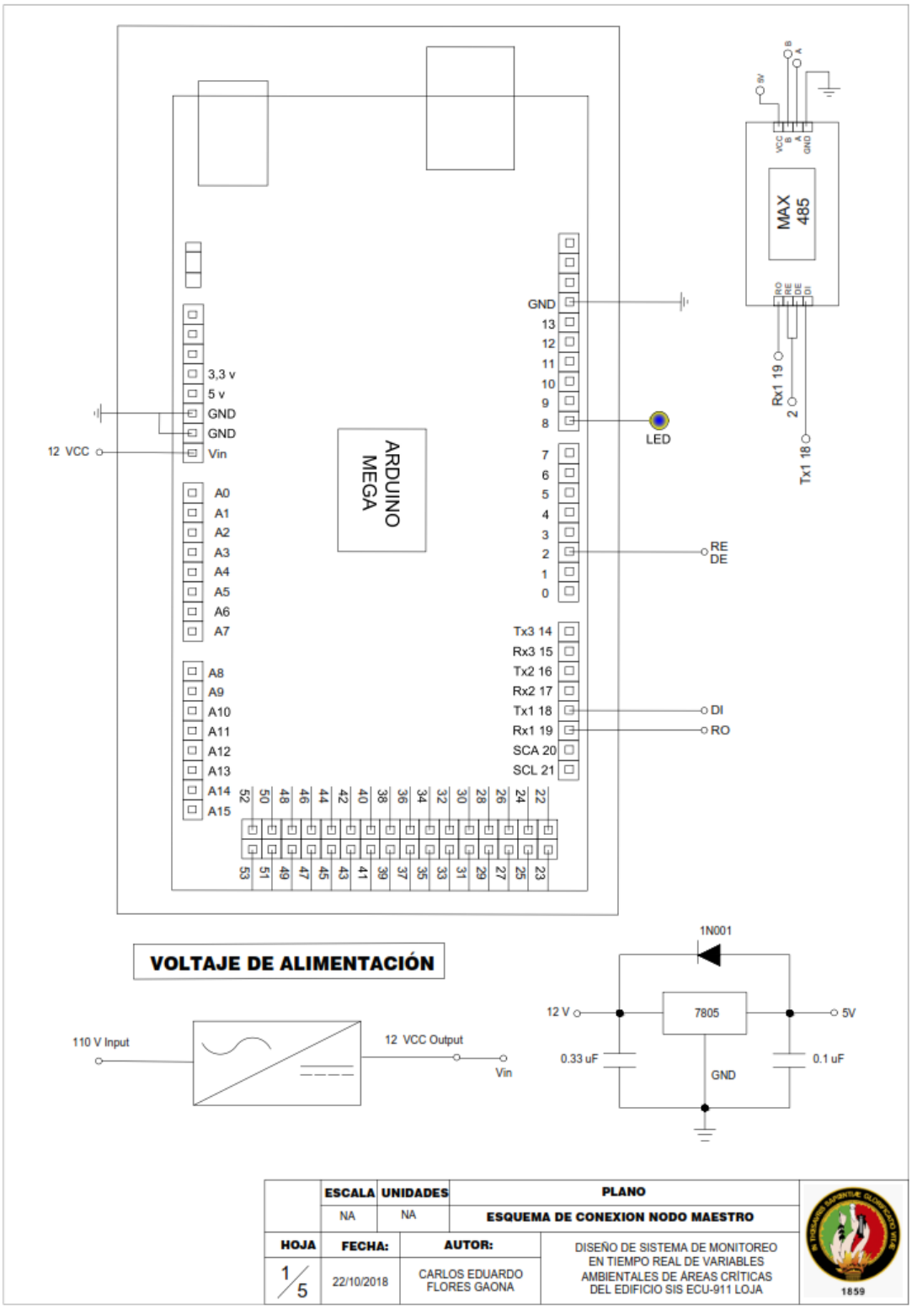
[39] Suárez Barón, J. C., & Suárez Barón, M. J. (2014). Monitoreo de variables ambientales en invernaderos usando tecnología Zigbee. *6º Congreso Argentino de AgroInformática*, (págs. 165-173). Bogotá.

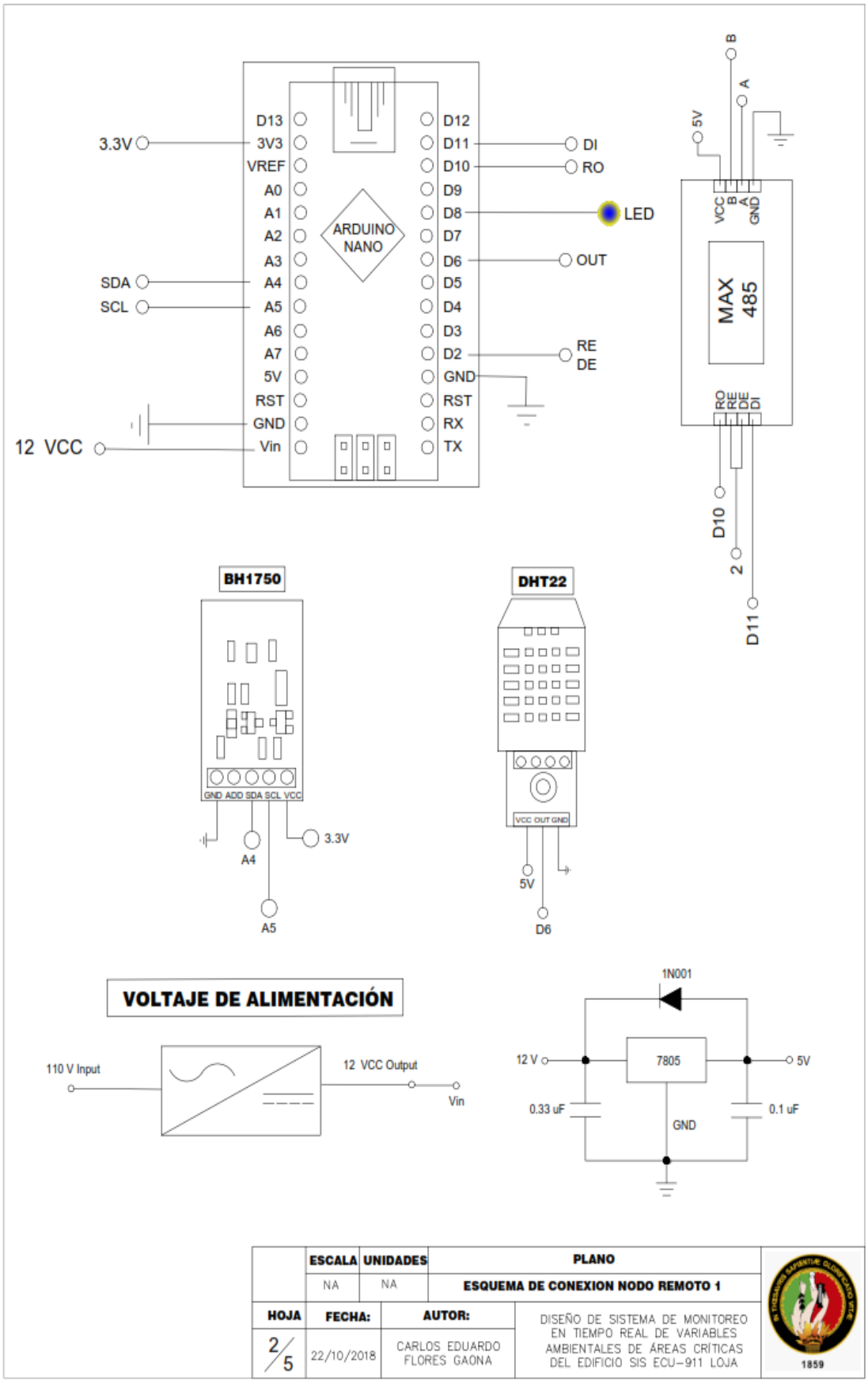
[40] Texas Instruments. (August de 2008). *The RS-485 Desing Guide*. Obtenido de https://www.google.com.ec/search?q=eia%2Ftia-485+standard&oq=EIA%2FTia+485&aqs=chrome.5.69i57j69i61j69i58j69i65j0l2.12098j0j7&sourceid=chrome&ie=UTF-8

[41] Tutoriales. (2018). *Naylamp Mechatronics*. Obtenido de https://naylampmechatronics.com/blog/37_Comunicaci%C3%B3n-RS485-con-Arduino.html

11.Anexos.

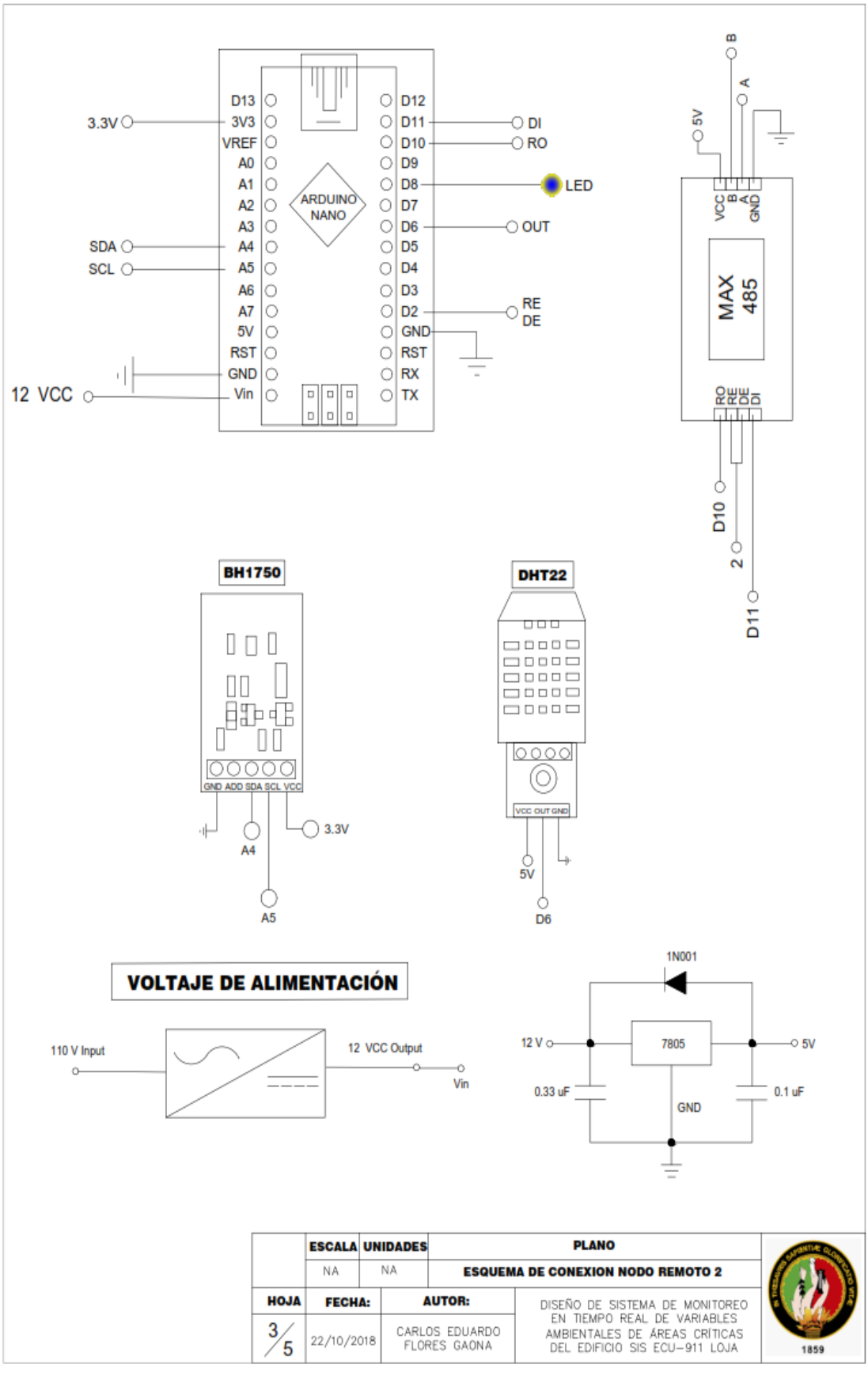
11.1 Planos de conexión de dispositivos para la adquisición/transmisión de los datos y esquema de conexión del bus de datos MODBUS RS-485.

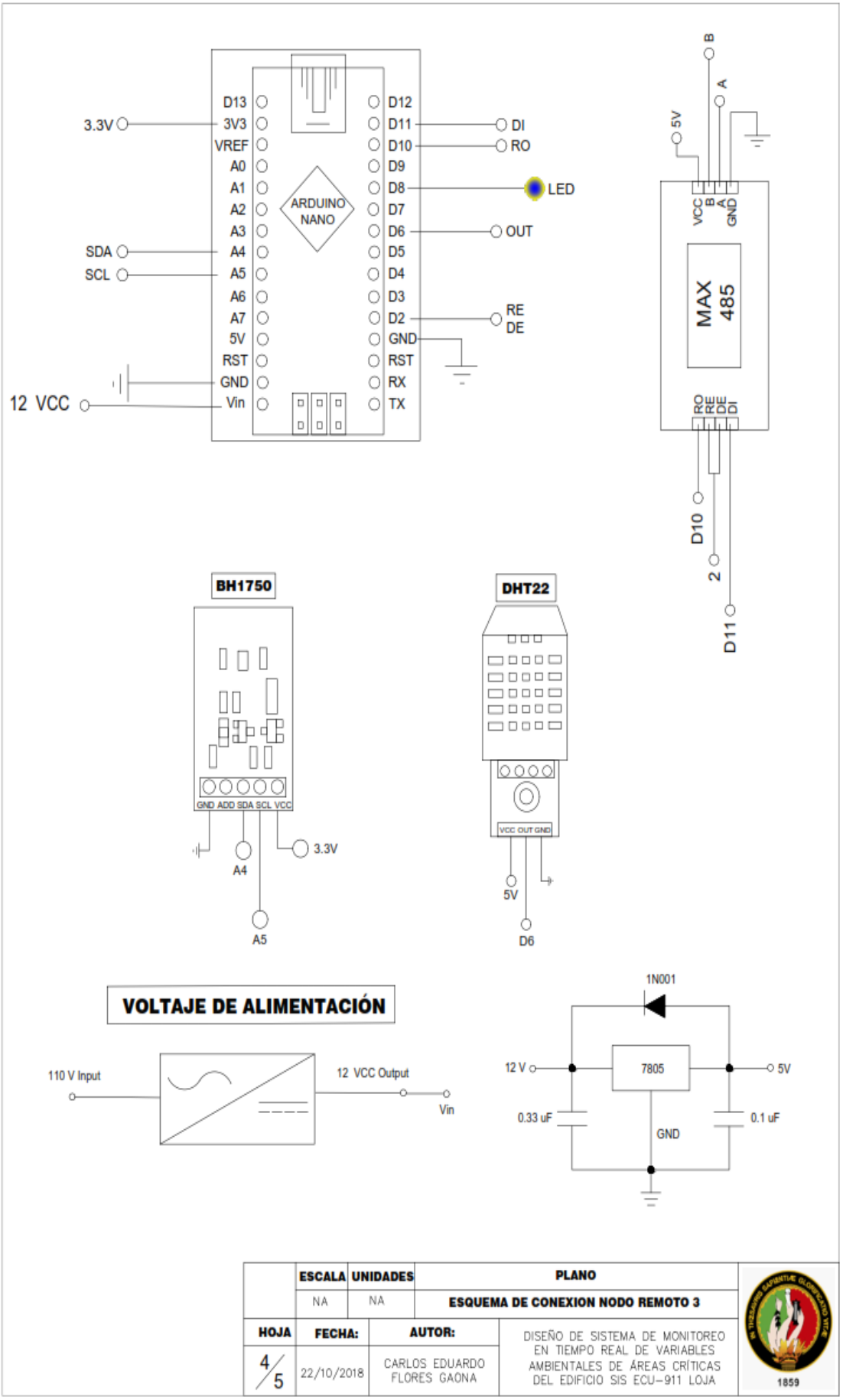




	ESCALA	UNIDADES	PLANO
	NA	NA	ESQUEMA DE CONEXION NODO REMOTO 1
HOJA	FECHA:	AUTOR:	DISEÑO DE SISTEMA DE MONITOREO EN TIEMPO REAL DE VARIABLES AMBIENTALES DE ÁREAS CRÍTICAS DEL EDIFICIO SIS ECU–911 LOJA
2/5	22/10/2018	CARLOS EDUARDO FLORES GAONA	

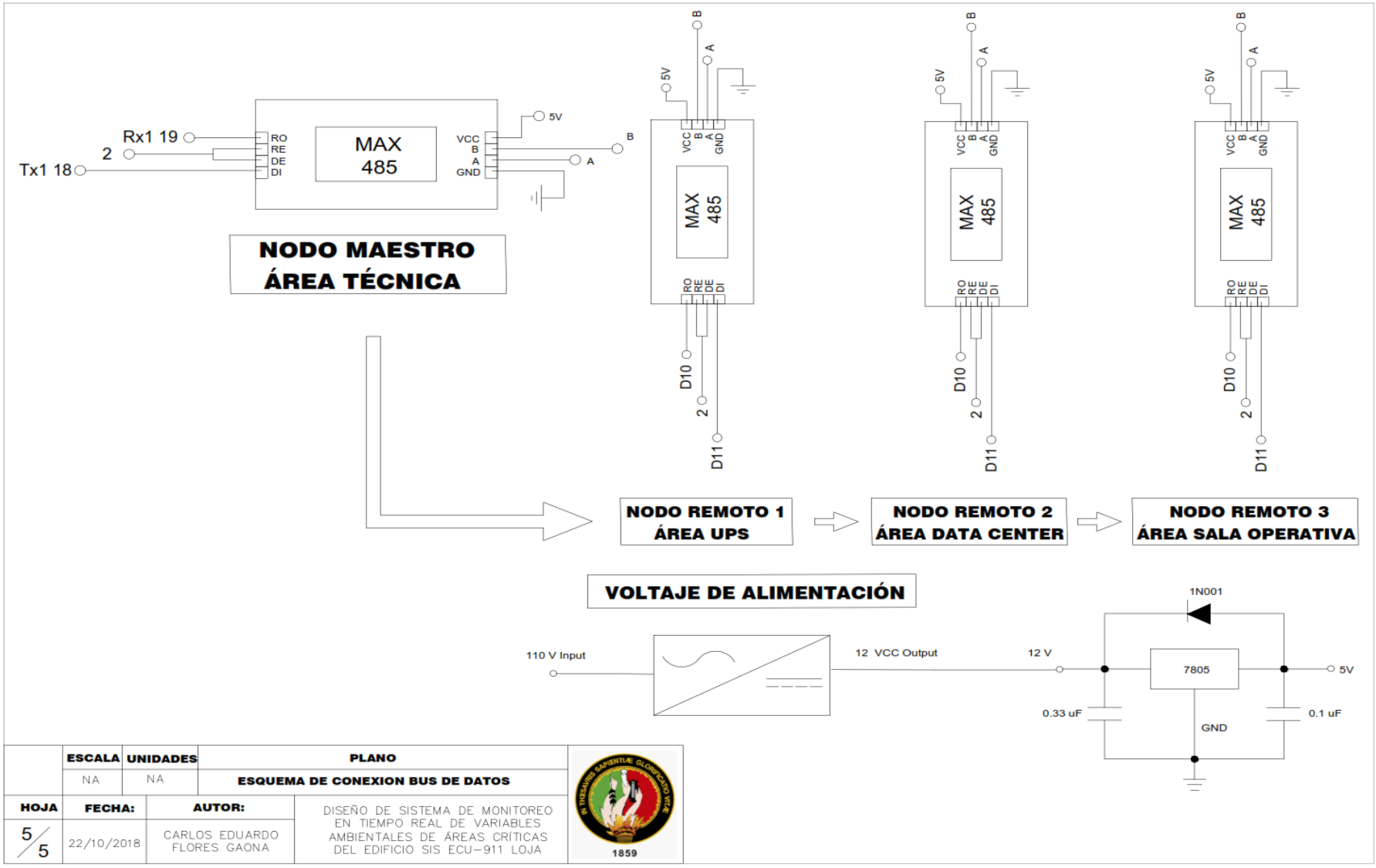
1859


3.3V
SDA
SCL
12 VCC
D13
3V3
VREF
A0
A1
A2
A3
A4
A5
A6
A7
5V
RST
GND
Vin
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
GND
RST
RX
TX
ARDUINO NANO
DI
RO
LED
OUT
RE
DE
B
A
5V
VCC
B
A
GND
MAX 485
RO
RE
DE
DI
D10
2
D11
BH1750
GND ADD SDA SCL VCC
3.3V
A4
A5
DHT22
VCC OUT GND
5V
D6
VOLTAJE DE ALIMENTACIÓN
110 V Input
12 VCC Output
Vin
1N001
12 V
7805
5V
0.33 uF
0.1 uF
GND
ESCALA
UNIDADES
PLANO
NA
NA
ESQUEMA DE CONEXION NODO REMOTO 2
HOJA
FECHA:
AUTOR:
3/5
22/10/2018
CARLOS EDUARDO FLORES GAONA
DISEÑO DE SISTEMA DE MONITOREO EN TIEMPO REAL DE VARIABLES AMBIENTALES DE ÁREAS CRÍTICAS DEL EDIFICIO SIS ECU-911 LOJA
1859



	ESCALA	UNIDADES	PLANO
	NA	NA	ESQUEMA DE CONEXION NODO REMOTO 3
HOJA	FECHA:	AUTOR:	DISEÑO DE SISTEMA DE MONITOREO EN TIEMPO REAL DE VARIABLES AMBIENTALES DE ÁREAS CRÍTICAS DEL EDIFICIO SIS ECU-911 LOJA
4/5	22/10/2018	CARLOS EDUARDO FLORES GAONA	

1859


Rx1 19
2
Tx1 18
RO
RE
DE
DI
MAX 485
VCC
B
A
GND
5V
B
A
NODO MAESTRO
ÁREA TÉCNICA
D10
2
D11
NODO REMOTO 1
ÁREA UPS
NODO REMOTO 2
ÁREA DATA CENTER
NODO REMOTO 3
ÁREA SALA OPERATIVA
VOLTAJE DE ALIMENTACIÓN
110 V Input
12 VCC Output
12 V
1N001
7805
5V
0.33 uF
0.1 uF
GND
ESCALA
NA
UNIDADES
NA
PLANO
ESQUEMA DE CONEXION BUS DE DATOS
HOJA
5/5
FECHA:
22/10/2018
AUTOR:
CARLOS EDUARDO FLORES GAONA
DISEÑO DE SISTEMA DE MONITOREO EN TIEMPO REAL DE VARIABLES AMBIENTALES DE ÁREAS CRÍTICAS DEL EDIFICIO SIS ECU-911 LOJA
1859

11.2 Anexo 2: Programación en software Arduino.

11.2.1 Programación de Nodo Maestro.

```
const int EnTxPin =  2; // HIGH:TX y LOW:RX
char cabecera;
char direccion[4];
byte h1;
byte h2;
byte h3;
byte h4;
byte t1;
byte t2;
byte t3;
byte t4;
byte l1;
byte l2;
byte l3;
byte l4;
char trailer;
int x = 1; //Contador para control de consultas
//Definimos estructuras union para manejar la lectura de los sensores
//Para la humedad
union union_h {
  byte byte_h[4];
  float float_h;
} h;
//Para la temperatura
union union_t {
  byte byte_t[4];
  float float_t;
} t;
//Para la luz
union union_l {
  byte byte_l[4];
  float float_l;
```

```
} l;
void setup()
{
  Serial.begin(9600);
  while(!Serial);
  Serial1.begin(9600);
  while(!Serial1);
  // Inicializamos el pin de control transmisor/receptor
  pinMode(EnTxPin, OUTPUT);
  digitalWrite(EnTxPin, HIGH); //RS485 como Transmisor
  pinMode(24, OUTPUT);//encender el led
}
 void loop()
{
  digitalWrite (24,HIGH);
//  if Serial1.available (){
  if (x == 1) {
      //---solicitamos una lectura del sensor----------
    Serial1.print('I'); //inicio de trama
    Serial1.print(int(101),DEC);//direccion del esclavo
    Serial1.print('H'); //H para indicarle que vamos a Leer el sensor
    Serial1.print('F'); //fin de trama
    Serial1.flush();
 }
  else if (x == 2) {
    Serial1.print('I'); //inicio de trama
    Serial1.print(int(102),DEC);//direccion del esclavo
    Serial1.print('H'); //H para indicarle que vamos a Leer el sensor
    Serial1.print('F'); //fin de trama
    Serial1.flush();
 }
  else if(x == 3) {
    Serial1.print('I'); //inicio de trama
    Serial1.print(int(103),DEC);//direccion del esclavo
    Serial1.print('H'); //H para indicarle que vamos a Leer el sensor
    Serial1.print('F'); //fin de trama
    Serial1.flush();
```

```
}
   //----Leemos la respuesta del Esclavo-----
 digitalWrite(EnTxPin, LOW); //RS485 como receptor
 delay(60);
 //Si existen bytes en el puerto de entrada
 if (Serial1.available() > 0) {
   while(Serial1.available() > 0){
   //Buscamos la cabecera de la trama recibida
   cabecera = Serial1.read();
   //Si la cabecera se encuentra entonces se lee el resto de la trama
   if (cabecera == 'i') {
     //Serial.print(cabecera);
     delay(3);
     direccion[0] = Serial1.read();
     //Serial.print(direccion[0]);
     delay(3);
     direccion[1] = Serial1.read();
     delay(3);
     direccion[2] = Serial1.read();
     delay(3);
     h1 = Serial1.read();
     delay(3);
     h2 = Serial1.read();
     delay(3);
     h3 = Serial1.read();
     delay(3);
     h4 = Serial1.read();
     delay(3);
     t1 = Serial1.read();
     delay(3);
     t2 = Serial1.read();
     delay(3);
     t3 = Serial1.read();
     delay(3);
     t4 = Serial1.read();
     delay(3);
     l1 = Serial1.read();
```

```
      delay(3);
      l2 = Serial1.read();
      delay(3);
      l3 = Serial1.read();
      delay(3);
      l4 = Serial1.read();
      delay(3);
      trailer = Serial1.read();
      //Comprobamos si los datos tienen la estrctura de una trama
      if (trailer == 'f') {
        //Si la tienen presentamos los datos de los sensores
//Convertimos los valores leídos por el puerto a unidades que se puedan
entender
        //Para la humedad
        h.byte_h[0] = h1;
        h.byte_h[1] = h2;
        h.byte_h[2] = h3;
        h.byte_h[3] = h4;
        //Para la temperatura
        t.byte_t[0] = t1;
        t.byte_t[1] = t2;
        t.byte_t[2] = t3;
        t.byte_t[3] = t4;
        //Para la luz
        l.byte_l[0] = l1;
        l.byte_l[1] = l2;
        l.byte_l[2] = l3;
        l.byte_l[3] = l4;

        if (atol(direccion) == 101) {
          Serial.print("S1, ");
          Serial.print(h.float_h,2);
          Serial.print(", ");
          Serial.print(t.float_t,2);
          Serial.print(", ");
          Serial.println(l.float_l,2);
        }
```

```
        else if (atol(direccion) == 102) {
          Serial.print("S2, ");
          Serial.print(h.float_h,2);
          Serial.print(", ");
          Serial.print(t.float_t,2);
          Serial.print(", ");
          Serial.println(l.float_l,2);
          }
        else if (atol(direccion) == 103) {
          Serial.print("S3, ");
          Serial.print(h.float_h,2);
          Serial.print(", ");
          Serial.print(t.float_t,2);
          Serial.print(", ");
          Serial.println(l.float_l,2);
        }
      }
      else {
        Serial.println("Datos recibidos con errores");
      }
    }
  }
  }
  digitalWrite(EnTxPin, HIGH); //RS485 como transmisor
  x = x+1;
  if (x == 4){
    x = 1;
  }
   digitalWrite(24, LOW);//apagar Led
delay(500);
}
```

11.2.2 Programación de Nodos Remotos.

- **Nodo Remoto 1: Área de UPS.**

```
#include <BH1750.h>//libreria para sensor de luz
#include <Wire.h>//libreria para comunicacion de sensores de luz
#include "DHT.h" //librería para sensores humedad y temperatura
#include <math.h>
#include <SoftwareSerial.h>
char cabecera;
char direccion[4];
char funcion;
char trailer;
long direccion_c;
SoftwareSerial mySerial(10, 11); // RX, TX
//Constantes para comunicacion RS485
const int EnTxPin =  2;  // HIGH:TX y LOW:RX
const int mydireccion = 101; //Direccion del esclavo
//se declaran variables para los sensores y los pines correspondientes
 de humedad y temperatura
DHT dht1(6, DHT22);
//Direccion de  sensor BH1750
byte u8_BH1750_address = 0x23; //Pin ADDR en nível lógico 0
//Instrucciones de modo lectura
byte mode_HighResolution_1 = 0x10; //Modo de alta precision con 1 lux
 de resolucion (120ms de tiempo de lectura)
//variables para sensores de luminancia
uint16_t sensor1_result = 0;
//Definimos estructuras union para dar a los datos de los sensores
//un formato apropiado para la transmisión
//Para la humedad
union h_tag {
  float float_h;
  byte byte_h[4];
} h;
//Para la temperatura
union t_tag {
  float float_t;
```

```
  byte byte_t[4];
} t;
//Para la luz
union l_tag {
  float float_l;
  byte byte_l[4];
} l;
void setup()
{
  mySerial.begin(9600);
  while(!mySerial);
  Serial.begin(9600);
  while(!Serial);
  //Configuramos el pin de control de transmisión/recepción
  pinMode(EnTxPin, OUTPUT);
  digitalWrite(EnTxPin, LOW); //RS485 como receptor
  Wire.begin();//Inicilizar el bus I2C, la biblioteca BH1750 no lo hace
 automaticamente
  dht1.begin();//se inicializa los sensores de humedad y temperatura
  pinMode(8, OUTPUT);//encender el led
}
void loop(){
  digitalWrite (8,HIGH);
  //Si existen datos en la entrada del puerto serial
  if(mySerial.available() > 0){
    while(mySerial.available() > 0){
      //Buscamos la cabecera de la trama
    cabecera = mySerial.read();
    if (cabecera == 'I'){
      //Si la encontramos, leemos el resto de la trama
      delay(3);
      direccion[0] = mySerial.read();
      delay(3);
      direccion[1] = mySerial.read();
      delay(3);
      direccion[2] = mySerial.read();
      delay(3);
```

```
      funcion = mySerial.read();
      delay(3);
      trailer = mySerial.read();
//Comprobamos si los datos recibidos tienen estructura de una trama
      if (trailer == 'F') {
        direccion_c = atol(direccion);
//Si la dirección de la trama es igual a la del dispostivo se envían
datos
        if (direccion_c == mydireccion){
          digitalWrite(EnTxPin, HIGH); //RS485 como transmisor
          //Leemos los sensores de humedad, temperatura y luz
          float h1 = dht1.readHumidity(); //variable humedad
          float t1 = dht1.readTemperature();//variable temperatura
          read_sensor1();
          int lectura = sensor1_result; //realizamos  la lectura del
 sensor     luz
          if (isnan(h1) || isnan(t1)) {
            h1=-1.0;//error obteniendo datos de humedad
            t1=-1.0;//error obteniendo datos de temperatura
          }
          //Damos a los datos un formato apropiado para la transmisión
          h.float_h = h1;
          t.float_t = t1;
          l.float_l = float(lectura);
          mySerial.print('i'); //inicio de trama
          mySerial.print(mydireccion,DEC); //direccion
          mySerial.write(h.byte_h, 4); //valor del sensor
          mySerial.write(t.byte_t, 4);
          mySerial.write(l.byte_l, 4);
          mySerial.print('f'); //fin de trama
          mySerial.flush(); //Esperamos hasta que se envíen los datos
          digitalWrite(EnTxPin, LOW); //RS485 como receptor
          Serial.print(cabecera);
          Serial.print(direccion[0]);
          Serial.print(direccion[1]);
          Serial.print(direccion[2]);
          Serial.print(funcion);
```

```
          Serial.println(trailer);
        }
     }
      else {
        Serial.println("Trama recibida con errores");
      }
    }
   }
  }
  digitalWrite(8, LOW);//apagar Led
}
//Funciones para sensores de luminancia
void read_sensor1(void){
   //Inicia a comunicacion I2C en la direccion selecionada
  Wire.beginTransmission(u8_BH1750_address);
  //Configuracion del sensor para el modo de operacion
  Wire.write(mode_HighResolution_1);
  //Finaliza la comunicacion
  Wire.endTransmission();
   //peticion delectura del sensor, esperando 2 bytes de respuesta
  Wire.requestFrom(u8_BH1750_address, 2); //
   //Espera la llegada de  dos 2 bytes
  if (2 <= Wire.available()) {
    sensor1_result = Wire.read();
    sensor1_result = sensor1_result << 8;
    sensor1_result |= Wire.read();
  }else{
      sensor1_result=-1;
  }
}
```

- **Nodo Remoto 2: Área de DATACENTER.**

```
#include <BH1750.h>//libreria para sensor de luz
#include <Wire.h>//libreria para comunicacion de sensores de luz
#include "DHT.h" //librería para sensores humedad y temperatura
#include <math.h>
#include <SoftwareSerial.h>
char cabecera;
char direccion[4];
char funcion;
char trailer;
long direccion_c;
SoftwareSerial mySerial(10, 11); // RX, TX
//Constantes para comunicacion RS485
const int EnTxPin =  2;  // HIGH:TX y LOW:RX
const int mydireccion = 102; //Direccion del esclavo
//se declaran variables para los sensores y los pines correspondientes
 de humedad y temperatura
DHT dht1(6, DHT22);
//Direccion de  sensor BH1750
byte u8_BH1750_address = 0x23; //Pin ADDR en nível lógico 0
//Instrucciones de modo lectura
byte mode_HighResolution_1 = 0x10; //Modo de alta precision con 1 lux
 de resolucion (120ms de tiempo de lectura)
//variables para sensores de luminancia
uint16_t sensor1_result = 0;
//Definimos estructuras union para dar a los datos de los sensores
//un formato apropiado para la transmisión
//Para la humedad
union h_tag {
  float float_h;
  byte byte_h[4];
} h;
//Para la temperatura
union t_tag {
  float float_t;
  byte byte_t[4];
```

```
} t;
//Para la luz
union l_tag {
  float float_l;
  byte byte_l[4];
} l;
void setup()
{
  mySerial.begin(9600);
  while(!mySerial);
  Serial.begin(9600);
  while(!Serial);
  //Configuramos el pin de control de transmisión/recepción
  pinMode(EnTxPin, OUTPUT);
  digitalWrite(EnTxPin, LOW); //RS485 como receptor
  Wire.begin();//Inicilizar el bus I2C, la biblioteca BH1750 no lo hace
 automaticamente
  dht1.begin();//se inicializa los sensores de humedad y temperatura
  pinMode(8, OUTPUT);//encender el led
}
void loop(){
  digitalWrite (8,HIGH);
  //Si existen datos en la entrada del puerto serial
  if(mySerial.available() > 0){
    while(mySerial.available() > 0){
      //Buscamos la cabecera de la trama
    cabecera = mySerial.read();
    if (cabecera == 'I'){
      //Si la encontramos, leemos el resto de la trama
      delay(3);
      direccion[0] = mySerial.read();
      delay(3);
      direccion[1] = mySerial.read();
      delay(3);
      direccion[2] = mySerial.read();
      delay(3);
      funcion = mySerial.read();
```

```
      delay(3);
      trailer = mySerial.read();
//Comprobamos si los datos recibidos tienen estructura de una trama
      if (trailer == 'F') {
        direccion_c = atol(direccion);
//Si la dirección de la trama es igual a la del dispostivo se envían
datos
        if (direccion_c == mydireccion){
          digitalWrite(EnTxPin, HIGH); //RS485 como transmisor
          //Leemos los sensores de humedad, temperatura y luz
          float h1 = dht1.readHumidity(); //variable humedad
          float t1 = dht1.readTemperature();//variable temperatura
          read_sensor1();
          int lectura = sensor1_result; //realizamos  la lectura del
 sensor     luz
          if (isnan(h1) || isnan(t1)) {
            h1=-1.0;//error obteniendo datos de humedad
            t1=-1.0;//error obteniendo datos de temperatura
          }
          //Damos a los datos un formato apropiado para la transmisión
          h.float_h = h1;
          t.float_t = t1;
          l.float_l = float(lectura);
          mySerial.print('i'); //inicio de trama
          mySerial.print(mydireccion,DEC); //direccion
          mySerial.write(h.byte_h, 4); //valor del sensor
          mySerial.write(t.byte_t, 4);
          mySerial.write(l.byte_l, 4);
          mySerial.print('f'); //fin de trama
          mySerial.flush(); //Esperamos hasta que se envíen los datos
          digitalWrite(EnTxPin, LOW); //RS485 como receptor
          Serial.print(cabecera);
          Serial.print(direccion[0]);
          Serial.print(direccion[1]);
          Serial.print(direccion[2]);
          Serial.print(funcion);
          Serial.println(trailer);
```

```
        }
     }
      else {
        Serial.println("Trama recibida con errores");
      }
    }
   }
  }
  digitalWrite(8, LOW);//apagar Led
}
//Funciones para sensores de luminancia
void read_sensor1(void){
   //Inicia a comunicacion I2C en la direccion selecionada
  Wire.beginTransmission(u8_BH1750_address);
  //Configuracion del sensor para el modo de operacion
  Wire.write(mode_HighResolution_1);
  //Finaliza la comunicacion
  Wire.endTransmission();
   //peticion delectura del sensor, esperando 2 bytes de respuesta
  Wire.requestFrom(u8_BH1750_address, 2); //
   //Espera la llegada de  dos 2 bytes
  if (2 <= Wire.available()) {
    sensor1_result = Wire.read();
    sensor1_result = sensor1_result << 8;
    sensor1_result |= Wire.read();
  }else{
      sensor1_result=-1;
  }
}
```

- **Nodo Remoto 3: Área de SALA OPERATIVA.**

```
#include <BH1750.h>//libreria para sensor de luz
#include <Wire.h>//libreria para comunicacion de sensores de luz
#include "DHT.h" //librería para sensores humedad y temperatura
#include <math.h>
#include <SoftwareSerial.h>
char cabecera;
char direccion[4];
char funcion;
char trailer;
long direccion_c;
SoftwareSerial mySerial(10, 11); // RX, TX
//Constantes para comunicacion RS485
const int EnTxPin =  2;  // HIGH:TX y LOW:RX
const int mydireccion = 103; //Direccion del esclavo
//se declaran variables para los sensores y los pines correspondientes
 de humedad y temperatura
DHT dht1(6, DHT22);
//Direccion de  sensor BH1750
byte u8_BH1750_address = 0x23; //Pin ADDR en nível lógico 0
//Instrucciones de modo lectura
byte mode_HighResolution_1 = 0x10; //Modo de alta precision con 1 lux
 de resolucion (120ms de tiempo de lectura)
//variables para sensores de luminancia
uint16_t sensor1_result = 0;
//Definimos estructuras union para dar a los datos de los sensores
//un formato apropiado para la transmisión
//Para la humedad
union h_tag {
  float float_h;
  byte byte_h[4];
} h;
//Para la temperatura
union t_tag {
  float float_t;
  byte byte_t[4];
```

```
} t;
//Para la luz
union l_tag {
  float float_l;
  byte byte_l[4];
} l;
void setup()
{
  mySerial.begin(9600);
  while(!mySerial);
  Serial.begin(9600);
  while(!Serial);
  //Configuramos el pin de control de transmisión/recepción
  pinMode(EnTxPin, OUTPUT);
  digitalWrite(EnTxPin, LOW); //RS485 como receptor
  Wire.begin();//Inicilizar el bus I2C, la biblioteca BH1750 no lo hace
 automaticamente
  dht1.begin();//se inicializa los sensores de humedad y temperatura
  pinMode(8, OUTPUT);//encender el led
}
void loop(){
  digitalWrite (8,HIGH);
  //Si existen datos en la entrada del puerto serial
  if(mySerial.available() > 0){
    while(mySerial.available() > 0){
      //Buscamos la cabecera de la trama
    cabecera = mySerial.read();
    if (cabecera == 'I'){
      //Si la encontramos, leemos el resto de la trama
      delay(3);
      direccion[0] = mySerial.read();
      delay(3);
      direccion[1] = mySerial.read();
      delay(3);
      direccion[2] = mySerial.read();
      delay(3);
      funcion = mySerial.read();
```

```
      delay(3);
      trailer = mySerial.read();
//Comprobamos si los datos recibidos tienen estructura de una trama
      if (trailer == 'F') {
        direccion_c = atol(direccion);
//Si la dirección de la trama es igual a la del dispostivo se envían
datos
        if (direccion_c == mydireccion){
          digitalWrite(EnTxPin, HIGH); //RS485 como transmisor
          //Leemos los sensores de humedad, temperatura y luz
          float h1 = dht1.readHumidity(); //variable humedad
          float t1 = dht1.readTemperature();//variable temperatura
          read_sensor1();
          int lectura = sensor1_result; //realizamos  la lectura del
 sensor     luz
          if (isnan(h1) || isnan(t1)) {
            h1=-1.0;//error obteniendo datos de humedad
            t1=-1.0;//error obteniendo datos de temperatura
          }
          //Damos a los datos un formato apropiado para la transmisión
          h.float_h = h1;
          t.float_t = t1;
          l.float_l = float(lectura);
          mySerial.print('i'); //inicio de trama
          mySerial.print(mydireccion,DEC); //direccion
          mySerial.write(h.byte_h, 4); //valor del sensor
          mySerial.write(t.byte_t, 4);
          mySerial.write(l.byte_l, 4);
          mySerial.print('f'); //fin de trama
          mySerial.flush(); //Esperamos hasta que se envíen los datos
          digitalWrite(EnTxPin, LOW); //RS485 como receptor
          Serial.print(cabecera);
          Serial.print(direccion[0]);
          Serial.print(direccion[1]);
          Serial.print(direccion[2]);
          Serial.print(funcion);
          Serial.println(trailer);
```

```
        }
     }
      else {
        Serial.println("Trama recibida con errores");
      }
    }
   }
  }
  digitalWrite(8, LOW);//apagar Led
}
//Funciones para sensores de luminancia
void read_sensor1(void){
   //Inicia a comunicacion I2C en la direccion selecionada
  Wire.beginTransmission(u8_BH1750_address);
  //Configuracion del sensor para el modo de operacion
  Wire.write(mode_HighResolution_1);
  //Finaliza la comunicacion
  Wire.endTransmission();
   //peticion delectura del sensor, esperando 2 bytes de respuesta
  Wire.requestFrom(u8_BH1750_address, 2); //
   //Espera la llegada de  dos 2 bytes
  if (2 <= Wire.available()) {
    sensor1_result = Wire.read();
    sensor1_result = sensor1_result << 8;
    sensor1_result |= Wire.read();
  }else{
      sensor1_result=-1;
  }
}
```

11.3 Anexo 2: Desarrollo de programación en Python

11.3.1 Script de programación en Python.

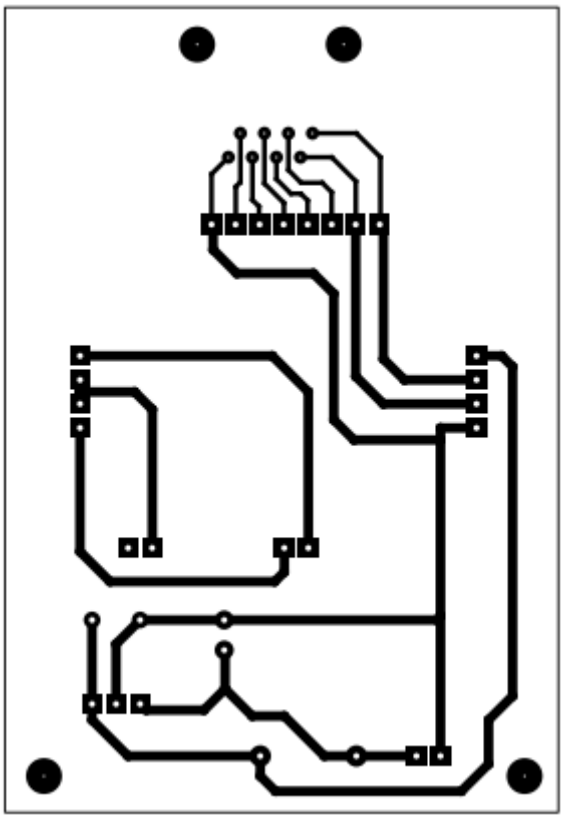
```
import psycopg2
import serial
import time
import smtplib
# Comunicacion con puerto virtual que crea arduino
puerto = "/dev/ttyACM0"
# funcion ser para obtener datos arduino
ser = serial.Serial(puerto, 9600, timeout=10)
# agregar columnas_sx necesarias segun el numero de sensores
columnas_sensor = "id, n_sensor ,dshum, dstem, dslux, fecha"
postgres_insert = ""  # variable vacia
# Insertar datos en postgresql
conn  =  psycopg2.connect(host="localhost",  database="datosarduino",
 user="user", password="password")
cur = conn.cursor()
# crear variabe sensor vacia para alamacenar el nombre del mismo
sensor = ""
 # crear variable auxiliar para tomar los datos por separado del puerto
 serial
  rx_aux = ""
  rx_auxx = ""
  ini_time = int(round(time.time() * 1000))
  fin_time = int(round(time.time() * 1000))
  postgres_insert1 = ""
  postgres_insert2 = ""
  postgres_insert3 = ""
# consultar los niveles maximos registrados en la tabla de niveles para
 comparar con las lecturas
niveles_sql = "SELECT nivmax,dispos FROM niveles where estado='A'"
cur.execute(niveles_sql)
 rows = cur.fetchall()
for row in rows:
    if row[1].strip() == 's1':
```

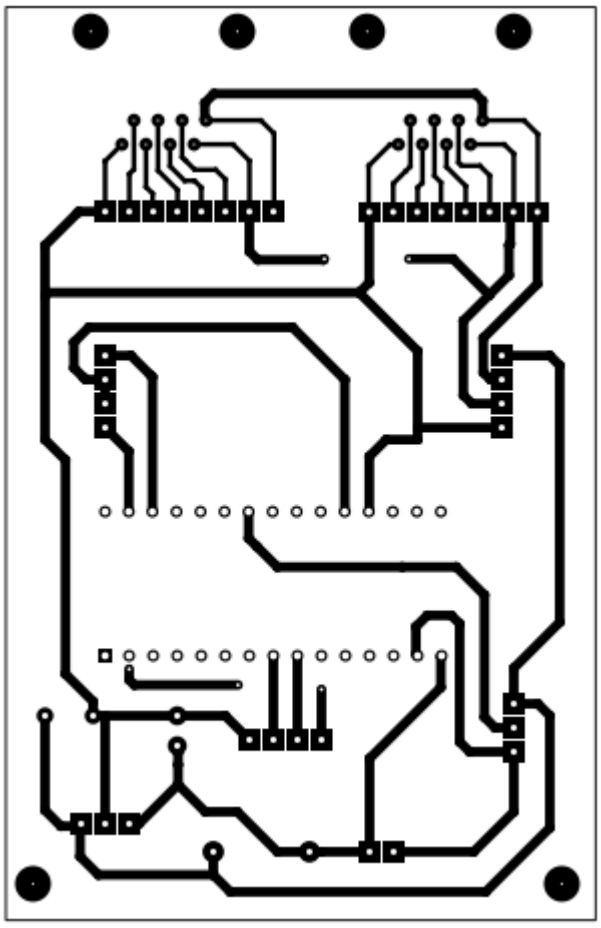
```
        rx_auxx1 = row[0]
    if row[1].strip() == 's2':
        rx_auxx2 = row[0]
    if row[1].strip() == 's3':
        rx_auxx3 = row[0]
while True:
    try:
        if ser.inWaiting() > 0:
            rx = ser.readline()
             # Tratamiento de la cadena que contiene la trama de datos.
           rx_auxx = str(rx)
           rx_auxx = rx_auxx.replace('\\r\\n', ' ')
           rx_auxx = rx_auxx.replace('\\', '')
           rx_auxx = rx_auxx.rsplit(",")
            if len(rx_auxx) == 4:
                rx_aux = str(rx_auxx[1]) + ',' + str(rx_auxx[2]) + ","
 + str(rx_auxx[3]).split(" ")[0]
                if ("S1" in str(rx)):
                    postgres_insert1 = "INSERT INTO sensor (" +
 columnas_sensor + ") VALUES ( nextval('serial'),1, " + rx_aux +
 ",NOW());"
                    rx_auxns = 's1'
                elif ("S2" in str(rx)):
                    postgres_insert2 = "INSERT INTO sensor (" +
columnas_sensor + ") VALUES ( nextval('serial'),2, " + rx_aux +
",NOW());"
                    rx_auxns = 's2'
                elif ("S3" in str(rx)):
                    postgres_insert3 = "INSERT INTO sensor (" +
columnas_sensor + ") VALUES ( nextval('serial'),3, " + rx_aux +
",NOW());"
                    rx_auxns = 's3'

                 if int(round(time.time() * 1000)) >= ini_time + (60 *
1000) and \
                      (( "S1" in str(rx) and float(rx_auxx[2]) >
rx_auxx1) \
```

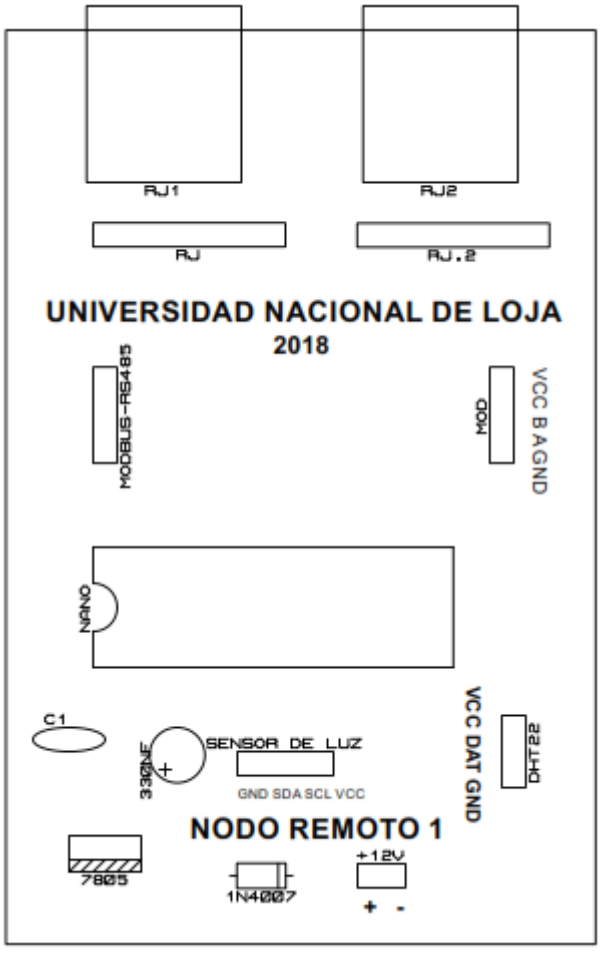
```
                                    or ("S2" in str(rx) and
float(rx_auxx[2]) > rx_auxx2) \
                                     or ("S3" in str(rx) and
float(rx_auxx[2]) > rx_auxx3) \
                                ):
                    cur.execute(postgres_insert1)
                    cur.execute(postgres_insert2)
                    cur.execute(postgres_insert3)
                    ini_time = int(round(time.time() * 1000))
                    origen = " "
                    if rx_auxns == 's1':
                        origen = "ups"
                    if rx_auxns == 's2':
                        origen = "data center"
                    if rx_auxns == 's3':
                        origen = "sala de operaciones"
                    #  tomar los correos designados para enviar el mail
                   nivel_sql = "SELECT emailp, emails, emailt FROM
emails where estado='A'"
                    cur.execute(nivel_sql)
                    rows = cur.fetchall()
                    correo1 = ''
                    correo2 = ''
                    correo3 = ''
                    for row in rows:
                        correo1 = row[0]
                        correo2 = row[1]
                        correo3 = row[2]
                    # especificar el servidor start TLS
                    s = smtplib.SMTP('smtp.gmail.com', 587)
                    s.starttls()
                    # autenticar la cuenta
                    s.login("ecu911.loja.tecnologia@gmail.com",
"contraseña")
                    # mensaje a enviar
                   message = "Alerta de temperatura más allá del nivel
permitido para el sensor " + origen
```

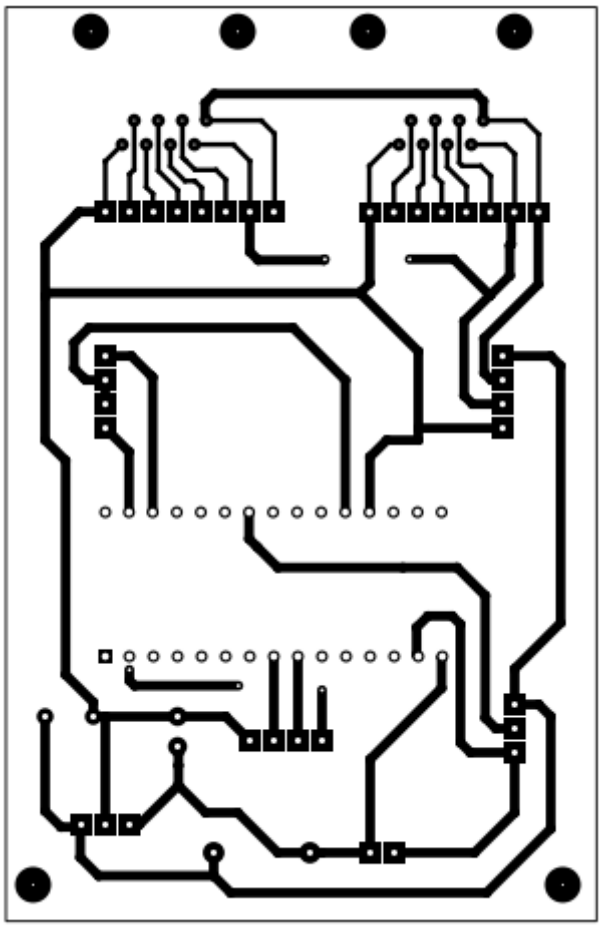
```
                    # enviar el mail  (de quien envía, quien recibe,
mensaje a enviar)
                     s.sendmail("ecu911.loja.tecnologia@gmail.com",
correo1, message)
                     s.sendmail("ecu911.loja.tecnologia@gmail.com",
correo2, message)
                     s.sendmail("ecu911.loja.tecnologia@gmail.com",
correo3, message)
                     # culminar la sesión
                     s.quit()
                if int(round(time.time() * 1000)) >= fin_time + (120 *
1000):
                     cur.execute(postgres_insert1)
                     cur.execute(postgres_insert2)
                     cur.execute(postgres_insert3)
                     fin_time = int(round(time.time() * 1000))
                 conn.commit()
                except serial.SerialException:
       print('\n...Error Fatal: Comunicacion serial perdida!!!')
        bar.close()
    except KeyboardInterrupt:  # cortar con teclado para interrumpir
Ctrl+C
        print('\n...Programa Detenido Manualmente (Ctrl + c)!!!')
        bar.close()
        break
cur.close()
conn.close()
```

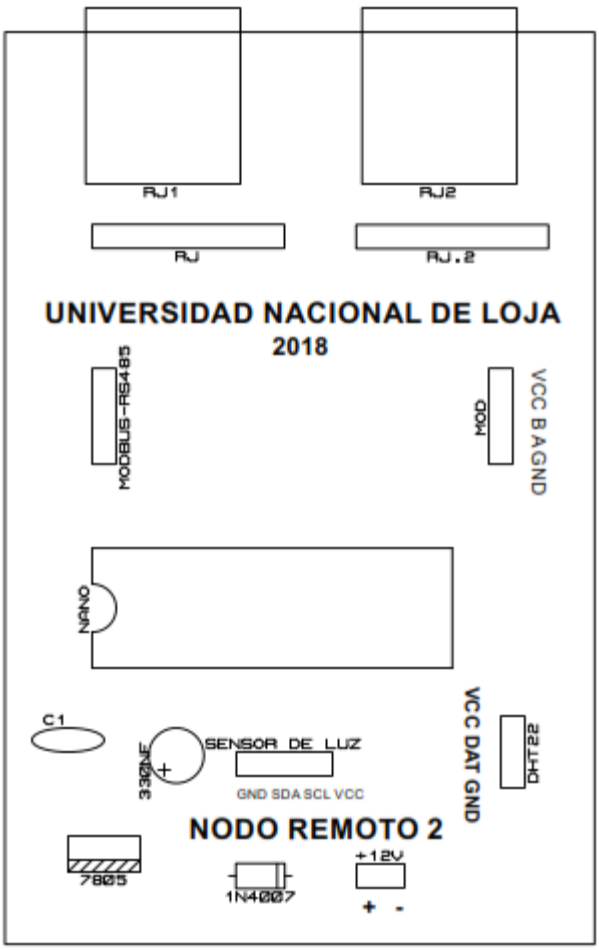
11.4 Anexo 3: Circuito Impreso para diseño de placas PCB

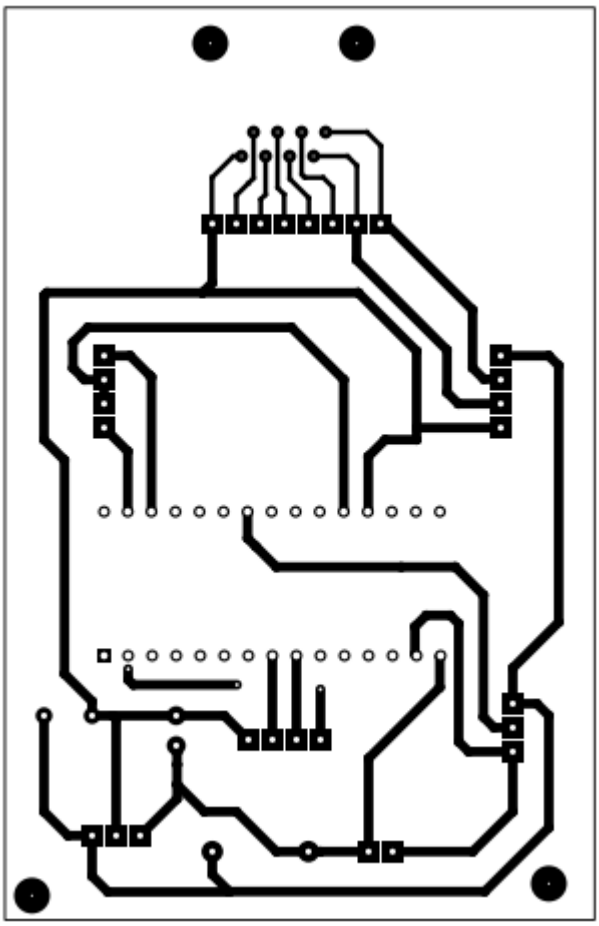


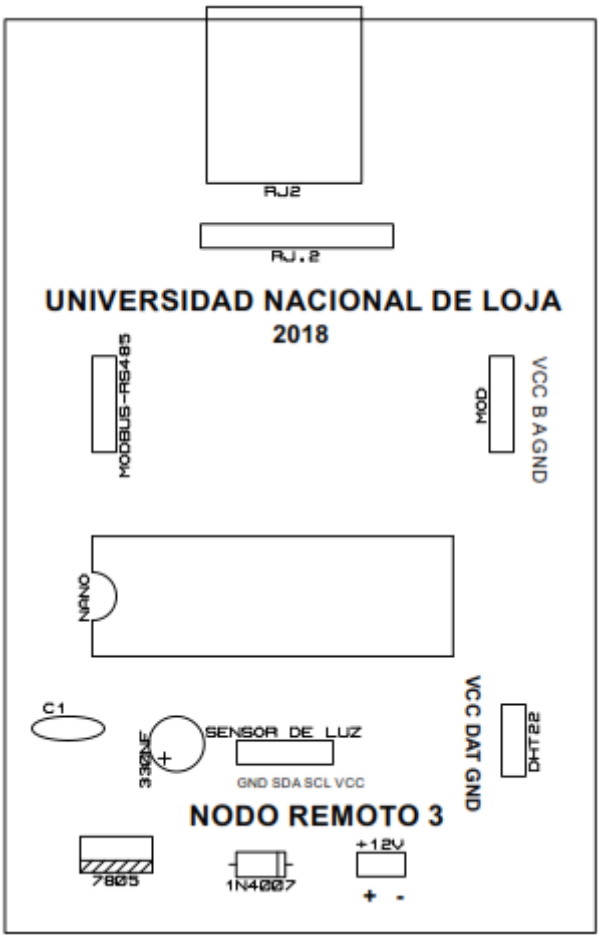







RJ1
RJ2
RJ
RJ.2
UNIVERSIDAD NACIONAL DE LOJA
2018
MODBUS-RS485
VCC B A GND
NANO
C1
330NF
SENSOR DE LUZ
GND SDA SCL VCC
VCC DAT GND
DHT22
NODO REMOTO 2
7805
1N4007
+12V
+ -




RJ2
RJ.2
UNIVERSIDAD NACIONAL DE LOJA
2018
MODBUS-RS485
VCC B A GND
NANO
C1
330NF
SENSOR DE LUZ
GND SDA SCL VCC
VCC DAT GND
DHT22
NODO REMOTO 3
7805
1N4007
+12V
+ -

11.5 Anexo 4: Placas y Módulos ensamblados

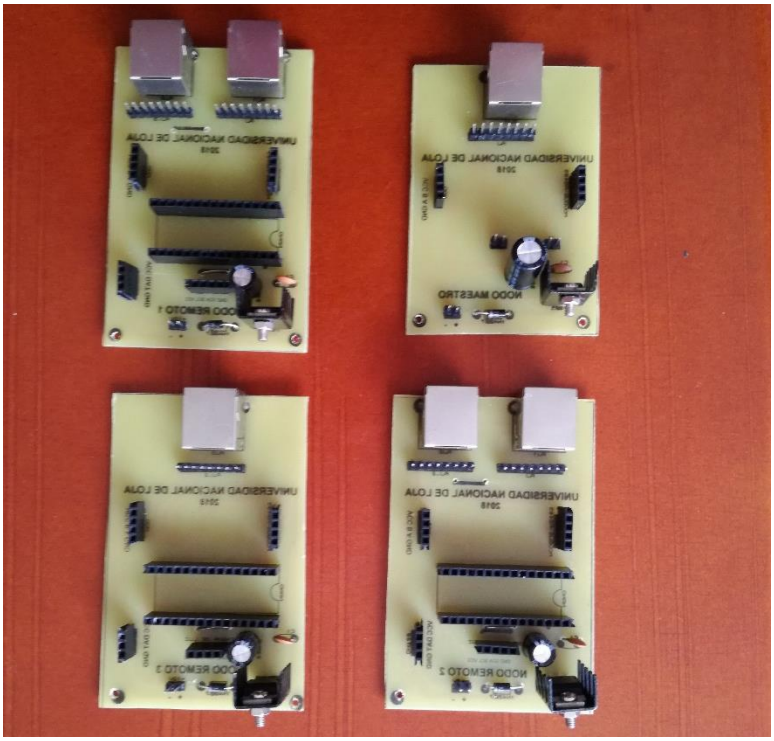


Figura 50. Placas PCB terminadas. Fuente [El Autor]

Figura 51. Presentación de Módulos. Fuente [El Autor]

11.6 Anexo 5: Especificaciones Técnicas.

11.6.1 Datos Técnicos de sensor DHT22.

Model	AM2302	
Power supply	3.3-5.5V DC	
Output signal	digital signal via 1-wire bus	
Sensing element	Polymer humidity capacitor	
Operating range	humidity 0-100%RH;	temperature -40~80Celsius
Accuracy	**humidity +-2%RH**(Max +-5%RH);	temperature +-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH;	temperature 0.1Celsius
Repeatability	humidity +-1%RH;	temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH	
Long-term Stability	+-0.5%RH/year	
Interchangeability	fully interchangeable	

11.6.2 Datos Técnicos de sensor BH1750.

●Absolute Maximum Ratings

Parameter	Symbol	Ratings	Units
Supply Voltage	Vmax	4.5	V
Operating Temperature	Topr	-40～85	℃
Storage Temperature	Tstg	-40～100	℃
SDA Sink Current	Imax	7	mA
Power Dissipation	Pd	260※	mW

※ 70mm × 70mm × 1.6mm glass epoxy board. Derating in done at 3.47mW/℃ for operating above Ta=25℃.

●Operating Conditions

Parameter	Symbol	Ratings			Units
		Min.	Typ.	Max.	
Vcc Voltage	Vcc	2.4	3.0	3.6	V
I^2C Reference Voltage	VDVI	1.65	-	Vcc	V

11.6.3 Datos Técnicos de módulo MAX485.

PIN DESCRIPTION

No.	Name	Function
1	RO	Receive output: if A > B by 200mV, RO will be high; if A < B by 200mV, RO will be low.
2	$\overline{RE}$	Receiver Output Enable. RO is enabled when $\overline{RE}$ is low; RO is high impedance when $\overline{RE}$ is high.
3	DE	Driver Output Enable. The driver outputs are enabled when DE is high. They are high impedance when DE is low. If the driver outputs are enabled, the parts function as line drivers. While they are high impedance, they function as line receivers if $\overline{RE}$ is low.
4	DI	Driver input. A low on DI forces output A low and output B high. Similarly, a high on DI forces output A high and output B low.
5	GND	Ground
6	A	Driver Output and Receiver differential input.
7	B	Driver Output and Receiver differential input.
8	V_{CC}	Positive Supply: 4.75V ≤ Vcc ≤ 5.25V

ABSOLUTE MAXIMUM RATINGS

Parameter	Symbol	Value	Unit
Supply Voltage	V_{CC}	12V	V
Control Input Voltage	V_{CIV}	-0.5 to (V_{CC}+0.5)	V
Driver Input Voltage	DI	-0.5 to (V_{CC}+0.5)	V
Driver Output Voltage (A, B)	DO	-8 to +12.5	V
Receiver Input Voltage (A, B)	V_{RIV}	-8 to +12.5	V
Receiver Output Voltage	RO	-0.5 to (V_{CC}+0.5)	V
8-Pin Plastic DIP Continuous Power Dissipation (derating 9.09mW/°C above +70°C)	P_{DIP}	727	mW
8-Pin SOP Continuous Power Dissipation (derating 5.88mW/°C above +70°C)	P_{SOP}	471	mW
Operating Temperature Range	T_A	0 to +70	°C
Storage Temperature Range	T_{STG}	-65 to +160	°C
Lead Temperature, 10 sec	T_L	+300	°C

11.6.4 Requisitos mínimos de hardware y sotfware del servidor local para instalar la Pila Elastic.

Tabla 15. Requisitos mínimos para instalación y funcionamiento de la pila Elastic.

Requisitos Mínimos	**Hardware**	**Software**
Procesador	Intel x86 o procesador compatible	Sistema operativo Linux x 64
Memoria RAM	2 GB	Soporte Protocolo TCP/IP
Disco Duro	8 GB	
		Repositorio: https://bitnami.com/stack/elk/README.txt

11.7 Anexo 6: Glosario de Terminos

ACK: bit de comprobación o cheksum.
ADC: convertidor analógico digital.
BSD: Distribución de Software de Berkeley.
DC: corriente contínua.
dahsboard: interfaz gráfica de usuario para presentación de datos.
dshum: datos sensor de humedad
dstem: datos sensor temperatura
dslux: datos sensor de iluminanción.
Elastic: conjunto de protocolos independientes.
ELK: Elasticsearch, Logstash y Kibana.
EIA/TIA: Asociación de Industrias Electrónicas/ Asociación de Industrias de las Telecomunicaciones.
GND: tierra.
Gb: Giga Byte.
GUI: Interfaz gráfica de usuario.
Gauge: Tipo de dato para presentación web.
half-dúplex: comunicación en una sola vía.
HIGH: nivel de estado lógico alto
IDE: Entorno de Desarrollo Integrado.
I/O: entrada/salida
json: Notación de Objeto de Java Script.
jdvc: java database connectivity.
KB: Kilobytes
Kbps: Kilobytes por segundo.
LOW: nivel de estado lógico bajo
Localhost: dispositivo local.
Logs: registros secuenciales de un archivo.
Mbps: Megabytes por segundo
MSB: byte más significativo
MHZ: Mega Hertz
mA: miliamperios
mV: milivoltios
n_sensor: número de sensor.
Pila Elastic: conjunto de herramientas informáticas
PWM: ancho de pulso

PK: claves primarias

PCB: Placa de circuito impreso.

Query: pedido de datos en lenguaje SQL.

RTU: unidad terminal remota

R/W: leer/escribir

RX: recepción de datos.

SDA: Datos seriales

SCL: Reloj Serial

SGDB: sistema de gestor de base de datos

SMTP: protocolo simple de transferencia de correo.

SSL: capa de sockets seguros.

Serial1: puerto serial virtual.

Source: fuente de datos.

SQL: lenguaje de consulta estructurada.

TX: transmisión de datos.

TTL: Lógica Transistor-Transistor.

TB: Tera byte.

V: voltaje

VCC: voltaje en corriente contínua.

11.8 Anexo 7: Solicitud de Aceptación y Certificado de Funcionamiento del Sistema Propuesto.

Loja, 09 de octubre de 2018

Sr.

Carlos Eduardo Flores Gaona.

Estudiante de la carrera de Ingeniería en Electrónica y Telecomunicaciones de la Universidad Nacional de Loja.

Ciudad. -

De mi consideración:

Por medio de la presente me permito comunicar a usted, que para fines de desarrollo del proyecto de tesis denominado **"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO EN TIEMPO REAL DE VARIABLES AMBIENTALES DE ÁREAS CRÍTICAS DEL EDIFICIO SIS ECU-911 LOJA."**, ha sido asignado a la Unidad de Gestión Local de Soporte Tecnológico ECU-911 Loja con la finalidad de que a partir del **12 de marzo de 2018** realice las actividades previstas para el cumplimiento de dicho proyecto. El responsable de supervisar las actividades del proyecto estará a cargo del Ing. Juan Pablo Cabrera el cuál vigilará el cumplimiento de los objetivos planteados.

Particular que informo a usted para fines pertinentes.

Atentamente.

Abogada María Esther Mora Valencia.

Jefe Operativo Local ECU-911 Loja

Loja 09 de octubre de 2018

Ing.

Juan Pablo Cabrera

Especialista de Soporte Tecnológico Local
SIS ECU-911 LOJA

CERTIFICA:

Que el Sr. **Carlos Eduardo Flores Gaona**, con cédula **1103753065** estudiante de la carrera de Ingeniería en Electrónica y Telecomunicaciones de la Universidad Nacional de Loja, realizó el desarrollo del proyecto de tesis denominado "DISEÑO E IMPLMENTACIÓN DE UN SISTEMA DE MONITOREO EN TIEMPO REAL DE VARIABLES AMBIENTALES DE ÁREAS CRÍTICAS DEL EDIFICIO SIS ECU-911 LOJA."

Las actividades fueron ejecutadas de manera satisfactoria cumpliendo con los objetivos específicos propuestos para el proyecto de tesis y dentro de los lineamientos establecidos, bajo supervisión del personal técnico encargado, desde el 12 de marzo 2018 al 08 octubre 2018, como se detalla en el anexo.

ANEXO

Semanas	**Actividades**
12 / 23 de marzo de 2018	Presentación de sensores y tarjetas electrónicas seleccionadas. Programación de sensores y tarjetas electrónicas seleccionadas. Pruebas de funcionamiento.
26 de marzo / 06 de abril de 2018	Presentación del diseño de red alámbrica de sensores. Revisión de módulos MAX485 para la red alámbrica. Pruebas de funcionamiento.
09 / 20 abril 2018	Programación en Software Arduino de dispositivos electrónicos seleccionados para el sistema de monitoreo.
23 de abril / 04 de mayo de 2018	Programación en Software Arduino de dispositivos electrónicos seleccionados para el sistema de monitoreo.
07 / 18 de mayo de 2018	Instalación de software requerido para la simulación de entorno virtual Pycharm. Diseño de la programación en software Python.
21 de mayo / 01 de junio de 2018	Desarrollo de la programación en software Python 3 para conexión con software Arduino.
04 / 15 de junio de 2018	Desarrollo de la programación en software Python 3 para conexión con el gestor de base de datos PostgreSQL. Pruebas de funcionamiento.
18 / 29 de junio de 2018	Crear base de datos en PostgreSQL.

	Conexión con software Python 3-
02 / 13 de julio de 2018	Conexión con software Python 3- Pruebas de funcionamiento.
16 / 27 de julio de 2018	Pruebas de funcionamiento.
30 de julio / 09 de agosto de 2018	Instalación de software para gestión y procesamiento de datos. Desarrollo de interfaz web. Pruebas de funcionamiento.
13 / 24 de agosto de 2018	Diseño y construcción de PCB. Diseño y construcción de módulos Nodo Maestro y Nodos Remotos.
27 de agosto / 07 de septiembre de 2018	Implementación de hardware del sistema de monitoreo. Implementación de la red alámbrica.
10 /021 de septiembre de 2018	Pruebas de funcionamiento del sistema de monitoreo implementado.
24 7 28 de septiembre de 2018	Pruebas de funcionamiento del sistema de monitoreo implementado. Validación de resultados
01 / 08 de octubre de 2018	Entrega del sistema de monitoreo al personal encargado del edificio SIS ECU 911-LOJA.