UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA: INGENIERÍA DE SISTEMAS

Trabajo de titulación previo a la obtención del título de: INGENIEROS DE SISTEMAS

TEMA:

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO DE LAS VARIABLES AMBIENTALES Y ESTADO DE CARGA DEL UPS CON PLATAFORMA DE HARDWARE ARDUINO A TRAVÉS DE SENSORES Y UNA APLICACIÓN ANDROID PARA EL DATA CENTER DE LA EPMAPS.

AUTORES:
DAVID MARCELO OÑA SALAZAR
BYRON ALEXIS VACA MÁRMOL

DIRECTOR:
JORGE ENRIQUE LÓPEZ LOGACHO

Quito, abril de 2015

DECLARATORIA DE RESPONSABILIDAD Y AUTORIZACIÓN DE USO DEL TRABAJO DE TITULACIÓN

Nosotros, autorizamos a la Universidad Politécnica Salesiana la publicación total o parcial de este trabajo de titulación y su reproducción sin fines de lucro.

Además, declaramos que los conceptos, análisis desarrollados y las conclusiones del presente trabajo son de exclusiva responsabilidad de los autores.

Quito, abril 2015

_____ _____

David Marcelo Oña Salazar
CC. 1719206375

Byron Alexis Vaca Mármol
CC. 1722685961

DEDICATORIA

Dedico este trabajo a Dios por protegerme durante todo mi camino y darme fuerzas para lograr uno de mis objetivos.

A mis padres por la confianza brindada y por su apoyo incondicional tanto económico como anímicamente para culminar mi formación académica, a mi compañero de tesis y amigo Alexis Vaca con quien a lo largo de este tiempo hemos superado los obstáculos para llegar a este término, a nuestro tutor de tesis y un gran amigo Ing. Jorge López por todo su apoyo y el valioso asesoramiento para la realización de la misma.

David Marcelo Oña Salazar.

Este trabajo lo dedico en primer lugar a mis padres, Augusto Vaca y Mary Mármol, por haberme dado unos estudios de calidad y haber sabido inculcarme los valores que ahora profeso, a mi tutor Ing. Jorge López, quien se ha tomado el arduo trabajo de transmitirme sus diversos conocimientos, especialmente del campo y de los temas que corresponden a mi profesión y finalmente al Ing. Cristóbal Morocho (Jefe de Infraestructura y Seguridad de la EPMAPS) por la colaboración, paciencia, apoyo brindados desde siempre y sobre todo por esa gran amistad que me brindó y me brinda, por escucharme y aconsejarme siempre.

Byron Alexis Vaca Mármol

AGRADECIMIENTOS

A la Universidad Politécnica Salesiana que nos abrió las puertas al conocimiento para formarnos como grandes profesionales.

A los docentes que en la trayectoria de la carrera universitaria compartieron sus conocimientos y valiosas experiencias que nos ayudaron a formarnos profesionalmente, y de manera muy especial a nuestro tutor Ing. Jorge López quien ha sabido guiarnos y aconsejarnos para llegar con éxito al último escalón de nuestra etapa universitaria gracias a su apoyo y a sus palabras de aliento que nos enseñó a ver los problemas y darles solución sin importar lo complejo que sean.

Finalmente un fraterno agradecimiento a la EPMAPS (Empresa Pública Metropolitana de Agua Potable y Saneamiento) que confió en nosotras para la elaboración de nuestro proyecto, en especial al Ing. Cristóbal Morocho por su colaboración en facilitar la información requerida.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Índice de Anexos

RESUMEN

Este trabajo muestra el diseño e implementación de un sistema de monitoreo de las variables ambientales y estado de carga del UPS con plataforma de hardware Arduino a través de sensores y una aplicación Android para el Data Center de la EPMAPS.

Nuestro sistema llamado MTI, tiene como objetivo monitorear la temperatura, humedad, fuga de refrigerante, estado de carga del UPS dentro de un Data Center y poder visualizarlo en un navegador web o una aplicación Android. La principal característica del MTI es que es un sistema autónomo, que permite enviar SMS y correos electrónicos si se produjera una variación en los parámetros de monitoreo de los sensores.

El código de la aplicación MTI está totalmente diseñado en Java, mediante el entorno gráfico NetBeans, Eclipse, PostgreSQL que son plataformas libres.

El hardware de monitoreo está compuesto por un sensor análogo de temperatura (LM35), un sensor digital (DHT11) y una fotocelda que capta la luz que es emitida por el dispositivo de detección de fugas de refrigerante de marca Honeywell. Los sensores están conectados al módulo ArduinoUno el cual envía a la Aplicación MTI los parámetros monitoreados para ser comparados con los umbrales definidos para cada sensor y en caso de producirse una variación retorna ArduinoUno la instrucción de enviar por SMS a través del módulo ArduinoGSMShield, el nombre del sensor y el valor de la variable a los números de celular registrados en la base de datos PostgreSQL.

El MTI tiene dos modos de funcionamiento uno manual que es controlado por el servidor web y otro autónomo que entra a funcionar en caso de que el suministro de energía del UPS se agote.

Complementado con varias otras pruebas de hardware finales, el MTI es estable y ha demostrado el potencial de resistencia ante los cambios bruscos de temperatura y humedad.

MTI representa un logro significativo en términos de integración de hardware y software manteniendo la capacidad de realizar mejoras futuras.

ABSTRACT

This paper presents the design and implementation of a system for monitoring environmental variables and charge state of the UPS with Arduino hardware platform through sensors and an Android application for the Data Center of EPMAPS.

Our system called MTI, aims to monitoring temperature, humidity, refrigerant leakage, charging status UPS within a Data Center and can be viewed from a web browser or an Android application. The main feature of MTI is that it is an autonomous system that lets you send SMS and emails if a change occurs in the monitoring parameters of the sensors.

The code MTI application is completely designed in Java, using the graphical environment NetBeans, Eclipse, PostgreSQL that are free platforms.

Monitoring hardware comprises an analog temperature sensor (LM35), a digital sensor (DHT11) and a photocell which captures light which is emitted by the device of refrigerant leak detection Honeywell brand. The sensors are connected to ArduinoUno module which sends Implementation MTI monitored parameters to be compared with the thresholds defined for each sensor and in the event of a variation returns ArduinoUno instruction sent by SMS through ArduinoGSMShield module, the sensor name and value of the variable cell numbers recorded in the data base PostgreSQL.

The MTI has two modes of operation a manual which is controlled by the web server and entering another autonomous function if the power supply is depleted UPS.

Supplemented with several other tests end hardware, the MTI is stable and has demonstrated the potential of resistance to sudden changes in temperature and humidity.

MTI represents a significant achievement in terms of integration of hardware and software while maintaining the ability to make future improvements**.**

INTRODUCCIÓN

El Data Center es un espacio acondicionado donde se concentran los recursos necesarios para el procesamiento de la información de una organización. Las plataformas Arduino son microcontroladores que tienen gran flexibilidad de programación y bajo costo que permiten el desarrollo de múltiples diseños de control. Al ser una plataforma de hardware tanto su diseño como distribución es libre, pueden utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin haber adquirido ninguna licencia.

La plataforma Arduino se está extendiendo y complementado enormemente en los últimos años en diferentes ámbitos tecnológicos tales como:

- Electrónica digital
- Programación en C/C++/Java
- Microcontroladores

Por todas estas características positivas, se ha considerado proponer a la EPMAPS en aportar el diseño e implementación de un sistema de monitoreo basado en la plataforma Arduino.

Se propondrá una solución que permita monitorear y actuar cuando se detecte amenazas ambientales en el Data Center de la EPMAPS (Empresa Pública Metropolitana de Agua Potable y Saneamiento) como:

- Si el aire acondicionado deja de funcionar y la temperatura comienza a aumentar.
- Si se tiene un problema de flujo de aire en el rack o una pérdida de refrigerante que pueda estar inadvertida debajo del piso flotante.
- Si existe variación o pérdida de energía en el UPS.

Al detectar estas amenazas ambientales a tiempo se puede actuar de forma preventiva y correctiva protegiendo las inversiones en tecnología y aumentando la continuidad de sus servicios.

El sistema de monitoreo propuesto ofrecerá una solución de vigilancia de los parámetros ambientales para el óptimo funcionamiento del Data Center optimizando recursos de infraestructura al minimizar la intervención de personal técnico.

CAPÍTULO 1

FASE INICIAL

1.1. Definición del proyecto

Se propone una solución que permita monitorear y actuar cuando se detecte amenazas ambientales en el Data Center de la EPMAPS como:

- Si el aire acondicionado deja de funcionar y la temperatura comienza a aumentar.
- Si se tiene un problema de flujo de aire en el rack o una pérdida de refrigerante que pueda estar inadvertida debajo del piso flotante.
- Si existe variación o pérdida de energía en el UPS (Uninterruptible Power Supply).

Al detectar estas amenazas ambientales a tiempo se puede actuar de forma preventiva y correctiva, lo cual permite proteger las inversiones tanto tecnológicas y mantener activos los servicios. El sistema de monitoreo propuesto ofrecerá una solución de vigilancia de los parámetros ambientales para el óptimo funcionamiento del Data Center; optimizando recursos de infraestructura al minimizar la intervención de personal técnico.

El sistema brindará alarmas en tiempo real vía SMS (Short Message Service), correo electrónico e interacción con la aplicación Android. Esto con el fin de tener el servicio de monitoreo en tiempo real y obtener la capacidad de tomar acciones proactivas y reactivas ante cualquier incidencia que pueda darse dentro del entorno del control de las variables ambientales en del Data Center.

Una vez que el sistema de monitoreo se encuentre operativo, los umbrales de alerta definidos en la plataforma Arduino y el servidor de monitoreo estarán en operación 24/7 (24 horas por 7 días a la semana). Adicionalmente se propone que las alarmas podrán ser enviadas automáticamente por SMS, correo electrónico, visualización en la aplicación Android.

El Sistema de monitoreo consiste en la vigilancia continua de las variables ambientales a través de una conexión entre la plataforma de hardware ArduinoUno y el servidor de

monitoreo donde se realiza la gestión de detección de condiciones y envío de alertas. Los parámetros de alertas se configurarán en el servidor de monitoreo dependiendo la gravedad de la situación detectada. Para los sensores que permiten obtener valores (como los de temperatura y humedad), la medida se puede procesar en base a parámetros definidos en el servidor de monitoreo a partir de los cuales se entiende que la condición es de atención o de gravedad; esto permite dos tipos de comportamiento en el sistema de notificaciones en los cuales una situación de gravedad usaría alertas más llamativas (como el intervalo de SMS continuos).

Se podrá generar reportes con las gráficas de evolución de las medidas y los eventos ocurridos. La aplicación "MTI (Monitoreo de Tecnología Informática) para Android", permitirá monitorear las variables ambientales del Data Center de la EPMAPS mediante un dispositivo con sistema Android.

La aplicación permitirá a UIS (Unidad de Infraestructura y seguridad), ganar en versatilidad y tiempo, gracias a que podrán visualizar datos de los sensores en tiempo real y reconocer alarmas de cualquier anomalía o falla. Como en toda gestión de monitorización, las estadísticas son parte clave para medir el comportamiento de los sensores de tal manera que ayudará a actuar proactivamente en caso de detectarse problemas que afecten las operaciones del Data Center. MTI proveerá gráficos de monitoreo en tiempo real, contará con una apariencia muy amigable, en la pantalla inicial se visualizará el estado de las variables ambientales del Data Center que el usuario puede mantener disponibles y a la mano, como es el caso de los medidores análogos que mostrarán y emitirán información en detalle sobre el estado real y funcionamiento de los sensores.

1.2. Objetivos

1.2.1. Objetivo general.

- Diseñar e implementar un sistema de monitoreo de las variables ambientales y estado de carga del UPS con plataforma de hardware Arduino a través de sensores y una aplicación Android para el Data Center de la EPMAPS.

1.2.2. Objetivos específicos.

- Realizar un levantamiento de estado inicial de la infraestructura de control de las variables ambientales del Data Center de la EPMAPS.
- Diseñar la plataforma de software y programar los módulos Arduino con los sistemas de redundancia.
- Realizar pruebas e implementación.
- Generar un manual de usuario para la MTI

1.3. Alcance

Se propondrá una solución que permita monitorear y actuar cuando se detecte amenazas ambientales en el Data Center de la EPMAPS. El sistema de monitoreo propuesto ofrecerá una solución de vigilancia de los parámetros ambientales para el óptimo funcionamiento del Data Center optimizando recursos de infraestructura al minimizar la intervención de personal técnico.
El sistema brindará alarmas en tiempo real vía SMS, correo electrónico e interacción con la aplicación Android. Esto con el fin de tener el servicio de monitoreo en tiempo real y obtener la capacidad de tomar acciones proactivas y reactivas ante cualquier incidencia que pueda darse dentro del entorno del control de los parámetros dentro del Data Center.

La metodología a utilizar en la aplicación Android y servidor web será SCRUM, ya que es una metodología ágil que permite gestionar procesos suficientemente manejables y adaptables a los proyectos de desarrollo de software, también brinda grandes beneficios a los proyectos debido a su apertura ante posibles cambios que pueden surgir en el transcurso del desarrollo de software.

Los módulos a implementar son:

- Módulo de acceso.- Administra cuentas de los usuarios que ingresan al sistema.
- Módulo de reportes.- Permite crear una bitácora de registros de las variables ambientales.
- Módulo de configuración de sensores.- Administra umbrales de Alertas y configuración de intervalos tiempo de envío de mensajes SMS

Una vez que el sistema de monitoreo se encuentre operativo, los umbrales de alerta definidos en la plataforma Arduino y el servidor de monitoreo estarán en operación 24/7 (24 horas por 7 días a la semana).

1.4. Análisis de estado inicial.

1.4.1. Gobernanza.

1.4.2. Misión de la EPMAPS.

Proveer servicios de agua potable y saneamiento con eficiencia y responsabilidad social y ambiental.

1.4.2.1. Visión de la EPMAPS

Ser empresa líder en gestión sostenible e innovadora de servicios públicos en la región.

1.4.2.2. Estructura orgánica de la EPMAPS

En la figura 1 se observa como está estructurada la EPMAP y los diferentes departamentos que consta.

Estructura orgánica de la EPMAPS

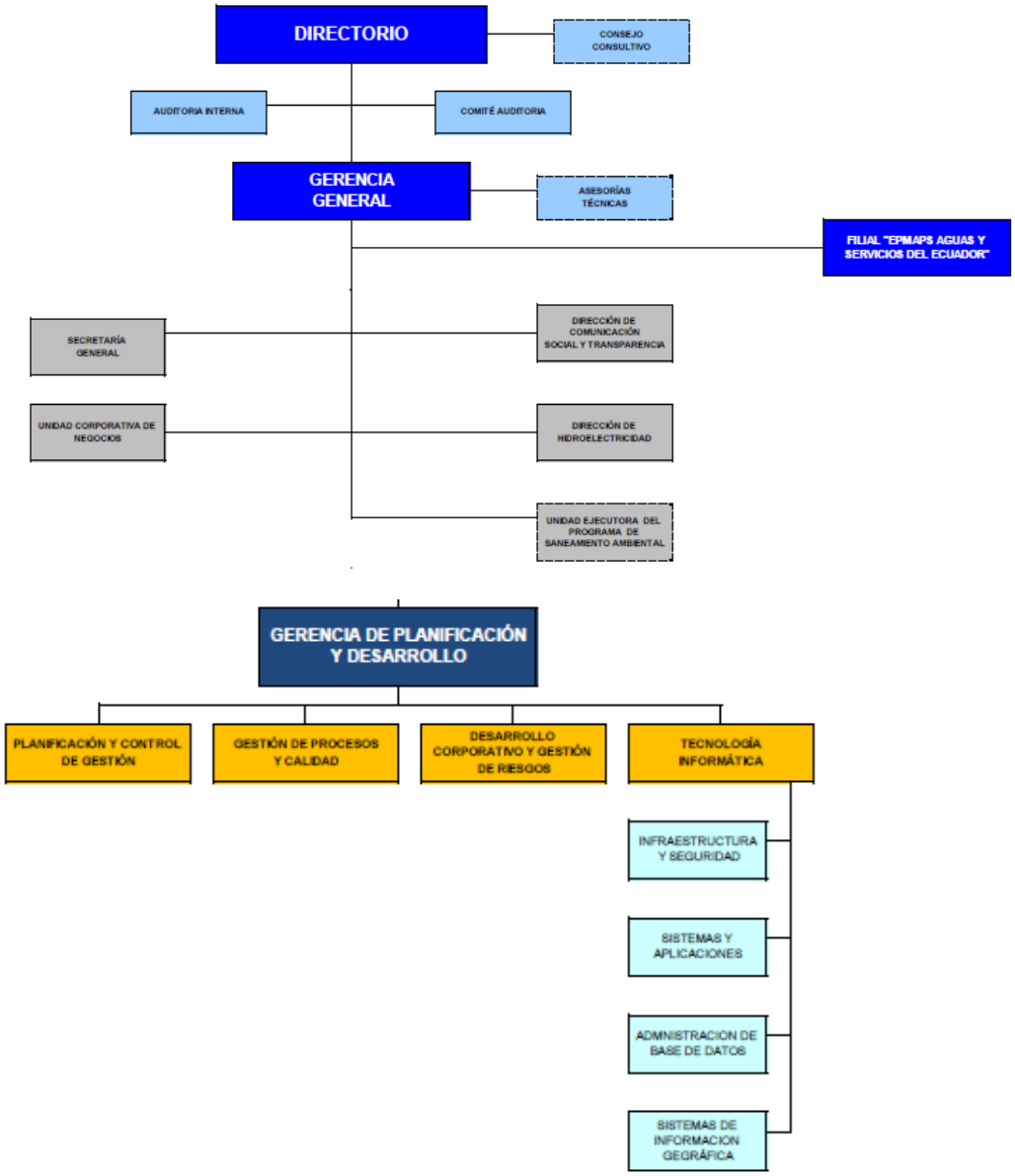


Figura 1. Mapa orgánico de la EPMAPS por departamentos Fuente: EPMAPS

1.5. TIC (Tecnologías de la Información y la Comunicación)

1.5.1. Data Center.

La infraestructura tecnológica del Data Center de la EPMAPS soporta servicios informáticos que por su naturaleza demandan disponibilidad permanente. Esta infraestructura contiene sistemas físicos de control y soporte como son: energía

regulada, UPS, aire acondicionado, control de incendios y control de accesos. El monitoreo de estos sistemas dentro del horario laboral, se supervisa por la UIS (Unidad de Infraestructura y Seguridad). Sin embargo, Es oportuno mencionar que fuera de horas laborables no se tiene un control y monitoreo directo puesto que no se cuenta con la función de operadores para el Data Center. El horario extendido es de 16:30 a 18:30 de lunes a viernes, 10:00 a 18:30 sábados dentro de un esquema 24 / 7 (24 horas por 7 días a la semana).

Por otro lado, el monitorear remotamente esta infraestructura depende del equipamiento que se pueda instalar para este efecto. Actualmente, se hace uso de un aplicativo general tipo "freeware" denominado PRTG (Paessler Router Traffic Grapher), el cual se utiliza por el personal de la UIS y que permite el monitoreo mediante el uso de la red empresarial. Como se entenderá, mientras dura el horario laboral normal este monitoreo es práctico; pero, fuera de este horario, no se tiene un medio de seguimiento y notificación directo y automático ante eventos o fallas de los sistemas físicos mencionados del Data Center.

Lamentablemente, es importante recordar que el último evento de indisponibilidad de la infraestructura informática (22 y 23 de marzo 2014), causado por un evento eléctrico no se detectó oportunamente por el departamento de TI (Tecnología informática.) en general y por UIS en particular debido a la falta de estos mecanismos de monitoreo automático y remoto.

En definitiva, se estima necesario y útil el contar con el equipamiento que concentre los sensores de los sistemas físicos del Data Center más importantes como son:

- UPS
- Aire acondicionado
- Temperatura ambiental
- Humedad ambiental
- Detección de líquidos bajo el piso falso

Y en el caso de recibir alertas sobre eventos o incidentes en estos sistemas, tomar acciones del caso según corresponda.

CAPÍTULO 2

MARCO TEÓRICO

2.1. Normas Tier.

Cuando se refiere a un Data Center no solo se está hablando de la integración de Software, Hardware y Telecomunicaciones. Los Data Center también están conformados por una serie de equipos adicionales que le permiten tener en cuenta factores importantes como son: el clima, la electricidad, sistemas de protección contra incendios, humedad, acceso de usuarios entre otros.

También se deben tener en cuenta aspectos como el recurso humano y todos los procesos asociados al Data Center, los cuales deben permanecer en correcto funcionamiento aun cuando se puedan presentar cualquier accidente o desastre natural.

El standard TIA-942 **(**Telecomunication Infrastructure Standard for Data Centers**)** se puede observar en la tabla 1 que contiene los Grados de Disponibilidad (Tier) con los que pueden clasificarse los Data Centers, estas normas Tiers están basados en información desarrollado por el Uptime Institute, un consorcio dedicado a proveer a sus miembros las mejores prácticas de planificación y gerenciamiento de Data Center, de esta manera a mayor número de Tier mayor grado de disponibilidad.

Tabla 1.

Grados de disponibilidad Tier

	Tier 1	**Tier II**	**Tier III**	**Tier IV**
Tipo de inmueble	Arrendado	Arrendado	Independiente	Independiente
Turnos del personal Persona/Turno	Ninguno Ninguno	1 Turno 1 Turno	1 + Turno 1-2/Turno	"24 siempre" 2+/Turno
Utilizable para carga crítica	100%	100%	90%	90%
KW inicial por el Gabinete (típico)	<1Kw	1-2 kW	1-2 kW	1-3 kW
KW Máxima por Gabinete (típico)	<1 kW	1-2 kW	>3 kW	>4 kW
Soporte en relación del piso	20%	30%	80-90+%	100+%
Altura del piso	12 pulgadas	18 pulgadas	30-36 pulgadas	30-42 pulgadas
Carga del piso lbs/pies (típico)	85	100	150	150+
Utilidad de voltaje	208.480	208,480	12-15 kV	12-15 kV
Puntos únicos de fallo Single Points-of-Failure	Muchos + Error Humano	Muchos + Error Humano	Algunos + Error Humano	Fuego, EPO + Algunos Errores Humano
Planes de mantenimiento de parada	2 Eventos Anuales en 12 horas	3 Eventos más de 2 años en 12 Horas	Ninguno Requerido	Ninguno Requerido
Fracasos representativos en el sitio	6 Fallas Más de 5 años	1 El fracaso Cada año	1 Fracaso cada 2,5 años	1 Fracaso Cada 5 años
Tiempo de inactividad del usuario final (basado en datos de campo)	28.8 horas	22.0 horas	1.6 horas	0.8 horas
Disponibilidad del Usuario Final Basado en ocasionada tiempo de inactividad	99,67%	99,75%	99,98%	99,99%
Meses típicos para planificar y construir	3	3-6	15-20	15-30
Fecha de Lanzamiento	1.965	1.970	1.985	1.995

Nota. KW= KiloWat, Kv= Kilovoltio.
Elaborado por: David Oña y Alexis Vaca.

2.1.1. Tier I: Data Center básico

Un Data Center de tipo Tier I puede admitir interrupciones tanto planeadas como no planeadas. Cuenta con sistemas de aire acondicionado y distribución de energía, pero puede no tener piso falso, UPS o generador eléctrico. Si los posee pueden tener varios puntos únicos de falla. La carga máxima de los sistemas en situaciones críticas es del 100%. La infraestructura del Data Center deberá estar fuera de servicio al menos una vez al año por razones de mantenimiento y/o reparaciones. Errores de operación o fallas en los componentes de su infraestructura causarán la interrupción del Data Center. La tasa de disponibilidad máxima del Data Center es 99.671% del tiempo.

2.1.2. Tier II: componentes redundantes

Un Data Center con componentes redundantes son ligeramente menos susceptibles a interrupciones, tanto planeadas como las no planeadas. Estos Data Centers cuentan con piso falso, UPS y generadores eléctricos, pero está conectado a una sola línea de distribución eléctrica. Su diseño es (N+1), lo que significa que existe al menos un duplicado de cada componente de la infraestructura. La carga máxima de los sistemas en situaciones críticas es del 100%. El mantenimiento en la línea de distribución eléctrica o en otros componentes de la infraestructura, pueden causar una interrupción del servicio. La tasa de disponibilidad máxima del Data Center es 99.741% del tiempo.

2.1.3. Tier III: mantenimiento concurrente

Las capacidades de un Data Center de este nivel le permiten realizar cualquier actividad planeada sobre cualquier componente de la infraestructura sin interrupciones en la operación. Actividades planeadas incluyen mantenimiento preventivo, reparaciones o reemplazo de componentes, agregar o eliminar componentes, realizar pruebas de sistemas o subsistemas, entre otros. Para infraestructuras que utilizan sistemas de enfriamiento por agua, significa doble conjunto de tuberías. Debe existir suficiente capacidad y doble línea de distribución de los componentes, de forma tal que sea posible realizar mantenimiento o pruebas en una línea y mientras que la otra atienda la totalidad de la carga. En este nivel, actividades no planeadas como errores de operación o fallas espontáneas en la infraestructura pueden todavía causar una

interrupción del Data Center. La carga máxima en los sistemas en situaciones críticas es de 90%.

Muchos Data Centers Tier III son diseñados para actualizarse a Tier IV, cuando los requerimientos del negocio justifiquen el costo. La tasa de disponibilidad máxima del Data Center es 99.982% del tiempo.

2.1.4. Tier IV: tolerante a fallas

Un Data Center de este nivel provee capacidad para realizar cualquier actividad planeada sin interrupciones en el servicio, pero además la funcionalidad tolerante a fallas le permite a la infraestructura continuar operando aún ante un evento crítico no planeado. Esto requiere dos líneas de distribución simultáneamente activas, típicamente en una configuración System + System. Eléctricamente esto significa dos sistemas de UPS independientes, cada sistema con un nivel de redundancia N+1. La carga máxima de los sistemas en situaciones críticas es de 90%. Persiste un nivel de exposición a fallas, por el inicio una alarma de incendio o porque una persona inicie un procedimiento de apagado de emergencia (EPO), los cuales deben existir para cumplir con los códigos de seguridad contra incendios o eléctricos. La tasa de disponibilidad máxima del Data Center es 99.995% del tiempo. (Institute, 2013)

2.2. Módulo de hardware Arduino.

2.2.1. Arduino.

Arduino es una plataforma de hardware de código abierto, basada en una sencilla placa de circuito impreso que contiene un microcontrolador de la marca “ATMEL” que cuenta con entradas y salidas, analógicas y digitales, en un entorno de desarrollo que está basado en el lenguaje de programación processing (lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java). El dispositivo conecta el mundo físico con el mundo virtual, o el mundo analógico con el digital controlando sensores, alarmas, sistemas de luces, motores, sistemas comunicaciones y actuadores físicos.

Hay muchos otros microcontroladores y plataformas disponibles donde las funcionalidades y herramientas son muy complicadas de programar. Arduino simplifica el proceso de trabajar con microcontroladores, ofrece algunas ventajas y características respecto a otros sistemas:

- Factible: Las placas Arduino son más accesibles y factibles comparadas con otras plataformas de microcontroladores.
- Multiplataforma: El software de Arduino funciona en los sistemas operativos Windows, Macintosh OSX y Linux. La mayoría de los entornos para microcontroladores están limitados a Windows.
- Ambiente de programación sencillo y directo: El ambiente de programación de Arduino es fácil de usar para los usuarios, Arduino está basado en el entorno de programación de processing (lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java) con lo que el usuario aprenderá a programar y se familiarizará con el dominio de desarrollo Arduino.
- Software ampliable y de código abierto: El software Arduino está publicado bajo una licencia libre y preparada para ser ampliado por programadores y desarrolladores experimentados. El lenguaje puede ampliarse a través de librerías de C++ y modificarlo a través del lenguaje de programación AVR C en el que está diseñado.
- Hardware ampliable y de código abierto: Arduino está basado en los microcontroladores ATMEGA168, ATMEGA328 y ATMEGA1280.

Los planos de los módulos están publicados bajo licencia creative commons, por lo que los diseñadores de circuitos pueden hacer su propia versión del módulo, ampliándolo u optimizándolo facilitando el ahorro. (Arduino, 2014)

En la tabla 2 se observa los diferentes tipos de Arduino que existen y sus características, estos módulos en su mayoría tienen la facilidad de poder trabajar con los microcontroladores ATmega168, ATmega256y ATmega328 (que en realidad lo que cambia es el espacio de memoria, es decir la cantidad de instrucciones que podemos introducir).

Tabla 2.

Familia Arduino

Prestaciones	**Arduino UNO**	**Arduino LiliPad**	**Arduino Mega 2560**	**Arduino Fio**	**Arduino ADK**	**Arduino PRO**	**Arduino Nano**
Microcontrolador	ATmega 328V	ATmegal6 8V ATmega3 28V	ATmega256	ATmega328P	ATmega2560	ATmega328V	ATmega 168 ATmega3 28
Voltaje de operación	5 V	2.7-55 V	5V	3.3V	5V	5 V	5 V
Voltaje de entrada (recomendado)	7-12 V	2.7-55 V	7-12V	3.35-12 V	7-12 V	7-12V	7-12 V
Voltaje de entrada (límites)	6-20V		6-20V		6-20V	6-20V	6-20 V
Voltaje de entrada para carga				3.7-7 V			
Pines Digitales I/O	14 (de las cuales 6 son salida PWM)	14 (de las cuales 6 proporcion an salida PWM)	54 (de las cuales 15 proporcionan salida PWM)	14 (de las cuales 6 proporcionan salida PWM)	54 (de las cuales 15 proporcionan salida PWM)	14 (de las cuales 6 proporcionan salida PWM)	14 (de las cuales 6 proporcion an salida PWM)
Pines de entrada analógica	6	6	16	8	16	6	8
Corriente DC por Pin I / O	40 mA	40 mA	40 mA	40 mA	40 mA	40 mA	40 mA
Corriente DC de 3.3V por pin	50 mA		50 mA		50 mA	50 mA	
Flash Memory	32 KB (ATmeg a328) de los cuales 0,5 KB utilizado por el gestor)	16 KB (de los cuales 2 KB utilizado por el gestor de arranque)	256 KB de los cuales 8 KB utilizado por el gestor de arranque	32 KB (de los cuales 2 KB utilizado por el gestor de arranque)	256 KB de los cuales 8 KB utilizado por el gestor de arranque	32 KB (ATmega328) de los cuales 0,5 KB utilizado por el gestor de arranque)	16 KB (ATmega1 68) o 32 KB (ATmega3 28)
SRAM	2 KB	1KB	8 KB	2 KB	8 KB	2 KB	(ATmegal 68) o 2 KB (ATmega3 28)
EEPROM	1KB	512 bytes	4 KB	1KB	4 KB	1KB	512 bytes(AT megal68) o 1KB
Velocidad del reloj	16 MHz	8 MHz	16 MHz	8 MHz	16 MHz	16 MHz	16 MHz

Nota. MHz= megahercio, mA= miliamperios.

Elaborado por: David Oña y Alexis Vaca

2.2.1.1. ArduinoUno

ArduinoUno es una de las placas más utilizadas en los proyectos tecnológicos de robótica y contiene un microcontrolador ATmega328 que tiene 32 KB de memoria flash para almacenar el código de los cuales 0,5 KB es utilizado por el gestor de arranque.

También dispone de 2 KB de SRAM y 1 KB de EEPROM, cuenta con 14 entradas y salidas digitales de los cuales 6 son utilizados como salidas PWM, contiene 6 entradas analógicas, un cristal de 16 MHZ oscilador, una conexión USB, un conector de alimentación, una cabecera ICSP, y el botón de reinicio. El diseño ha hecho posible que el microcontrolador se pueda conectar por medio de un cable USB a la computadora, con un adaptador de 9V o una batería. (Arduino, 2014)

Tarjeta ArduinoUno

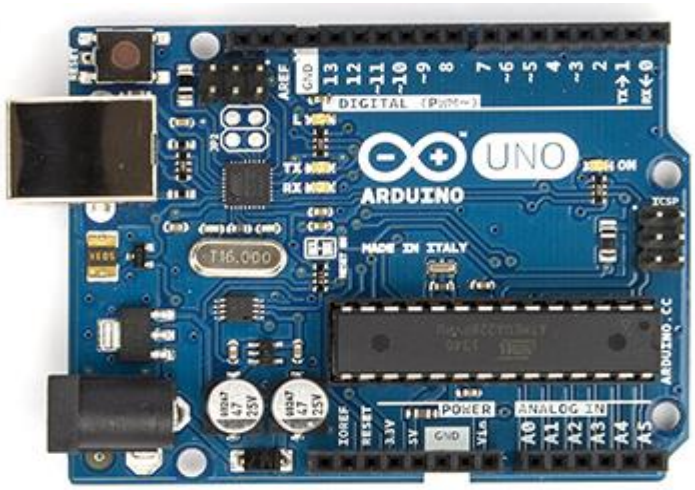


Figura 2. Arduino

2.2.1.2. Arduino LilyPad

La placa Arduino Lilypad es construida para el campo industrial del área textil y diseñada para coser con un hilo conductor prendas y accesorios dinámicos e

interactivos, se monta de manera igual las fuentes de alimentación, sensores y actuadores. Se basa en el microcontrolador ATmega168 o el ATmega328.

La placa Arduino LilyPad ha sido desarrollada por Leah Buechley y la versión comercial del kit por "SPARKFUN" electrónica. (Arduino, 2014)

Tarjeta Arduino Lilypod

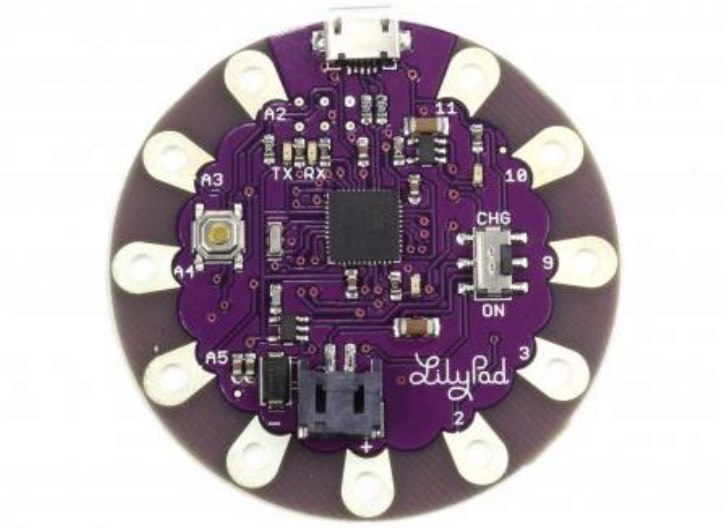


Figura 3. Arduino

2.2.1.3. Arduino Mega/2560

El Arduino Mega /2560 es una placa grande y más potente, electrónicamente está basado en el microcontrolador Atmega 2560 tiene 256 KB de memoria flash para almacenar código de los cuales 8 KB se utiliza para el gestor de arranque, 8 KB de SRAM y 4 KB de EEPROM. Tiene 54 pines digitales de entrada y salida de los cuales 15 se pueden utilizar como salidas PWM, además 16 entradas analógicas, 4 puertas seriales, un oscilador de 16MHz, una conexión USB, un conector de alimentación, una cabecera ICSP, y un botón de reinicio. Para empezar a trabajar con el microcontrolador basta con conectarlo a un computador con un cable USB o el poder con un adaptador AC-DC o batería. A diferencia de las demás tarjetas Arduino esta puede funcionar con un suministro externo de 6 a 20 voltios. (Arduino, 2014)

Tarjeta Arduino Mega 2560

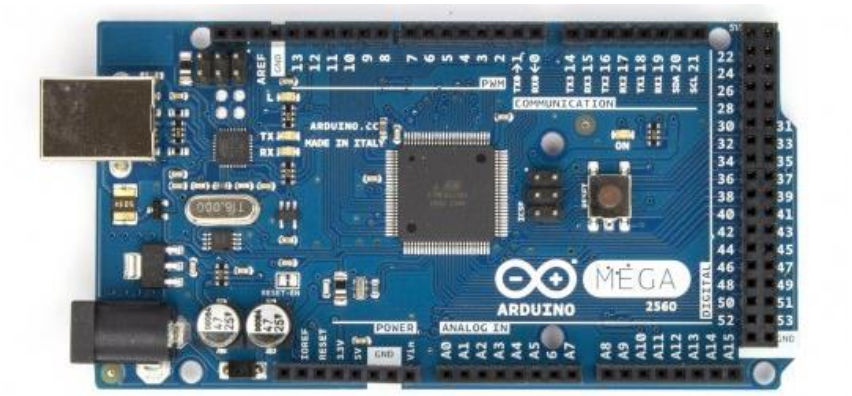


Figura 4. Arduino

2.2.1.4. Arduino Fio

La placa Arduino Fio está diseñada para aplicaciones inalámbricas, consta con un microcontrolador ATmega328P tiene 32 KB de memoria flash para el almacenamiento de código de los cuales 2 KB se utiliza para el gestor de arranque y dispone de 2 KB de SRAM y 1 KB de EEPROM. Cuenta con 14 pines de entrada y salida digital de las cuales 6 se puede utilizar como salidas PWM, 8 entradas analógicas, un resonador de a bordo, un botón de reinicio y dispone un circuito de carga a través de USB e incluye para sus conexiones una batería de polímero de litio.

Para la comunicación de la tarjeta el microcontrolador ATmega328P proporciona comunicación serie UART TTL, que está disponible en los pines digitales. (Arduino, 2014)

Tarjeta Arduino Fio

Figura 5. Arduino.

2.2.1.5. Arduino ADK

Arduino Mega ADK es una placa electrónica basada en el Atmega2560. Cuenta con una interfaz de host USB para conectar con los teléfonos basados en Android. Es compatible con los ejemplos del kit de desarrollo de accesorios de Android. Cuenta con 54 pines digitales de entrada y salida (de los cuales 14 se pueden utilizar como salidas PWM), 16 entradas analógicas, 4 UARTs (puertas seriales), un oscilador de 16MHz, una conexión USB, un conector de alimentación, una cabecera ICSP, y un botón de reinicio. El ADK está basado en el Mega 2560. Además de que tiene un circuito de host USB que permiten este foro para comunicarse con dispositivos USB, y darles la fuente de alimentación. (Arduino, 2014)

Las características adicionales que vienen con la versión R3 son: 1.0 de pinout: añadido SDA (Línea de datos) y SCL (Línea de reloj) pines para la comunicación TWI (dos hilos de interfaz serial) colocadas cerca al pin AREF y otros dos nuevos pasadores colocados cerca del pin RESET, la instrucción IOREF que permiten a los escudos para adaptarse a la tensión proporcionada de la placa y el segundo es un pin no conectado, que se reserva para usos futuros. (Arduino, 2014)

Tarjeta Arduino ADK

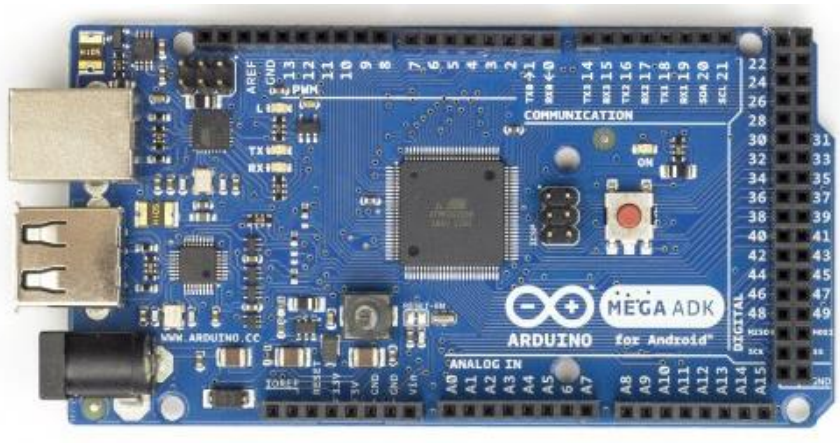


Figura 6. Arduino

2.2.1.6. Arduino Pro

El Arduino Pro es una placa electrónica basada en ATmega328. El Pro viene en 2 versiones 3.3V de 8 MHz y 5 V de 16 MHz. Cuenta con 14 pines digitales de entrada y salida (de los cuales 6 pueden utilizarse para salidas PWM), 6 entradas analógicas, un conector de alimentación de la batería, un interruptor de encendido, un botón de reinicio, y los agujeros para el montaje de un conector de alimentación, una cabecera ICSP, y cabezales de pin. 6 pines se puede conectar a un cable FTDI o tablero del desbloqueo de Sparkfun para proporcionar alimentación USB y la comunicación a la junta.

Tarjeta Arduino Pro

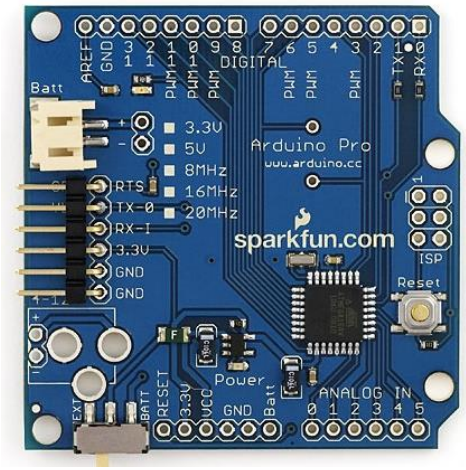


Figura 7. Arduino.

2.2.1.7. Arduino Nano

Arduino Nano es una pequeña tabla, basada en el ATmega328 (Arduino Nano 3.x) o ATmega168 (Arduino Nano 2.x). Tiene más o menos la misma funcionalidad de la Arduino Duemilanove, pero en un paquete diferente. Le falta sólo un conector de alimentación de CC, y funciona con un cable USB Mini-B en lugar de una normal. El Nano fue diseñado y está siendo producido por Gravitech.

Tarjeta Arduino Nano

Figura 8. Arduino

2.3. Sensores

Se denomina sensor a todo elemento que es capaz de transformar señales físicas en señales eléctricas, estas señales físicas pueden ser: temperatura, intensidad lumínica, distancia, aceleración, presión, fuerza, humedad, pH, etc. Las mediciones que realiza un sensor son de indicación directa o pueden estar conectados a un indicador (a través de un convertidor analógico a digital, una computadora y un display) de modo que los valores detectados puedan ser leídos. (Flores, 2013).

2.3.1. Sensor DHT11

El sensor de temperatura y humedad es denominado DHT11, cuenta con una salida de señal digital calibrada. Mediante el uso de la exclusiva señal adquisición digital la técnica y la tecnología de detección de temperatura y humedad, que garantiza una alta fiabilidad y excelente estabilidad a largo plazo. Este sensor incluye una medición de la humedad de tipo resistivo y un componente de medición de temperatura NTC que tiene un alto rendimiento en el microcontrolador, que ofrece una excelente calidad, respuesta precisa, anti interferencias capacidad y coste efectividad. En la tabla 3 se puede observar características de este tipo de sensor (Arduino, 2014)

Sensor DHT11

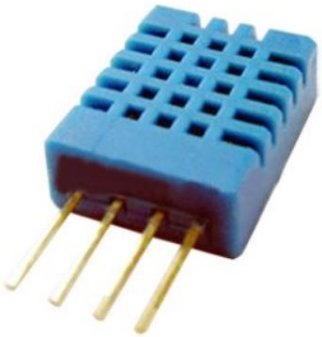


Figura 9. Arduino

Tabla 3.

Especificaciones técnicas DHT11

Parámetros	Condiciones	Mínimo	Típico	Máximo
Humedad				
Resolución		1%RH	1%RH	1%RH
			8 Bit	
Repetitividad			+ 1%RH	
Precisión	25°C		±4%RH	
	0-50 °C			±5%RH
Rango de medición	0°C	30%RH		90% RH
	25°C	20%RH		90% RH
	50°C	20%RH		80% RH
Tiempo de respuesta (segundos)	(63%) 25°C , 1m/s de aire	6 S	10 S	15 S
Resolución		1°C	1°C	1°C
		8 Bit	8 Bit	8 Bit
Precisión		±1°C		±2°C
Rango de medición		0°C		50°C

Nota. RH= humedad relativa, °C= grados Celsius.

Elaborado por: David Oña y Alexis Vaca

2.3.2. Sensor DHT22

El sensor de humedad y temperatura denominado DHT22 tiene señal digital de salida calibrada, asegura su fiabilidad y estabilidad en detección de temperatura y humedad

está conectada con 8 bits de un solo chip computador. Cada sensor de este modelo es compensado en temperatura y calibrado en la cámara de calibración precisa, la calibración de coeficiente se guarda en la memoria OTP (One Time Programmable), cuando el sensor está detectando.

Es de tamaño pequeño, bajo consumo y larga distancia de transmisión (20m), el DHT22 permitir que se adapten en todo tipo de ocasiones y en aplicaciones persistentes. Cuenta con una hilera de empaquetado con cuatro pines, por lo que la conexión es muy conveniente. En la tabla 4 se observa las especificaciones técnicas del sensor DHT22.

Tabla 4.

Especificaciones técnicas DHT22

Modelo	
Fuente de alimentación	3.3-6V DC
Señal de salida	señal digital a través de un solo bus
elemento de detección	Condensador de humedad Polímeros y DS1SB20 para detectar la temperatura
Rango de medición	humedad 0 -100% RH; temperatura -4 0 ~ 125 Celsius

Nota. DC= Corriente continua.

Elaborado por: David Oña y Alexis Vaca

Cuando el chip envía la señal de arranque, DHT22 cambia el modo-potencia-consumo de baja a running-mode. Cuándo el chip finaliza el envío de la señal de inicio, DHT22 enviará señal de respuesta de datos de los bits que reflejan la relación la humedad y la información de la temperatura al chip, como se muestra en la figura 10.

Proceso de comunicación

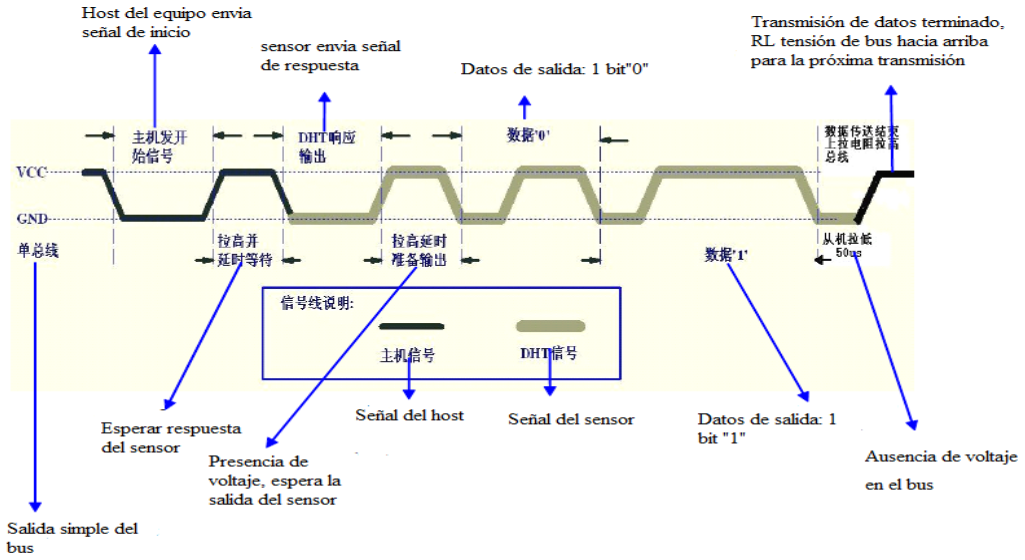


Figura 10. Envío de señal del sensor DHT22
Elaborado por: David Oña y Alexis Vaca

Sin señal de inicio del chip, el DHT22 no dará respuesta para indicar al chip que inicie. Una señal de inicio para los datos de respuesta de una sola vez que reflejan la humedad relativa y de la temperatura es información de DHT22 y cambiará a modo- potencia-consumo de baja cuando los datos recogidos indican que no recibe señal de inicio de chip de nuevo. (Adafruit, 2013)

Sensor DHT22

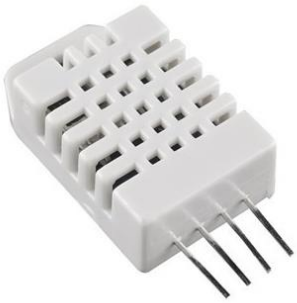


Figura 11. Arduino

En la tabla 5 se muestra una comparación de los sensores DHT11 y DHT22.

Tabla 5.

Comparación DHT11 Y DHT22

Modelo	DHT11	DHT22
Rango de medición de humedad	20-90 % HR	0-100 %HR
Rango de medición de temperatura	0 hasta 50 °C	-40 hasta 80 °C
Precisión de temperatura	±2 °C	±0.5 °C
Precisión de humedad	±5 % HR	±2 % HR

Nota. HR= humedad relativa, °C= grados Celsius.

Elaborado por: David Oña y Alexis Vaca

2.3.3. Sensor LM35.

El LM35 es un sensor análogo de temperatura integrado de precisión.

El LM35 no requiere ninguna calibración externa para proporcionar exactitudes típicas en un rango de -55 a +150 °C. La impedancia del rendimiento del LM35 es baja, tiene un rendimiento lineal, y la calibración que precisa la lectura o circuitería es relativamente sencilla. La serie de LM35 está disponible empaquetado y hermético TO-46 y en paquete plástico TO-92.

Características del sensor LM35:

- Calibrado directamente en ° Celsius.
- Factor de escala lineal + 10.0 mV/°C.
- Rango completo -55° a +150°C.
- Conveniente para las aplicaciones remotas.
- Bajo costo.
- Opera de 4 a 30 voltios.
- Menos de 60 μA corriente de desfogue.
- Baja salida de impedancia, 0.1 Ohm para 1 mA. (Moreno, 1999)

Sensor LM35

Figura 12. Arduino

En la tabla 6 se puede observar los parámetros de resistencia que tiene el sensor en los diferentes escenarios a los cuales puede estar expuesto.

Tabla 6.
Especificaciones técnicas LM35

Parametro	Valor
Temperatura Min	-40,-55
Temperatura Max	100, 110, 150
Precisión (+/-)	1, 50
Suministro Min (Volt)	4
Suministro Max (Volt)	30
Sensor Gain	10 mV/ Celsius (°C)
Pines	3

Nota. mV= mili voltios.

Elaborado por: David Oña y Alexis Vaca

2.3.4. Sensor de fugas de agua.

Un sensor de agua o inundación es un dispositivo que detecta las fugas de agua, empleando para ello una sonda de nivel que al detectar una variación del mismo emite una señal sonora avisando del peligro de inundación.

Los sensores de agua o inundación están compuestos por dos elementos:

La sonda o elemento sensor y el detector que analiza la señal procedente de la sonda y determina el estado de alarma (inundación) o reposo.

Algunos modelos disponen de indicadores luminosos independientes para los estados de funcionamiento y alarma, sonido interno para aviso en caso de alarma.

Sensor de fuga de agua

Figura 13. Honeywell

En la Tabla 7 se observa los diferentes tipos de medición para los diferentes sensores que existen.

Tabla 7.

Técnica de medición

	Sonda de neutrones	TDR (dominio del tiempo reflectometría)	Capacitivo
Problemas en la instalación: Aire	Problema menor	Principal problema	Principal problema
Campo de influencia: Radio	Seco: < 10 cm Húmedo: 10cm	0,5 a 2cm radio	0,5 a 2cm radio
Instalación en suelo disturbado	Si	Si	Si

Nota. cm= centímetros.

Elaborado por: David Oña y Alexis Vaca.

2.4. UPS

Para evitar el problema de falla de energía eléctrica se recurre a la alimentación ininterrumpida que permite que, en caso de un fallo momentáneo del suministro externo, se pueda seguir alimentando a los edificios y por tanto mantener su servicio ajeno a los cortes de suministro de energía.

UPS

Figura 14. New Abb

El edificio puede recibir el suministro de energía en corriente continua mediante una conexión al sistema propio de alimentación ininterrumpida del edificio (SAI o UPS) o de un módulo de alimentación que se instale exclusivamente para ella. (Julio Macías, 2002)

2.4.1. Tipos de UPS

- Serie 3 stand by: Los UPS serie 3 son de bajo costo y utilizados para estaciones de trabajo individuales y del hogar.
- Serie 5 interactivos: Los UPS serie 5 son recomendados para redes pequeñas de computadoras, cajeros automáticos aplicaciones profesionales y de oficina.
- Serie 9 on-line: Los UPS serie 9 ofrecen el más alto nivel de protección existente son recomendados para aplicaciones de misión crítica, grandes redes área financiera, telecomunicaciones, médica industrial, militares.

2.4.2. Ventajas por tipos de UPS

- Serie 3 stand by: bajo costo, fácil de instalar y operar batería para 10 a 15 minutos de respaldo tiempo de transferencia de 2 a 4 milisegundos.
- Serie 5 interactivos: tiene regulador de voltaje automático y tiempo de transferencia de 2 a 4 milisegundos
- Serie 9 on-line: eliminan todos los problemas eléctricos regulan constantemente voltaje y frecuencia trabajan libre de problemas con generadores tiempo de baterías extendidas hasta 8 horas tiempo de transferencia igual a cero.

2.4.3. Elementos principales del UPS

- Rectificador AC/DC: convierte la corriente alterna de la red en corriente continua para cargar la batería y alimentar al inversor
- inversor DC/AC: convierte la corriente continua del rectificador o de las baterías en corriente alterna para alimentar la carga o computadoras.
- Batería: almacena energía continua DC para alimentar el inversor durante la falta de energía de entrada (apagones).
- Bypass: circuito que conecta la entrada con la salida del UPS para permitir el paso directo de la energía en caso de falla de UPS.

2.5. Comunicación serie.

El puerto serie envía la información mediante una secuencia de bits. Para ello se necesitan al menos dos conectores para realizar la comunicación de datos, RX (recepción) y TX (transmisión), no obstante, pueden existir otros conductores para referencia de tensión, sincronismo de reloj, etc.

Los puertos de serie pueden referirse como UART. La UART (Universally Asynchronous Receiver/Transmitter) es una unidad que incorporan ciertos procesadores, encargada de realiza la conversión de los datos a una secuencia de bits y transmitirlos o recibirlos a una velocidad determinada.

El término TTL (transistor-transistor logic) en que la comunicación se realiza mediante variaciones en la señal entre 0V y Vcc (donde Vcc suele ser 3.3V o 5V). Los puertos RS-232 típicamente varían entre -13V a 13V. (LLamas, 2014)

Puerto serie

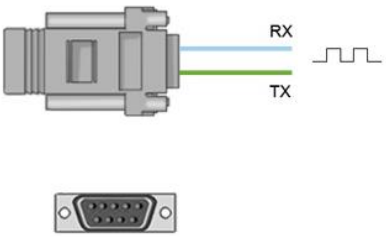


Figura 15. SerialPort

2.6. ArduinoGSMShield

ArduinoGSMShield permite que la placa de Arduino se conecte a Internet, realizar y recibir llamadas de voz y enviar y recibir mensajes SMS. Utiliza un módem de radio M10 de Quectel. Es posible comunicarse con la placa mediante comandos AT. La biblioteca GSM de Arduino tiene un gran número de métodos para la comunicación con GSMShield. Utiliza pines digitales 2 y 3 para el software de comunicación serie con el M10. El pin 2 se conecta al pin TX del M10 y el pin 3 a su pin RX. Pin PWRKEY del módem está conectado al pin 7 de Arduino.

El M10 es un módem cuatribanda GSM / GPRS que funciona en las frecuencias GSM850MHz, GSM900MHz, DCS1800MHz y PCS1900MHz. Es compatible con los protocolos TCP / UDP y HTTP a través de una conexión GPRS. GPRS enlace descendente de datos y la transferencia de enlace ascendente de velocidad máxima es de 85,6 kbps.

Para interactuar con la red celular se requiere una tarjeta SIM proporcionada por un operador de red. La tarjeta utiliza el pinout de la placa ArduinoUno. (Arduino, 2014).

2.6.1. GSM.

GSM es un estándar internacional para los teléfonos móviles. Es un acrónimo que significa Sistema Global para Comunicaciones Móviles. También se refiere a veces como 2G, el uso de GPRS para acceso a Internet, para el Arduino para solicitar páginas web, es necesario obtener el Access Point Name (APN) un nombre de usuario y contraseña del operador de red. GSM soporta llamadas salientes y entrantes de voz, sistema Mensaje simple (SMS o mensajes de texto), y la comunicación de datos (a través de GPRS). (HUIDOBRO MOYA, 2012)

2.6.2. GPRS

GPRS es una tecnología de conmutación de paquetes que significa General Packet Radio Service. Se puede proporcionar velocidades de datos ideales entre 56 a 114 kbit por segundo. Una serie de tecnologías como SMS depende de GPRS para funcionar. Con el GSMShield es posible aprovechar la comunicación de datos para acceder a Internet. Al igual que en las bibliotecas de ethernet y WiFi, la biblioteca GSM permite a Arduino actuar como cliente o servidor, utilizando http para enviar y recibir páginas web. (Arduino, 2014)

ArduinoGSMShield

Figura 16. Arduino

2.7. Correo electrónico.

El correo electrónico es una aplicación de internet cuya finalidad consiste en procurar la comunicación entre dos o más personas a través del intercambio de textos escritos digitalmente. Se trata de un sistema basado en un principio simple: un usuario de correo electrónico dispone de un espacio (buzón) en un computador conectado constantemente a la red (servidor) en el que se almacenan los mensajes enviados por otros usuarios. El correo electrónico se diferencia en un aspecto muy importante del resto de las aplicaciones de Internet como Telnet, FTP o web.

El computador del destinatario no tiene que estar conectado a la red al mismo tiempo que el del remitente, ya que el en el correo electrónico intervienen unas entidades denominadas Mail-Router servidores locales de correo electrónico que reciben y aceptan los mensajes para transmitirlos, posteriormente, a sus destinatarios finales. Para llevar a cabo esta operación los usuarios deben utilizar programas adecuados de gestión de correo que dominen el protocolo SMTP. (Delfa, 2006).

2.8. Aplicación en la plataforma Android.

Android es una plataforma para dispositivos móviles que contiene una fuente de software donde se incluye un sistema operativo, middleware (intercambio de información entre aplicaciones) y aplicaciones básicas para el usuario, con las siguientes características:

- Android está basado en el kernel de Linux
- Desarrollo rápido de aplicaciones, que sean reutilizables y portables entre diferentes dispositivos.
- Cuenta con su propia máquina virtual, Dalvik, que interpreta y ejecuta código escrito en Java.
- Permite la representación de gráficos 2D y 3D.
- Posibilita el uso de bases de datos.
- Soporta un elevado número de formatos multimedia.
- Servicio de localización GSM.
- Controla los diferentes elementos hardware: Bluetooth, WiFi, cámara fotográfica o de vídeo, GPS, acelerómetro, infrarrojos, etc.

- Cuenta con un entorno de desarrollo muy cuidado mediante un SDK disponible de forma gratuita.
- Ofrece un plugin para uno de los entornos de desarrollo más populares, Eclipse, y un emulador integrado para ejecutar las aplicaciones. (Libre, 2014)

La figura 17 representa la arquitectura de un dispositivo Android y las diferentes capas que se utiliza.

Arquitectura de Android

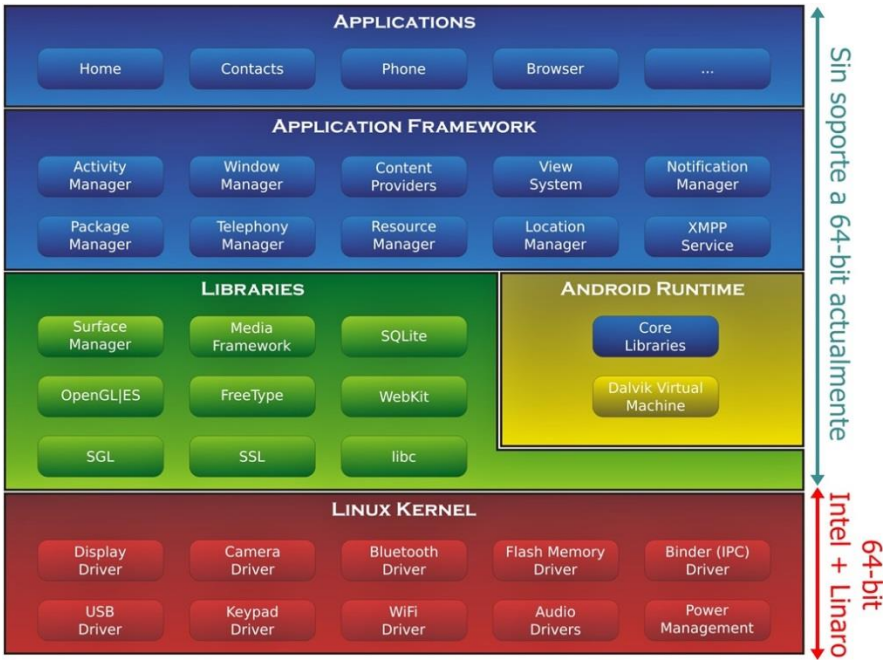


Figura 17. Android libre

La capa más cercana es la que corresponde al núcleo de Android, utiliza el núcleo de Linux 2.6 como una capa de abstracción para el hardware disponible en los dispositivos móviles, la cual contiene los drivers necesarios para cualquier componente de hardware pueda ser utilizado. Cabe recalcar que el fabricante es el encargado de crear las correspondientes librerías de control o drivers.

La siguiente capa corresponde con las librerías utilizadas por Android, estas han sido escritas utilizando C/C++ y proporcionan a Android la mayor parte de sus capacidades y características:

- La librería libc incluye todas las cabeceras y funciones según el estándar del lenguaje C. Todas las demás librerías se definen en este lenguaje.

- La librería SurfaceManager es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.
- Open GL/SL y SGL representan las librerías gráficas y por tanto sustentan la capacidad gráfica de Android.
- La librería MediaPlayer proporciona todos los códecs necesarios para el contenido multimedia soportado en Android (vídeo, audio, imágenes estáticas y animadas, etc.).
- A través de la librería SQLite, Android ofrece la creación y gestión de bases de datos relacionales.
- La librería WebKit proporciona un motor para las aplicaciones de tipo navegador, y forma el núcleo del actual navegador incluido por defecto en la plataforma Android.
- En este nivel también se encuentran las librerías de Android, entre las cuales se encuentran las CoreLibraries, estas están desarrolladas en lenguaje Java, y la máquina virtual Dalvik, que constituyen el framework de aplicaciones el cual representa fundamentalmente el conjunto de herramientas de cualquier aplicación.
- Dentro de este framework de aplicaciones podemos mencionar algunas de las librerías más importantes:
- Activity Manager: importante conjunto de APIs que gestiona el ciclo de vida de las aplicaciones en Android.
- Window Manager: gestiona las ventanas de las aplicaciones
- TelephoneManager: incluye todas las APIs vinculadas a las funcionalidades propias del teléfono (llamadas, mensajes, etc.).
- Content Providers: permite a cualquier aplicación compartir sus datos con las demás aplicaciones de Android. Por ejemplo, gracias a esta API la información de contactos, agenda, mensajes, etc. Será accesible para otras aplicaciones.
- View System: proporciona un gran número de elementos para poder construir interfaces de usuario (GUI), como listas, mosaicos, botones, check-boxes, tamaño de ventanas, control de las interfaces mediante tacto o teclado, etc. Incluye también algunas vistas estándar para las funcionalidades más frecuentes.

- Location Manager: posibilita a las aplicaciones la obtención de información de localización y posicionamiento.
- Notification Manager: permite que las aplicaciones usen un mismo formato para comunicar al usuario eventos que ocurran durante su ejecución, por ejemplo, una llamada entrante, un mensaje recibido, conexión Wi-Fi disponible, ubicación en un punto determinado, etc.
- La máquina virtual Dalvik ha sido optimizada y adaptada a las peculiaridades propias de los dispositivos móviles (menor capacidad de proceso, baja memoria, alimentación por batería, etc.) y trabajar con ficheros de extensión .dex (DalvikExecutables). Dalvik no trabaja directamente con el bytecode de Java, sino que lo transforma en un código más eficiente que el original, pensado para procesadores pequeños.
- Los ficheros .class de Java se compilan en ficheros .dex, de forma que cada fichero .dex puede contener varias clases. Después, este resultado se comprime en un único archivo de extensión .apk(AndroidPackage), el cual es el que se distribuirá a los dispositivos móviles.

Una vez vista la arquitectura de Android, se debe revisar cuáles son los componentes básicos de una aplicación:

- Activity: un componente Activity refleja una determinada actividad llevada a cabo por una aplicación y se asocia típicamente a una ventana o interfaz de usuario.
- BroadcastIntent Receiver: Se utiliza para lanzar una ejecución dentro de la aplicación actual cuando un determinado evento se produzca. El sistema lanzará la aplicación si es necesario cuando el evento monitorizado tenga lugar.
- Service: representa una aplicación ejecutada sin interfaz de usuario y que generalmente tiene lugar en segundo plano mientras otras aplicaciones son las que están activas en la pantalla del dispositivo.
- Content Provider: una clase que implemente este componente contendrá una serie de métodos que permite almacenar, recuperar, actualizar y compartir los datos de una aplicación ya sea en archivos o la base de datos SQLite.
- Cada uno de los componentes básicos de Android tiene un ciclo de vida bien definido; esto implica que el desarrollador puede controlar en cada momento

en qué estado se encuentra dicho componente, pudiendo así programar las acciones.

- El componente Activity, probablemente el más importante como se muestra en la figura 18 del ciclo de vida:

Ciclo de vida de una Aplicación Android

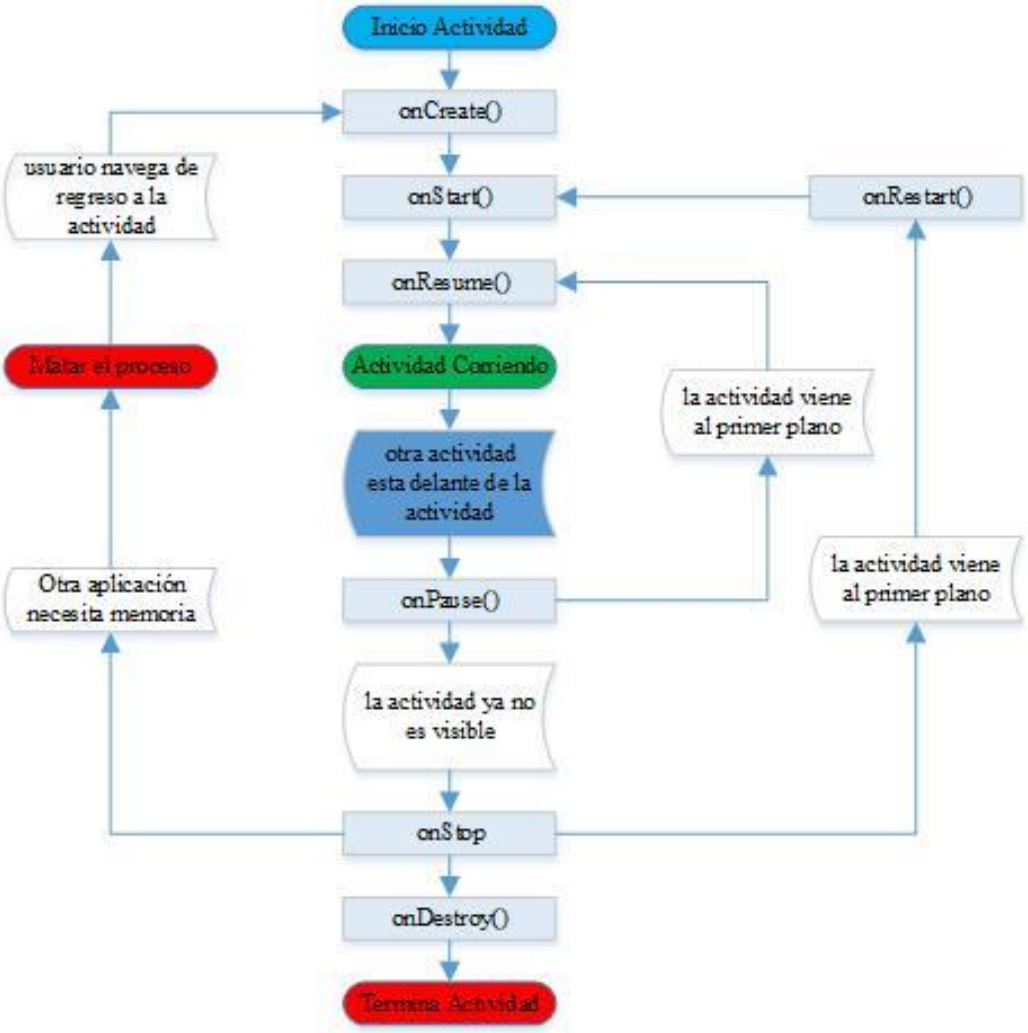


Figura 18.

Elaborado por: David Oña y Alexis Vaca

A continuacion se detallan los métodos del ciclo de vida que se muestra en la figura 18:

- onCreate(), onDestroy(): abarcan todo el ciclo de vida. Cada uno de estos métodos representan el principio y el fin de la actividad.

- onStart(), onStop(): representan la parte visible del ciclo de vida. Desde onStart() hasta onStop(), la actividad será visible para el usuario.
- onResume(), onPause(): delimitan la parte útil del ciclo de vida. Desde onResume() hasta onPause(), la actividad no es visible.

La mayoría de las medidas de seguridad entre el sistema y las aplicaciones deriva de los estándares de Linux 2.6. Por defecto, ninguna aplicación tiene permiso para realizar ninguna operación o comportamiento que pueda impactar negativamente en la ejecución de otras aplicaciones o del sistema operativo. La única forma de poder saltar estas restricciones impuestas por Android, es mediante la declaración explícita de un permiso que autorice a llevar a cabo una determinada acción.

Android tiene varios avances en tecnología y funcionalidad, cada uno de estos avances conlleva a una actualización en las cuales se arreglan los problemas y se agregan nuevas funciones. Actualmente Android se encuentra en la versión 5.02 (Lillipop). (Libre, 2014)

2.9. Servidor web.

Un servidor web es un programa que se ejecuta continuamente en un computador, manteniéndose a la espera de peticiones de ejecución que le hará un cliente o un usuario de Internet. Se encarga de contestar a estas peticiones de forma adecuada, entregando como resultado una página web o información de todo tipo de acuerdo a los comandos solicitados. Este intercambio es mediado por el navegador y el servidor que hablan el uno con el otro mediante HTTP. Se pueden utilizar varias tecnologías en el servidor para aumentar su potencia más allá de su capacidad de entregar páginas HTML; éstas incluyen scripts CGI, seguridad SSL y páginas activas del servidor (ASP).

Un servidor web básico cuenta con un esquema de funcionamiento muy simple, basado en ejecutar infinitamente el siguiente bucle:

1. Espera peticiones en el puerto TCP indicado (el estándar por defecto para HTTP es el 80).
2. Recibe una petición.

3. Busca el recurso.
4. Envía el recurso utilizando la misma conexión por la que recibió petición.
5. Vuelve al segundo punto. (Cavsi, 2014)

2.10. Metodología Scrum

Scrum es una metodología ágil de desarrollo de proyectos que toma su nombre y principios de los estudios realizados sobre nuevas prácticas de producción por HirotakaTakeuchi e IkujijoNonaka a mediados de los 80. Aunque surgió como modelo para el desarrollo de productos tecnológicos, también se emplea en entornos que trabajan con requisitos inestables y que requieren rapidez y flexibilidad; situaciones frecuentes en el desarrollo de determinados Sistemas de software. Jeff Sutherland aplicó el modelo SCRUM al desarrollo de software en 1993 en Easel Corporation (Empresa que en los macro-juegos de compras y fusiones se integraría en VMARK, luego en Informix y finalmente en Ascential Software Corporation). En 1996 lo presentó junto con Ken Schwaber como proceso formal, también para gestión del desarrollo de software en OOPSLA 96. Más tarde, en 2001 serían dos de los promulgadores del manifiesto ágil. En el desarrollo de software Scrum está considerado como modelo ágil por la Agile Alliance.

Scrum es una metodología de desarrollo muy simple, que requiere trabajo duro porque no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto.

Scrum es una metodología ágil, y como tal:

- Es un modo de desarrollo de carácter adaptable más que predictivo.
- Orientado a las personas más que a los procesos.
- Emplea la estructura de desarrollo ágil: incremental basada en iteraciones y revisiones.

VENTAJAS

- Alineamiento entre cliente y equipo
- Gestión regular de las expectativas del cliente

- Resultados a corto plazo
- Retorno de inversión (ROI).
- Equipo motivado
- Flexibilidad y adaptación a los cambios.
- Calidad del producto final. (Agiles, 2015)

En la tabla 8 se muestra una comparación de la metodología Scrum con las metodologías tradicionales existentes.

Tabla 8.

Comparación de metodologías

Metodología Scrum	Metodologías tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo de desarrollo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas o normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (menos de 10 integrantes) trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Mas roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Nota. Metodología Scrum vs metodologías tradicionales.

Elaborado por: David Oña y Alexis Vaca

2.10.1. Fases

El proceso de desarrollo Scrum se compone de cinco actividades importantes:

- Planes de lanzamientos: se determinan los requisitos iníciales y la visión del producto desde el punto de vista del cliente. Se consolida el Backlog del Producto.
- Distribución, revisión y ajuste de los estándares de producto: se auto-asignan las tareas a un integrante del equipo y se estiman los recursos necesarios para su desarrollo. El tamaño de las tareas debe ser el adecuado para realizar un seguimiento diario de su avance. Los miembros del grupo realizarán preguntas para solucionar todas sus dudas que serán resueltas por el propietario del producto. El producto resultante de esta planificación es el Sprint Backlog.
- Sprint: la fase de Sprint es el período en el cuál se lleva a cabo el desarrollo del software. El sprint se establece al comienzo del proyecto, y se mantiene durante toda la vida del mismo. Un Sprint consta de los siguientes sub-actividades:
    - Elaborar
    - Integrar
    - Revisar
    - Ajustar.

Esta fase no tiene una secuencia. A veces un elemento del backlog se tiene que desarrollar, integrar, y revisar cuando otras sólo debe ser revisado o ajustado.

- Revisión de Sprint: cada Sprint es seguido por una revisión de Sprint. Durante esta revisión, el software desarrollado en el Sprint anterior se revisa y si es necesario se le añaden nuevos ítems del backlog. El grupo de revisores pueden ser las partes interesadas del proyecto, gestores, desarrolladores y, en ocasiones los clientes, ventas y marketing.

Las actividades, y la revisión de Sprint se repiten hasta que el producto se considera listo para su distribución por los participantes en el proyecto. Luego, el proyecto pasa a la fase de cierre en que el producto se prepara para el lanzamiento y la distribución.

- Cierre: en esta fase tienen lugar las actividades de debugging, marketing y promoción. Al acabar esta fase el proyecto quedará cerrado. Es decir que en esta etapa se hace la entrega pactada de un sub producto: listo, revisado y probado. Puede incluir también documentación de usuario o técnica según se haya pactado.

2.10.2. Roles de Scrum

El método Scrum reconoce tres roles; la primera son las responsabilidades del dueño del producto el cuál define las características del producto, determina la fecha de lanzamiento y el contenido, asegura la rentabilidad del producto, prioriza las características según el valor de mercado, ajusta las características y las prioridades cada treinta días (según sea necesario), y acepta o rechazar resultados del trabajo.

El Scrum Master es un facilitador y líder de equipo, que trabaja en contacto estrecho con el dueño del producto. Sus responsabilidades son asegurar que el equipo se mantenga plenamente funcional y productivo; permitir la cooperación estrecha entre todos los roles y funciones; eliminar las barreras que obstaculicen el desarrollo del proyecto; proteger al equipo de las interferencias externas, y asegurar que el proceso se lleve a cabo correctamente, asegurando la concurrencia de los involucrados a las reuniones diarias de Scrum, a las revisiones de sprint y a las planificaciones de sprint. Durante las reuniones diarias de Scrum, debe saber qué tareas han sido completadas, cuáles se han iniciado, qué nuevas tareas se han descubierto y qué estimaciones cambiaron.

El equipo debe ser poli funcional, compuesto por siete miembros (más/menos dos). Su labor consiste en seleccionar el objetivo final de cada sprint, especificar los resultados del trabajo y llevarlo a cabo. Posee el derecho de realizar lo que sea dentro de los límites que impongan los lineamientos del proyecto– para alcanzar el objetivo final de un sprint. Debido a que opera como una “caja negra”, debe organizarse a sí mismo y a su trabajo, y debe preparar una demo de los resultados para exhibir ante el dueño del producto.

CAPÍTULO 3

DESARROLLO CONSTRUCCIÓN Y PRUEBAS DE LA APLICACIÓN

3.1. Diagrama de bloques del hardware del MTI

Para tener una visión general del hardware empleado en el diseño del MTI, se ha dividido en varios segmentos otorgando una mejor compresión del mismo. En la figura 19 se muestra el diagrama de bloques del hardware del MTI.

Diagrama de bloques

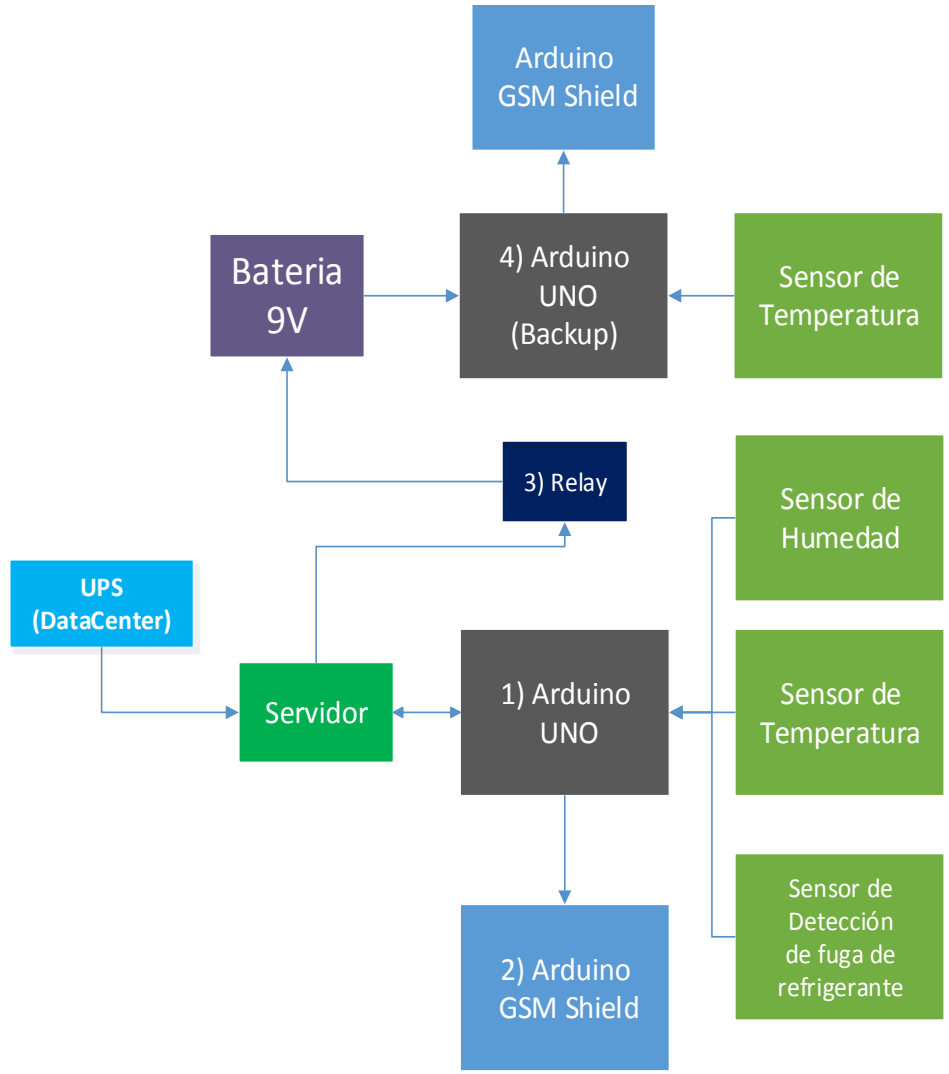


Figura 19.
Elaborado por: David Oña y Alexis Vaca

EL hardware del MTI consta de los siguientes componentes:

1. Tarjeta Arduino UNO

   Realiza la lectura de las mediciones de los sensores de las variables ambientales (humedad, temperatura) y estado de carga del UPS del Data Center en un intervalo de (2 segundos), posteriormente envía los parámetros al servidor el cual se encarga de procesarlos y enviarlos nuevamente.

2. Tarjeta Arduino GSM Shield

   Permite enviar (SMS) a través de datos adquiridos de la tarjeta ArduinoUNO.

3. Relay

   Permite activar automáticamente el módulo backup, en el caso que se agote el suministro de energía del módulo principal.

4. Tarjeta ArduinoUNO (backup)

   Realiza la lectura de las mediciones de los sensores de las variables ambientales (humedad, temperatura) en un intervalo de (5 minutos), envía los parámetros a la tarjeta Arduino GSM Shield, la cual envía mensajes de SMS, indicando "Falla total del sistema".

En la figura 20 se muestra el diagrama de conexiones del MTI con su respectiva conexión sobre la placa Arduino.

Diagrama de Conexiones módulo principal

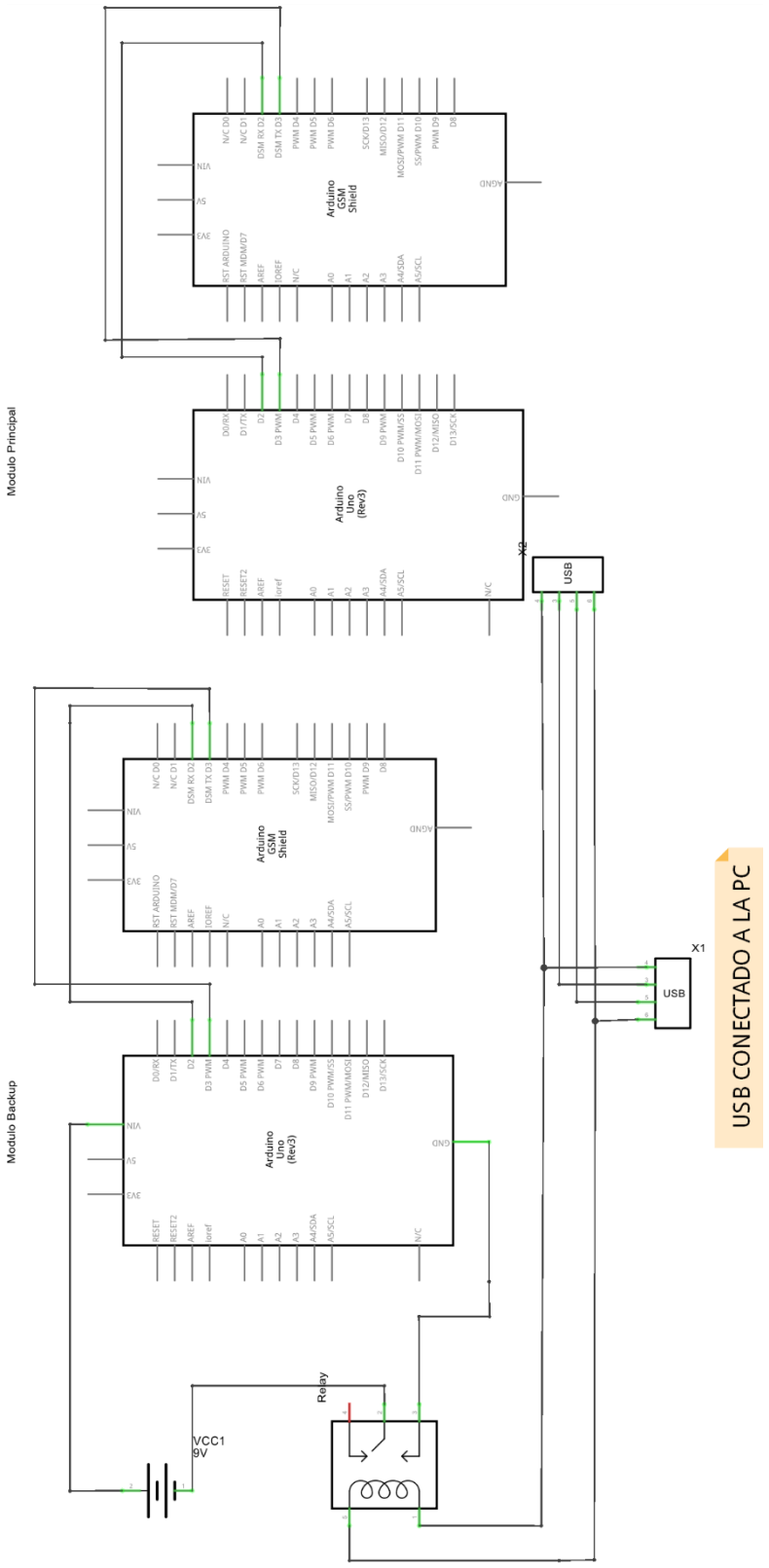


Figura 20.
Elaborado por: David Oña y Alexis Vaca

3.2. Sensores

Un sensor es un dispositivo diseñado para recibir información o estímulos externos y transformar las magnitudes físicas o químicas en magnitudes eléctricas, las cuales se puedan cuantificar y manipular. En la figura 21 se muestra el diagrama electrónico del sensor análogo de temperatura LM35 y el sensor digital DHT11 con la placa ArduinoUno.

3.2.1. Sensor de humedad y temperatura

Diagrama de los sensores

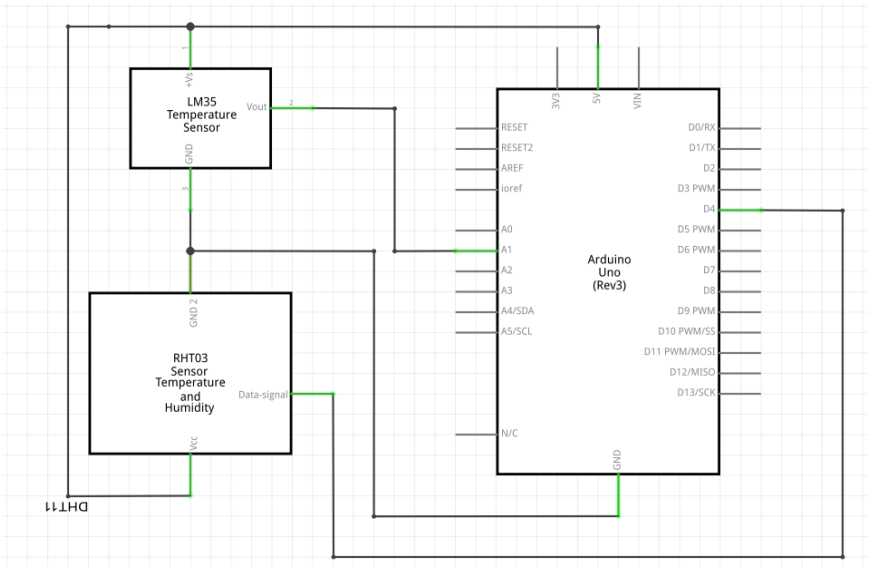


Figura 21.
Elaborado por: David Oña y Alexis Vaca

La conexión final del sensor de temperatura y humedad se realizó mediante un cable UTP cat 5e con la configuración cable directo T568A como se muestra en la figura 22.

Conexión final sensores

Figura 22.
Elaborado por: David Oña y Alexis Vaca

3.2.1.1. Sketch

Sketch

```
//Leer la temperatura tarda aprox 250 milisegundos
//El sensor toma hasta 2 segundos en hacer la lectura, es lento
float h =  dht.readHumidity(); //leemos humedad y lo ponemos en la variable h
float t = (5.0 * analogRead(1)*100.0)/1024.0;
```

Figura 23.
Elaborado por: David Oña y Alexis Vaca

En la figura 23 se muestra el sketch de ArduinoUno el cual detalla el código para la lectura del sensor digital de humedad (DHT11), que se realiza a través del uso de la función ".readHumidity()" perteneciente a la librería "DHT.h"

La lectura de sensor análogo de temperatura (LM35) se realiza a través de una lectura análoga del pin 1 y convertida a grados centígrados mediante una función matemática detallada a continuación:

La sensibilidad es 10.0 mV/ºC. Por lo que cada ºC que aumenta, se obtiene un aumento de 10mV en la salida.

La función AnalogRead lee la tensión del pin 1 y la divide en 1024 niveles, del modo que 0 Voltios equivale al nivel 0 y 5 Voltios a 1024, por lo que:

$$\frac{5 \text{ Voltios}}{1024 \ niveles} = 0{,}0048828125 \text{ V/Step}$$

Si analogRead devuelve 1, significa que la tensión es igual a 0,0048828125.

Para obtener el valor de temperatura en ºC se utiliza la siguiente formula:

$$\text{Temperatura} = \text{analogRead}(\text{pin\_1}) * (\frac{5}{1024})$$

El sensor de temperatura seleccionado fue el sensor análogo LM35 debido a su respuesta inmediata al detectar un cambio de temperatura en el ambiente del Data Center, además que no requiere ninguna calibración externa para proporcionar valores en un rango de -55 a +150 °C, tiene un rendimiento lineal, y la calibración que precisa

la lectura o circuitería es relativamente sencilla. La serie de LM35 está disponible empaquetado y hermético

Es importante conocer los datos característicos del sensor en el datasheet mostrado en el anexo B.

El sensor de humedad seleccionado fue el sensor digital DHT11, este sensor se caracteriza por tener la señal digital calibrada en laboratorio por lo que asegura una alta calidad y una fiabilidad a lo largo del tiempo, ya que contiene un microcontrolador de 8 bits integrado.

El protocolo de comunicación con ArduinoUno es a través de un único hilo (protocolo 1-wire) conectado en el pin digital D4

3.3. Sensor de fugas de refrigerante

La figura 24 muestra el diagrama de conexión con la placa ArduinoUno de los componentes electrónicos, para lograr la comunicación con un dispositivo de Hardware modificado con utilización de un led y una fotocelda encargada de captar la intensidad de la luz y ser leída por ArduinoUno.

Diagrama de conexión

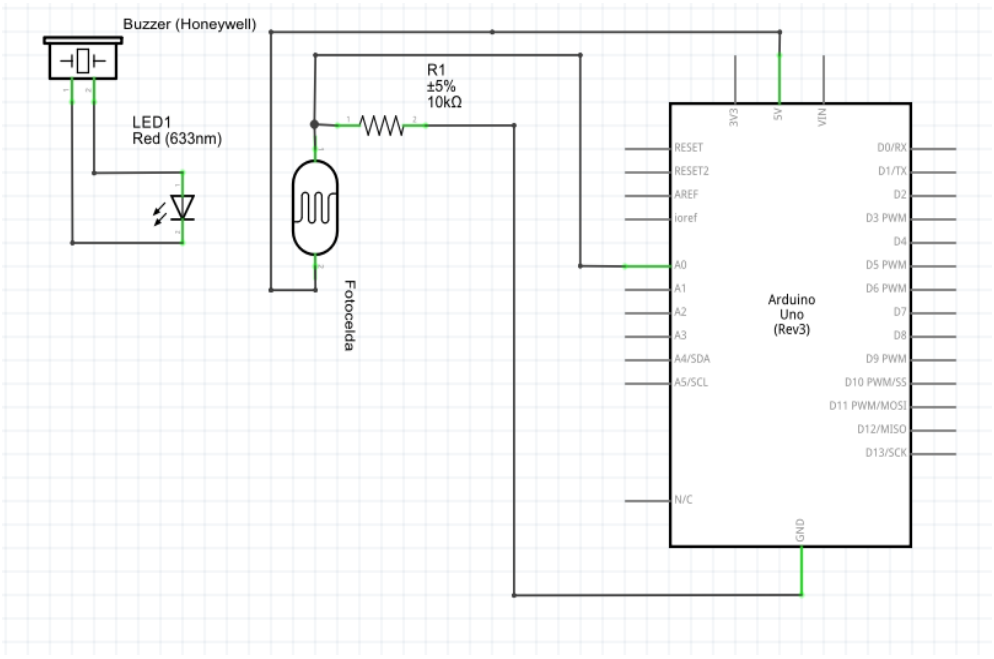
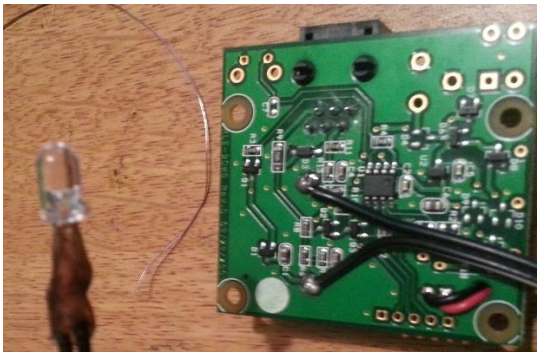


Figura 24.
Elaborado por: David Oña y Alexis Vaca

La comunicación del sensor de fugas de refrigerante se logró gracias la modificación de hardware del dispositivo Honeywell en el cual se realizó una conexión de un led en el buzzer, el cual se enciende cuando detecta presencia de líquidos en la sonda y emite sonidos. Como se muestra en la figura 25.

Conexión de sensores



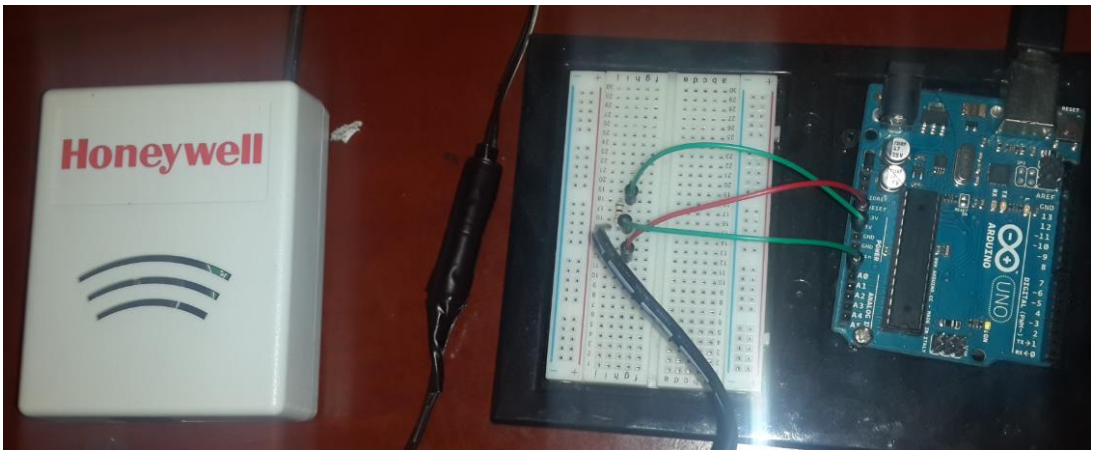


Figura 25.
Elaborado por: David Oña y Alexis Vaca

La siguiente línea de código Guarda el valor entero de la lectura análoga del sensor que está conectado en el pon 0 como se muestra en la figura 26.

Sketch sensor de agua

```
int val = analogRead(0);//leer el valor del sensor de A0 SENSOR DE AGUA
```

Figura 26.
Elaborado por: David Oña y Alexis Vaca

Para lograr una comunicación con Arduino se ubica una fotocelda frente al LED que está conectada con resistencia de 10k, la cual captura la luz emitida por el led, por lo que cambia su resistencia dependiendo de la cantidad de luz a la que es expuesta, en el pin análogo A0 de Arduino, el cual lee valores entre 0 y 1023 que son interpretados como disminución o aumento de presencia de luz.

3.4. Conexión serial con el UPS

El Data Center de la EPMAPS cuenta con un UPS marca PowerScale trifásico de media potencia de 50 kVA que proporciona protección de energía superior para las crecientes cargas en los servidores.

3.4.1. Principales características UPS PowerScale

PowerScale está disponible en siete rangos de potencias: 10, 15, 20, 25, 30, 40 y50 kVA. Tiene como principales prioridades el ahorro de costes y un 100% de operatividad, proporcionando rendimiento energético, escalabilidad flexible, alta disponibilidad y fácil mantenimiento. Esta solución "todo en uno", incluye una unidad de distribución de energía, un bypass manual de mantenimiento, gestión inteligente de batería y espacio para baterías internas. El sistema UPS trifásico independiente es la solución ideal para salas de servidores, redes, pequeños centros de proceso de datos, infraestructuras de telecomunicaciones y sanitarias, y para aplicaciones de banca e industriales.

PowerScale está equipado con una variedad de tipos de comunicaciones estándar y opcionales para comunicación en redes y aplicaciones de gestión.
Prestaciones estándar:

- RS 232 en puerto Sub-D9
- 4 contactos de entrada
- Fuente de 12 VCC
- RJ 45 para comunicaciones entre equipos

3.5. Conexión serial con el computador

El Puerto serie RS232 proporciona informaciones sobre todo el UPS y permite que el UPS sea conectado al computador. El software UPS Monitor permite al computador monitorizar la tensión de red y el estado del UPS en cada momento como se muestra en la siguiente figura:

La Figura 27 Muestra software WAVEMON del UPS PowerScale

Software del UPS

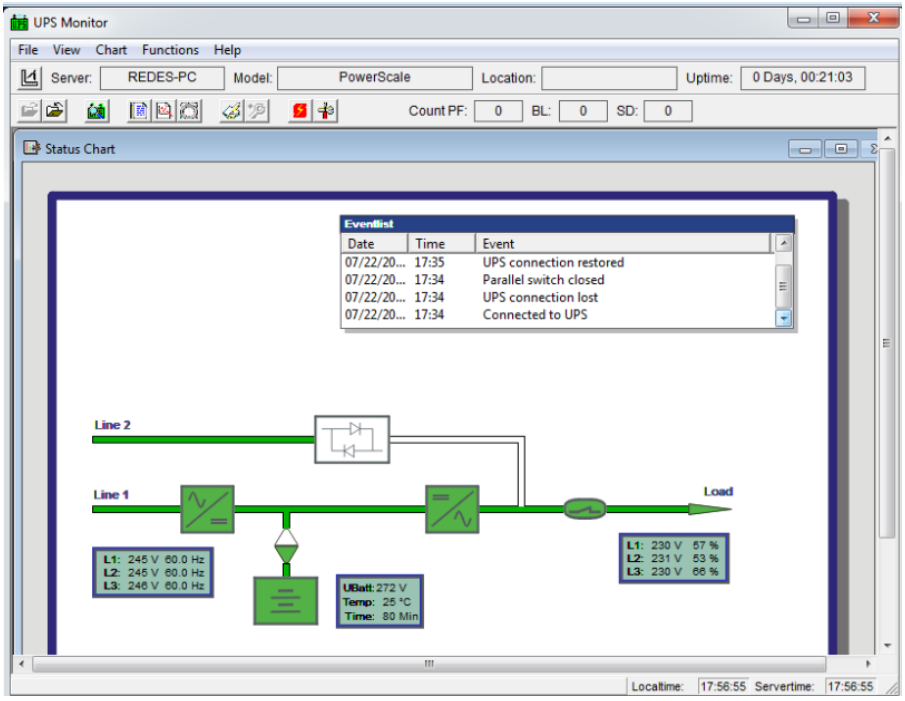


Figura 27.
Elaborado por: David Oña y Alexis Vaca

La Figura 28 Muestra como conectar un computador al UPS PowerScale mediante el puerto RS232

Conexión RS232

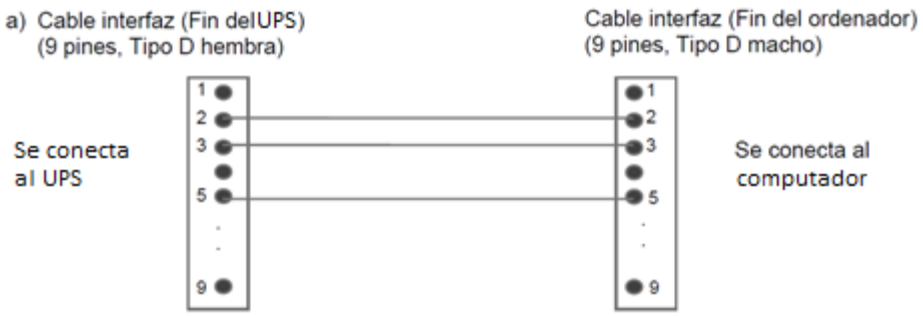


Figura 28.
Elaborado por: David Oña y Alexis Vaca

La Figura 29. Muestra la conexión RS232 mediante un cable de tipo nullmodem entre computador portátil al UPS PowerScale ubicado en el Data Center de la EPMAPS.

Conexión RS232 UPS

Figura 29.
Elaborado por: David Oña y Alexis Vaca

3.6. Módulo backup ArduinoUno

El Modulo backup entra en funcionamiento automáticamente en el caso que se perdiera el suministro de energía del UPS enviando SMS con el mensaje "Falla total del sistema" con los valores de temperatura y humedad actuales del Data Center a todo el personal de Infraestructura y Seguridad.

Módulo Backup

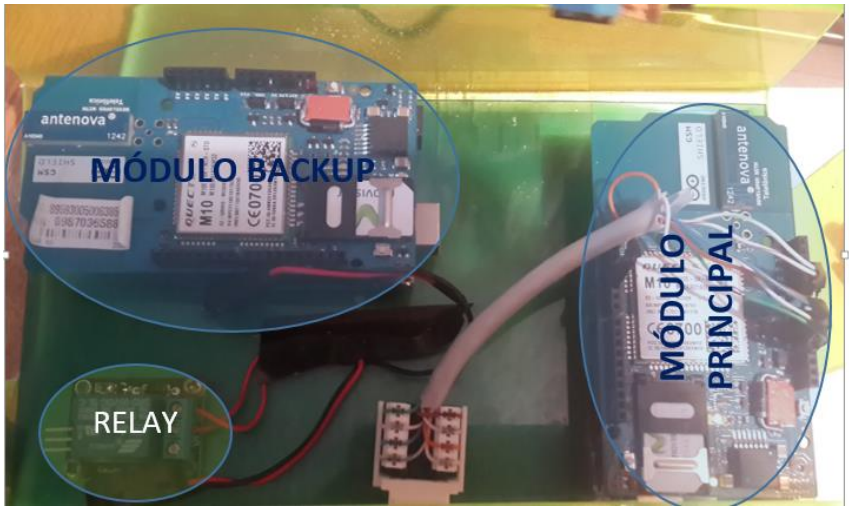


Figura 30.
Elaborado por: David Oña y Alexis Vaca

A continuación se muestra en la figura 31 el diagrama de componentes de hardware del MTI del módulo principal el cual es alimentado directamente por una conexión vía USB con el servidor del MTI, y el módulo backup que es alimentado con una batería stamina de 9V y cuanta con un sensor independiente de humedad y temperatura (DHT11) .

El cambio automático entre módulos se logró gracias a un relay que en este caso funciona como un interruptor ya que en el momento que se pierde el suministro de energía del cable USB permite el paso de corriente de la batería al módulo backup.

Diagrama MTI

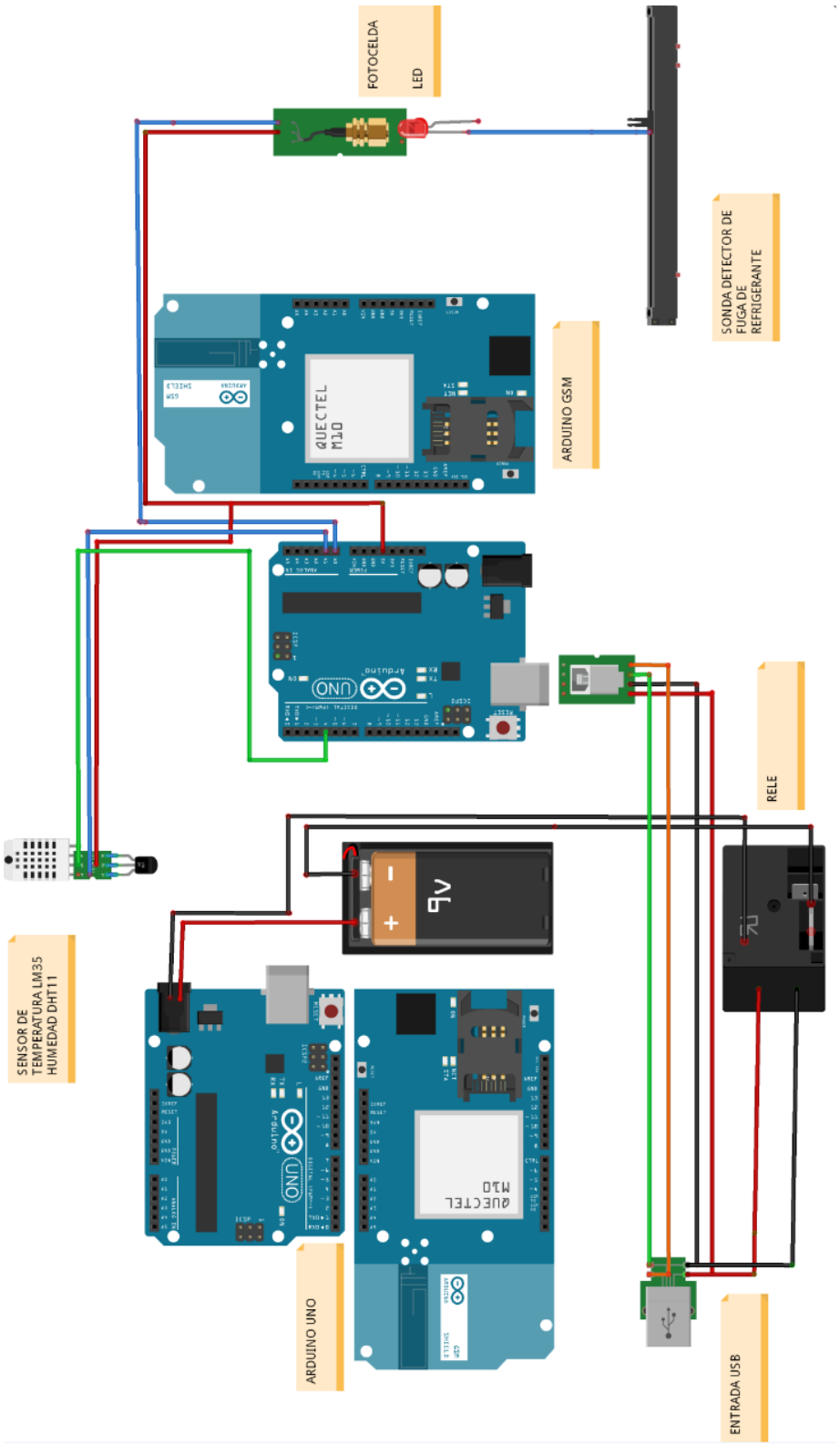


Figura 31.

Elaborado por: David Oña y Alexis Vaca

3.7. Diseño del software

3.7.1. Funciones

Una función es un bloque de código identificado por un nombre y que es ejecutado cuando la función es llamada. La declaración de una función incluye en primer lugar el tipo de datos que devuelve la función, y se clasifica en dos tipos de funciones E/S digitales y E/S analógicas

- E/S digitales
    - pinMode()
    - digitalWrite()
    - digitalRead()
- E/S analógicas
    - analogRead()
    - analogWrite() *(PWM)*

En la tabla 9 se muestra la Estructura, funciones y variables utilizadas en el IDE de Arduino.

Tabla 9.
Estructura, función y variables del IDE de Arduino

Estructura	**Funciones**	**Variables**
setup() (inicialización)	E/S digitales	Tipos de datos
loop() (bucle)	pinMode()	boolean (booleano)
Sintaxis	digitalWrite()	byte
;	digitalRead()	char (carácter)
{}		long (entero 32b)
// (comentarios en una línea)	E/S analógicas	float
/* */ (comentarios en varias líneas)	analogRead()	doublé

Estructura de control	analogWrite() (PWM)	String
If…else		
For	Tiempo	
switch case	millis()	
While	micros()	
do…while	delay()	
break (salida de bloque de código)		
Return	Matemáticas	
	min() (mínimo)	
	max() (máximo)	

Nota. PWM= pulse width modulation.

Elaborado por: David Oña y Alexis Vaca

3.7.2. Librerías

Las librerías facilitan la manipulación de los datos dentro del Sketch. Para utilizar una librería, se selecciona en el menú Sketch> Import Library.

Algunas librerías se incluyen con el software de Arduino y otras se pueden importar del directorio "arduino-x.x.x/libraries/" y utilizarlo en un sketch abierto.

3.7.2.1. Librería Dht11.h

Es una Librería para el uso del sensor digital de temperatura y humedad DHT11, entre las principales funciones se puede Obtener la temperatura de la muestra en distintas escalas: Kelvin, Celsius, Fahrenheit y la obtención de la humedad relativa de la muestra en tanto por ciento (%).Esta librería tiene las siguientes funciones;

- int getCelsius(): temperatura tomada en la última muestra, en la escala Celsius.
- float getFahrenheit(): temperatura tomada en la última muestra, en la escala Fahrenheit.
- float getKelvin(): temperatura tomada en la última muestra, en la escala Kelvin.

3.7.2.2. Librería Gsm

Es una librería que permite la conexión a una red / GRPS GSM con el módulo GSM Shield.

La librería permite a la placa Arduino hacer la mayoría de las operaciones que se pueden hacer con un teléfono GSM: realizar y recibir llamadas de voz, enviar y recibir SMS, y conectarse a Internet a través de una red GPRS.

La placa tiene un módem GSM que transfiere datos desde un puerto serie a la red GSM. El módem realiza operaciones a través de una serie de comandos AT. La biblioteca abstrae comunicaciones de bajo nivel entre el módem y la tarjeta SIM. Se basa en la biblioteca de software de serie para la comunicación entre el modem y Arduino.

La clase GSM permite la comunicación con el modem y se compone de las siguientes funciones:

- begin (): se conecta a la red GSM identificado en la tarjeta SIM.
- shutdown (): se desconecta de la red GSM identificado en la tarjeta SIM

La clase GSMVoiceCall permite la comunicación de voz a través del módem y se compone de las siguientes funciones:

- getVoiceCallStatus(): devuelve el estado de la llamada de voz (llamando, receiving call, hablar)
- ready (): informa si el comando de voz anterior ha ejecutado con éxito (devuelve 1 si se ha ejecutado correctamente, 0 si no).
- VoiceCall (): realiza una llamada de voz a un número determinado.
- answerCall (): acepta una llamada de voz entrante.
- hangCall (): colgar una llamada establecida o durante las llamadas entrantes.
- retrieveCallingNumber (): recupera el número que llama, y lo almacena.

La clase GSM_SMS facilita el envío y recepción de servicio de mensajes cortos (SMS).y se compone de las siguientes funciones:

- beginSMS (): identifica el número de teléfono para enviar un mensaje SMS
- ready (): obtiene el estado del GSMSMS.
- endSMS (): indica al módem que el mensaje SMS está completo y lo envía.
- disponible (): comprueba si hay un mensaje SMS en la tarjeta SIM para ser leídos, y notifica el número de caracteres en el mensaje.
- remoteNumber (): recupera el número de teléfono de un mensaje SMS entrante y lo almacena en una matriz llamada.
- read (): lee un byte de un mensaje SMS.
- write (): escribe un carácter de un mensaje SMS.
- print (): escribe una matriz de caracteres como un mensaje SMS
- flush (): borra la memoria del módem de los mensajes enviados una vez que todos los personajes salientes han sido enviados.

La clase GSMBand proporciona información acerca de la banda de frecuencia que el módem se conecta y se compone de las siguientes funciones:

- begin (): comprueba el estado del módem y lo reinicia.
- getBand (): obtiene y devuelve la banda de frecuencia que el módem está conectado actualmente.
- setBand (): establece la banda de frecuencia que el módem se conecta

3.7.3. Entorno de desarrollo Arduino

El entorno de desarrollo Arduino contiene un editor de texto para escribir código, un área de mensajes, una consola de texto, una barra de herramientas con botones para las funciones comunes, y una serie de menús como se muestra en la figura 32

Entorno de desarrollo Arduino

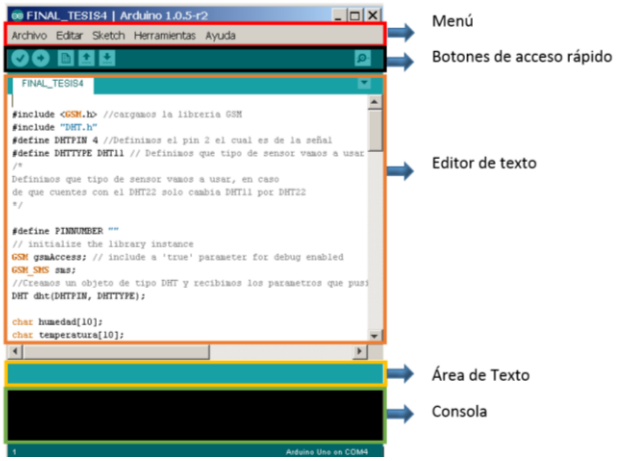


Figura 32.
Elaborado por: David Oña y Alexis Vaca

Para programar en el entorno de desarrollo Arduino utiliza lo que denomina "sketch" que es la unidad de código que se carga en y ejecutar en una placa Arduino.

En la parte superior encontramos el menú en el cual tenemos funciones como carga de archivos, edición del texto del código, carga de librerías, configuración, herramientas, etc. Entre las principales funciones de los botones de acceso rápido del IDE de Arduino tenemos los siguientes iconos:

Tabla 10.
Botones de acceso IDE Arduino

	Verificar Comprueba si el código contiene errores.
	Subir Compila el código y lo carga en la placa Arduino E / S.
	Nuevo Crea un nuevo sketch.
	Abierto Presenta un menú con todos los sketch guardados
	Guardar Guarda el sketch actual.
	Serial Monitor Abre una ventana de comunicación con la placa Arduino en la que se puede observar las respuestas que Arduino envía o recibe, siempre que el USB conectado.

Nota. Botones principales en el IDE de Arduino

Elaborado por: David Oña y Alexis Vaca

En el cuadro del editor de texto se escribe el código del programa el cual se compilara y se enviara a la tarjeta ArduinoUNO.

Finalmente, en el área de mensajes y la consola Arduino se mostrara información sobre si la consola está compilando, cargando y errores que se produzcan tanto en las diferentes líneas de código o en el IDE.

3.7.3.1. Programación en ArduinoUNO

Como se había mencionado en el capítulo 2 el microcontrolador de la placa Arduino UNO (Atmega328) se programa mediante el lenguaje de programación (basado en Wiring) y el entorno de desarrollo Arduino en un lenguaje propio basado en el lenguaje de programación de alto nivel Processing.

3.7.3.2. Estructura básica de un programa

La estructura básica del lenguaje de programación de Arduino se compone de dos partes necesarias:

- **setup()**
  Se inicia una sola vez cuando el programa empieza. Se utiliza para inicializar los modos de trabajo de los pines, o el puerto serie.
- **loop()**
  Es la que contiene el programa que se ejecutará cíclicamente.

Estructura básica de un programa en Arduino

```
void setup() //Primera Parte
{
  estamentos;
}
void loop() //Segunda Parte
{
  estamentos;
}
```

Figura 33.
Elaborado por: David Oña y Alexis Vaca

3.7.4. Programación de la tarjeta ArduinoUno

Para cumplir con el objetivo planteado se diseñó tres tareas para la tarjeta ArduinoUno y estos son:

- Obtención de valores de los sensores de Humedad y Temperatura y Fuga de refrigerante cada 1000 milisegundos
- Envío de Datos de las variables ambientales y estado del detector de fuga de refrigerante al computador.
- Envío de SMS si se produce una variación en los umbrales ambientales o carga del UPS.

En la figura 34 se muestra el diagrama de flujo de la programación que se implementó en la tarjeta ArduinoUno.

Diagrama de flujo Arduino

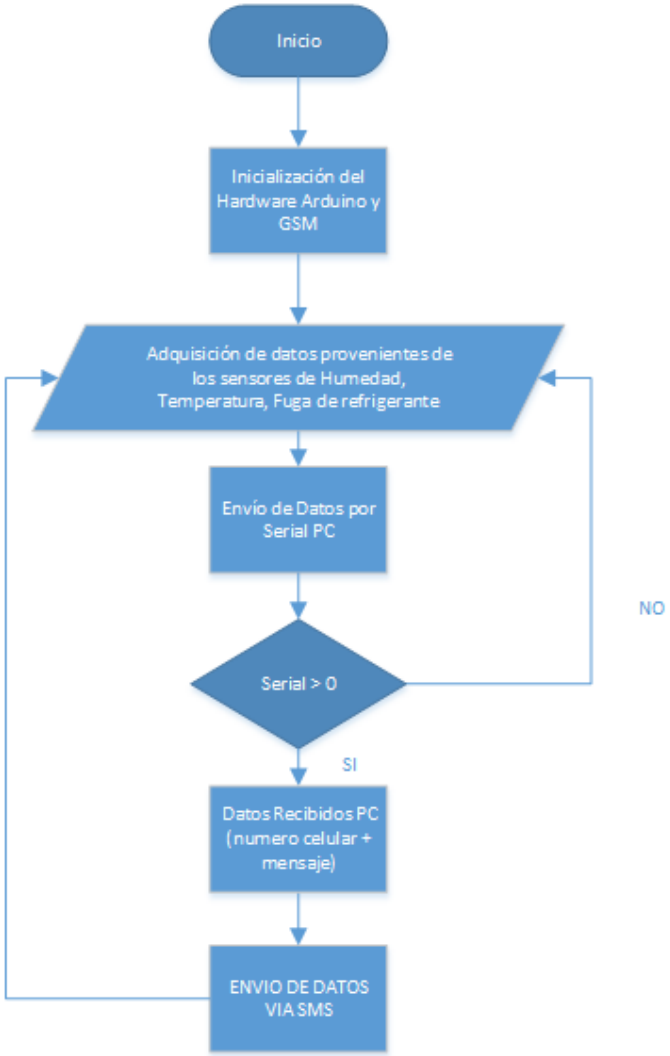


Figura 34.
Elaborado por: David Oña y Alexis Vaca

La estructura del lenguaje de programación de Arduino del sistema MTI está formada las siguientes partes:

1. Inicialización del Hardware Arduino y GSM por medio del cable USB conectado al servidor del MTI en el cual se ejecutan los siguientes procesos:
   - Creación de variables e invocación de librerías.
   - La función void setup () es llamada una vez al iniciar el sketch.
   - La función void loop () incluye el código que se ejecutará de forma continua.

2. La adquisición de datos de los sensores se realiza con una lectura análoga y digital de los siguientes pines:
   - Pin (0) análogo, sensor de temperatura (LM35)
   - Pin (1) digital, sensor de humedad (DHT11)
   - Pin (2) análogo, fotocelda (detector de fuga de refrigerante)
3. Una vez recolectados los datos de los diferentes sensores se crea un arreglo de caracteres de tamaño (40) el cual contiene los valores del sensor de temperatura, humedad y fuga de refrigerante y los envía por el puerto serial (virtual) al servidor MTI.
4. Si durante la ejecución de la función void loop () se detecta que se recibe datos del servidor MTI por medio de una lectura serial >0 ,el servidor MTI envía un arreglo de caracteres el cual contiene:
   - Nombre del sensor alarmado
   - Valor numérico del sensor alarmado
   - Número celular del destinatario

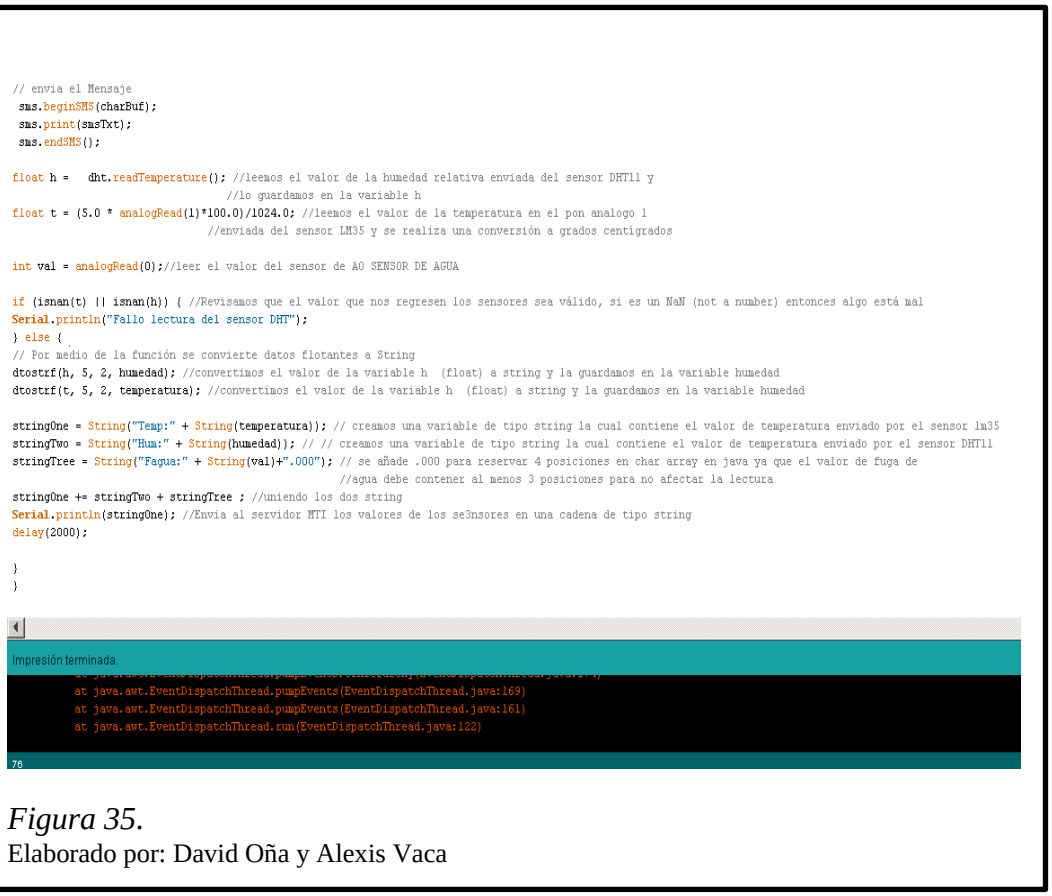
   Caso contrario Arduino sigue recolectando datos de los sensores ambientales y detección de refrigerante, enviando al servidor TMI.

5. Mediante la descomposición de la cadena de caracteres enviados por el servidor MTI, se procede mediante la función sendSMS() el envío de SMS al destinatario con el nombre del sensor alarmado y el valor del mismo, mediante el módulo ArduinoGSM.

   En la figura 35 se muestra el Sketch de funcionamiento del módulo principal de Arduino.

Sketch de funcionamiento


```
#include <GSM.h> //cargamos la librería GSM
#include "DHT.h" // cargamos la librería DHT.h
#define DHTPIN 4 //definimos el pin 4 (envió y recepción de datos) del sensor digital DHT11
#define DHTTYPE DHT11 // Definimos que tipo de sensor vamos a usar, DHT11 o DHT22
#define PINNUMBER ""
// Inicializar la instancia de la librería GSM
GSM gsmAccess;
GSM_SMS sms;
DHT dht(DHTPIN, DHTTYPE);//Creamos un objeto de tipo DHT y recibimos los parámetros

char humedad[10];  //creamos arreglo de tipo char de 10 caracteres
char temperatura[10]; //creamos arreglo de tipo char de 10 caracteres
String stringOne, stringTwo,stringTree ; // se crea variables de tipo string para los diferentes  sensores
String number; // creación de una variable de tipo string para definir el número de teléfono al cual se enviara el SMS
String smsTxt; //creación de una variable de tipo string la cual contiene el mensaje de texto
char charBuf[50];//creación de un arreglo de caracteres de tamaño 50, en el cual se almacenara el número de teléfono ,
               //nombre del sensor, valor medido enviado desde el servidor MTI

void setup() {
Serial.begin(9600);// Abre el puerto serie, establece la velocidad de datos a 9600 bps

dht.begin();//Iniciamos la variable del sensor DHT11

 Serial.println("SMS Messages Sender");

// Estado de conexión
boolean notConnected = true;

 // Inicio Modulo GSM
 // Si la tarjeta SIM tiene PIN, se  pasa como un parámetro para comenzar el envío de SMS


while(notConnected)
{
  if(gsmAccess.begin(PINNUMBER)==GSM_READY)
    notConnected = false;
  else
  {
    Serial.println("No conectado");
    delay(1000); // espera un segundo
  }
}

Serial.println("GSM Inicializado"); // envía un mensaje al monitor serial indicando que el módulo GSM está listo para enviar SMS

}
void loop() {

if (Serial.available()>0) // Enviar datos sólo cuando recibe datos del sistema MTI:
     {
        String inData = Serial.readString(); //lee  una secuencia de caracteres enviados  y las guarda en una variable de tipo string.

   delay(1000);// sensa variavles cada segundo

         number = inData.substring(0,12); // Obtiene una subcadena de una cadena y guarda en un avariable,
                                          //en este caso  extrae el número de teléfono enviado desde el MTI.
        smsTxt = inData.substring(13,35);  // Obtiene una subcadena de una cadena y guarda en un avariable,
                                          //en este caso  extrae el mensaje de texto  enviado desde el MTI.
     number.toCharArray(charBuf, 50);    // conversión de la variable number a una arreglo de caracteres.
     }
```



```
// envia el Mensaje
 sms.beginSMS(charBuf);
 sms.print(smsTxt);
 sms.endSMS();

float h =   dht.readTemperature(); //leemos el valor de la humedad relativa enviada del sensor DHT11 y
                                   //lo guardamos en la variable h
float t = (5.0 * analogRead(1)*100.0)/1024.0; //leemos el valor de la temperatura en el pon analogo 1
                                //enviada del sensor LM35 y se realiza una conversión a grados centigrados

int val = analogRead(0);//leer el valor del sensor de A0 SENSOR DE AGUA

if (isnan(t) || isnan(h)) { //Revisamos que el valor que nos regresen los sensores sea válido, si es un NaN (not a number) entonces algo está mal
Serial.println("Fallo lectura del sensor DHT");
} else {
// Por medio de la función se convierte datos flotantes a String
dtostrf(h, 5, 2, humedad); //convertimos el valor de la variable h  (float) a string y la guardamos en la variable humedad
dtostrf(t, 5, 2, temperatura); //convertimos el valor de la variable h  (float) a string y la guardamos en la variable humedad

stringOne = String("Temp:" + String(temperatura)); // creamos una variable de tipo string la cual contiene el valor de temperatura enviado por el sensor lm35
stringTwo = String("Hum:" + String(humedad)); // // creamos una variable de tipo string la cual contiene el valor de temperatura enviado por el sensor DHT11
stringTree = String("Fagua:" + String(val)+".000"); // se añade .000 para reservar 4 posiciones en char array en java ya que el valor de fuga de
                                                    //agua debe contener al menos 3 posiciones para no afectar la lectura
stringOne += stringTwo + stringTree ; //uniendo los dos string
Serial.println(stringOne); //Envia al servidor MTI los valores de los se3nsores en una cadena de tipo string
delay(2000);

}
}
```

```
Impresión terminada.
        at java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:169)
        at java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:161)
        at java.awt.EventDispatchThread.run(EventDispatchThread.java:122)
76
```

Figura 35.
Elaborado por: David Oña y Alexis Vaca

3.8. Desarrollo web

En la figura 36 se muestra las clases utilizadas para el desarrollo de la aplicación web donde se observa que la clase cls_acceso depende de la clase cls_conexion la clase cls_consulta no puede existir sin la clase cls_monit_var y la clase cls_consulta no puede existir sin la clase cls_conexion.

Diagrama de clases

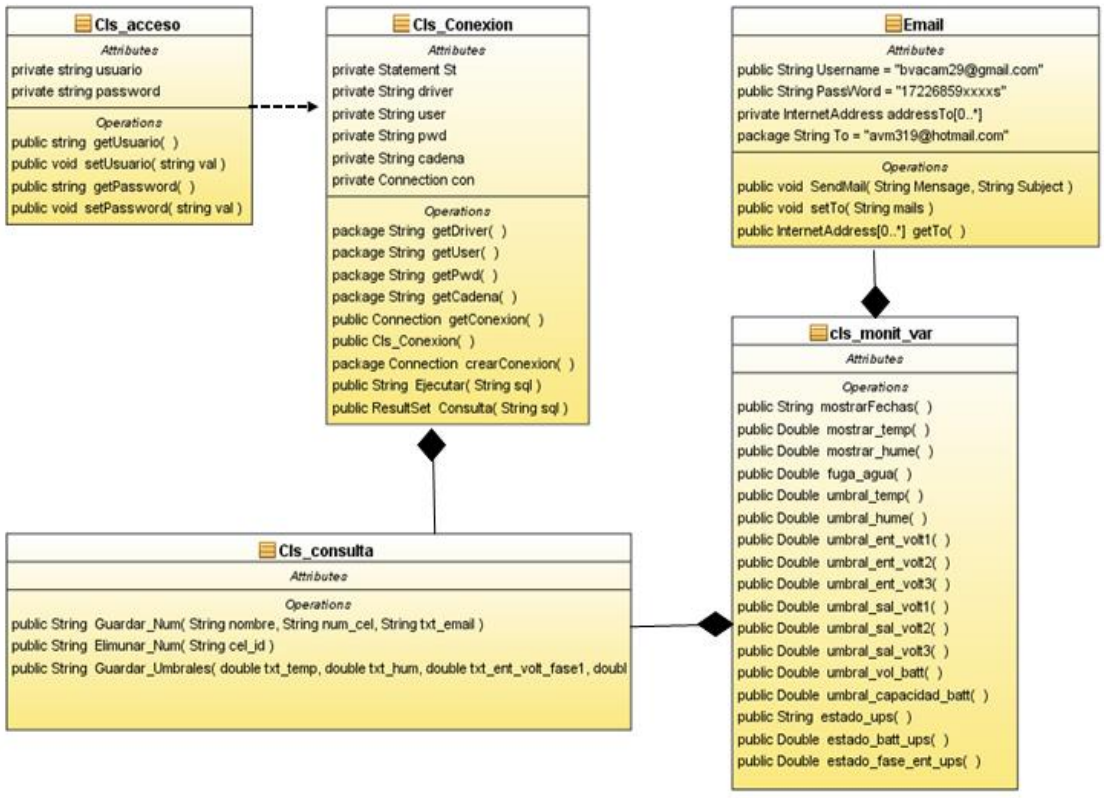


Figura 36.
Elaborado por: David Oña y Alexis Vaca

Para iniciar con el desarrollo de la aplicación web es necesario elegir una metodología que mejor se adapte a las características del proyecto de software, para lo cual se eligió Scrum, por las características explicadas en el Capítulo 2.

La metodología Scrum clasifica a todas las personas que intervienen o tienen interés en el desarrollo del proyecto, a continuación se definen los roles que tienen cada uno de las personas que intervienen.

Propietario del producto: Ing. Cristóbal Morocho, quién proporcionó la información y funcionalidades del sistema, es el encargado de crear y mantener el Product Backlog asegurándose que se realicen con anterioridad aquellos requerimientos con mayor prioridad.

Equipo desarrollador: Los autores de la tesis, quiénes son los encargados de desarrollar las funcionalidades del software. El Equipo crea el Sprint Backlog eligiendo qué hacer

durante el siguiente Sprint. Sin embargo su creación se hace con la supervisión del Propietario del producto,

Scrum Master: El tutor de tesis, responsable de hacer un seguimiento al equipo desarrollador, para que se adhieran a las normas que indica la metodología, además de realizar las pruebas funcionales e informar los resultados al equipo desarrollador.

Estructura Scrum

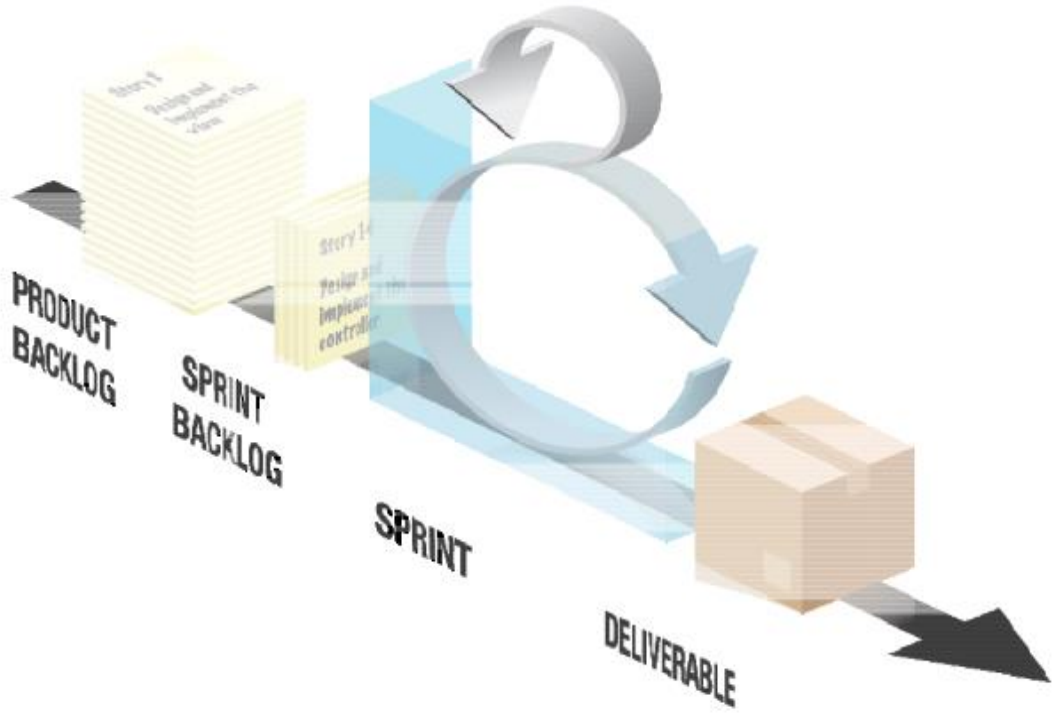


Figura 37. Metodologías Rápidas

3.8.1. Product BlackLog

En la tabla 11 se observa los requerimientos que tendrá el sistema; la misma puede variar en el transcurso de la iteración.

Tabla 11.
Product BlackLog

Requisito 1	Página inicial de acceso a la aplicación	Prioridad: alta
La página inicial debe contener un panel principal de notificaciones donde se muestre los eventos que sucedan en el Data Center.		
Requisito 2	Diseño de la Interfaz del portal	Prioridad: alta
El portal debe tener un diseño llamativo y de fácil uso para quien acceda. En la parte inferior de la página deben existir enlaces al sitio web de la EPMAPS.		
Requisito 3	Control de acceso a la aplicación	Prioridad: alta
Dentro de la página inicial del portal tendrá un enlace donde existirá un control de acceso: usuario y contraseña que serán validados con la base de datos, en caso de no existir el usuario no se permitirá el registro de un nuevo usuario, solo el usuario administrador puede realizar el registro de un nuevo usuario.		
Requisito 4	Usuario no registrado	Prioridad: media
El usuario no registrado no podrá ver información del portal.		
Requisito 5	Usuario Registrado	Prioridad: alta
El usuario registrado solo tiene acceso a visualizar las variables ambientales y el estado de carga del UPS.		
Requisito 6	Usuario Administrador	Prioridad: alta
El usuario administrador tendrá la posibilidad de: Agregar un nuevo usuario Eliminar un usuario existente Cambiar los parámetros de funcionamiento tanto del UPS como de las variables ambientales.		
Requisito 7	Generación de Informes	Prioridad: media
Generar Reportes.		

Nota. Requisitos metodología Scrum

Elaborado por: David Oña y Alexis Vaca

3.8.2. Sprint del Sistema

Durante el Sprint, el equipo se encarga de transformar el Sprint Backlog en un producto funcional y listo para entregar, pasando las fases de análisis, diseño, desarrollo y pruebas para todos los requerimientos que haya que implementar.

El Sprint Backlog.- Es un documento que nace a partir del Product Backlog; diseñado en forma de un listado de requerimientos que se prevé que se terminarán durante el Sprint.

Tabla 12.
Sprint Backlog del Sprint

	Descripción	Prioridad
Requisito 1	Página inicial de acceso a la aplicación	Alta
Requisito 2	Diseño de la Interfaz portal	Alta
Requisito 3	Control de acceso a la aplicación	Alta
Requisito 4	Usuarios no registrados	Alta
Requisito 5	Registro de usuarios	Alta
Requisito 5	Usuario Administrador	Alta

Nota. Sprint Backlog: listas de tareas de iteración.
Elaborado por: David Oña y Alexis Vaca

Una vez determinado el Sprint Backlog, se define las tareas necesarias para poder completar cada uno de los requisitos seleccionados. En la tabla 13 se detallan las tareas de cada una de ellas.

3.8.3. Tareas del Sprint

Tabla 13.
Tareas de Sprint

	Tarea	**Responsables**	**Estimación**
Tarea 1	Diseño de la Base de Datos	David Oña y Alexis Vaca	12 h
Tarea 2	Diseño de la interfaz de la página de inicio.	David Oña y Alexis Vaca	12 h
Tarea 3	Diseño de la arquitectura	David Oña y Alexis Vaca	08 h

	Tarea	Responsables	Estimación
Tarea 4	Programación de conexiones con la base de datos.	David Oña y Alexis Vaca	06 h
Tarea 5	Programación de registro de usuario.	David Oña y Alexis Vaca	12 h
Tarea 6	Programación de las umbrales de monitoreo y carga del UPS	David Oña y Alexis Vaca	12 h
Tarea 7	Pruebas	Ing. Cristobal Morocho, David Oña y Alexis Vaca	8 h
		Total Horas Trabajo	**78 h**

Nota. Tiempo estimado de tareas que se realizan con Scrum

Elaborado por: David Oña y Alexis Vaca

Una vez que se ha tomado en cuenta estas tareas se debe revisar el seguimiento diario de actividades hechas en el sprint en la tabla 14 se muestra la lista de tareas y sus estados.

3.8.4. Lista de tareas de la iteración

Tabla 14.

Lista de tareas de la iteración

Tarea	Tipo	Estado
Diseño de la base de datos	Prototipo	Terminado
Diseño de la arquitectura	Prototipo	Terminado
Página inicial de acceso a la aplicación	Prototipo	Terminado
Diseño de la interfaz del portal	Prototipo	Terminado
Control de acceso a la aplicación	Codificación	Terminado
Usuario no registrado	Prototipo	Terminado
Usuario registrado	Prototipo	Terminado
Usuario administrador	Prototipo	Terminado
Generación de informes	Codificación	Terminado
Pruebas de la versión	Pruebas	Terminado

Nota. Proceso de tareas en ejecución.

Elaborado por: David Oña y Alexis Vaca

3.8.5. Ejecución del sprint

A continuación se detalla cada uno de las tareas de la tabla 14

3.8.5.1. Diseño de la base de datos

El diseño de la base de datos es la parte principal para empezar con el desarrollo de cualquier aplicación, ya que un buen diseño de esta definirá el óptimo funcionamiento y éxito del software. La base esta creada sobre el gestor de base de datos PostreSql.

La base de datos creada consta de cuatro tablas con su respectiva clave primaria y atributos que la identifican; como se muestra en la figura 38 las tablas tb_arduino y tb_ups_dtc mantiene una relación de uno a uno; mientras que las tablas tb_usuario y tb_valores_sensores son independientes.

Modelo conceptual

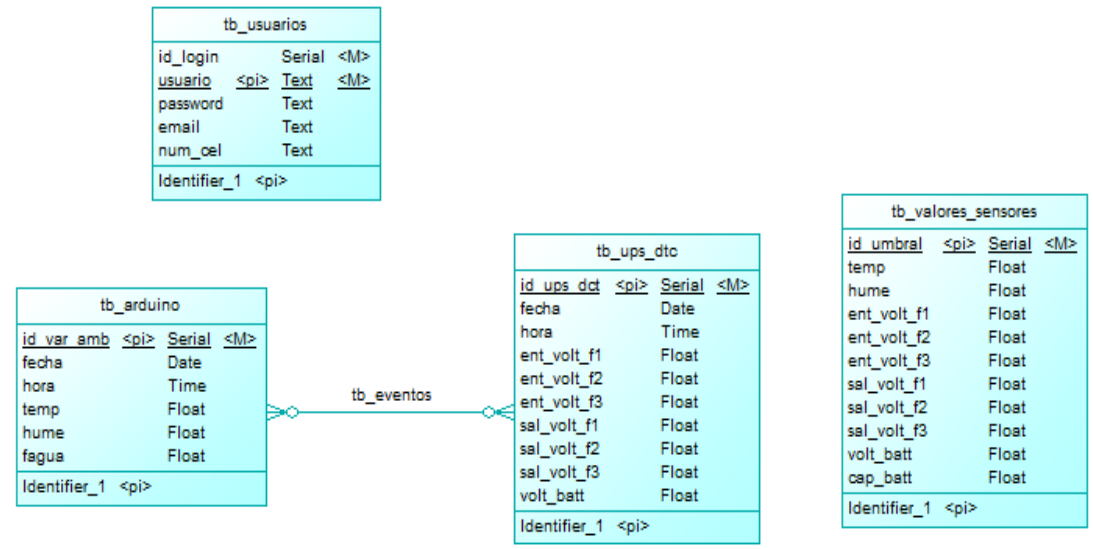


Figura 38.
Elaborado por: David Oña y Alexis Vaca

En la figura 39 se observa el modelo físico de la base de datos

Modelo Físico

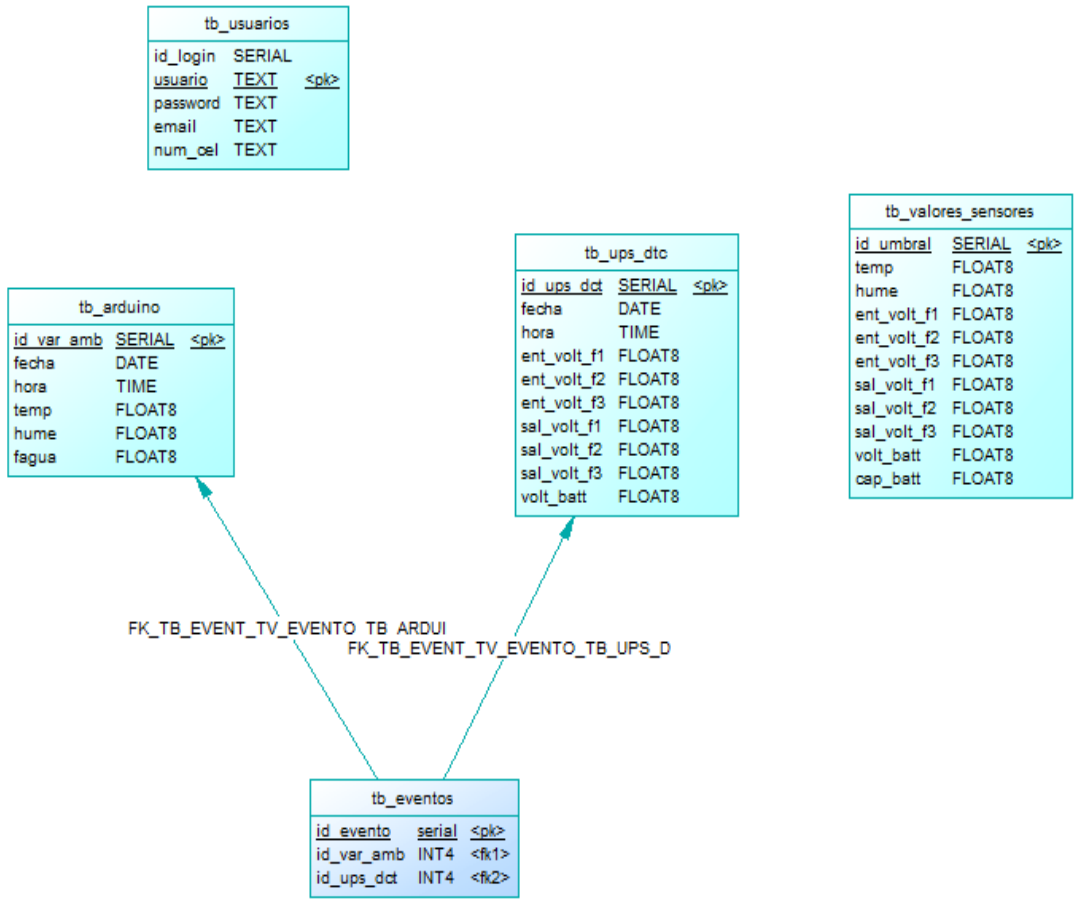


Figura 39.
Elaborado por: David Oña y Alexis Vaca

3.8.5.2. Diccionario de base de datos

Se hace uso de tablas con las que cuenta la base de datos con la descripción de los campos incluyendo información como el nombre de los campos, el tipo de dato y la descripción de la información que se guardará.

La tabla tb_usuarios contiene un conjunto datos de los usuarios y permisos para acceder al sistema MTI como se muestra en la tabla 15.

Tabla 15.
Diccionario de datos tabla usuarios

Nombre	tb_usuarios			
Descripción	Almacena la información de usuarios para envío de alertas y acceso al MTI			
Primary Kev	usuario			
Kev	**ColumnName**	**Data Type**	**NotNulI**	**Descripción**
	id_login	Serial	Yes	Identifica si el usuario registrado pertenece al grupo de administradores o clientes.
PK	Usuario	Text	Yes	Identificador único del usuario al cual hace referencia
	Password	Text	Yes	Contraseña del usuario para acceso al MTI.
	Email	Text	Yes	Dirección de correo electrónico para él envió de alertas.
	num_cel	Text	Yes	Numero de celular del usuario para el envío de Alertas,

Nota. Descripción de la tabla tb_usuarios de la base de datos.

Elaborado por: David Oña y Alexis Vaca

La tabla tb_arduino contiene un conjunto datos de los sensores ambientales y fuga de refrigerante enviados por Arduino UNO al sistema MTI. Como se muestra en la tabla 16.

Tabla 16.
Diccionario de datos tabla Arduino

Nombre	tb_arduino			
Descripción	Almacena información enviada por los sensores ambientales y de fuga de refrigerante			
Primary Kev	id_var_amb			
Kev	**ColumnName**	**Data Type**	**NotNulI**	**Descripción**
PK	id_var_amb	Serial	Yes	Identificador único de registro de variables ambientales y fuga de refrigerante.
	Fecha	Date	Yes	Representa la fecha actual del sistema.
	Hora	Time	Yes	Representa la hora actual del sistema.
	Temp	Float	Yes	Representa el valor registrado del sensor de temperatura.
	Hume	Float	Yes	Representa el valor registrado del sensor de humedad.
	Fagua		Yes	Representa el valor registrado del sensor de fuga de refrigerante.

Nota. Descripción de la tabla tb_arduino de la base de datos.
Elaborado por: David Oña y Alexis Vaca

La tabla ups_dct contiene un conjunto datos del UPS enviados al sistema MTI. Como se muestra en la tabla 17.

Tabla 17.
Diccionario de datos tabla tb_ups_dct

Nombre	tb_ups_dct			
Descripción	Almacena registros de monitoreo enviados por el UPS del Data Center			
Primary Kev	id_ups_dct			
Kev	**ColumnName**	**Data Type**	**NotNullI**	**Descripción**
PK	id_ups_dct	Serial	Yes	Identificador único de registro de variables de monitoreo del UPS.
	Fecha	Date	Yes	Representa la fecha envida por el UPS.
	Hora	Time	Yes	Representa la hora envida por el UPS.
	ent_volt_f1	Float	Yes	Representa el valor del voltaje de entrada de la fase 1 del UPS
	ent_volt_f2	Float	Yes	Representa el valor del voltaje de entrada de la fase 2 del UPS
	ent_volt_f3	Float	Yes	Representa el valor del voltaje de entrada de la fase 3 del UPS
	sal_volt_f1	Float	Yes	Representa el valor del voltaje de salida de la fase 1 del UPS
	sal_volt_f2	Float	Yes	Representa el valor del voltaje de salida de la fase 2 del UPS
	sal_volt_f3	Float	Yes	Representa el valor del voltaje de salida de la fase 3 del UPS
	volt_batt	Float	Yes	Representa el valor de voltaje del banco de baterías del UPS.

Nota. Descripción de la tabla tb_ups_dct de la base de datos.
Elaborado por: David Oña y Alexis Vaca

La tabla tb_valores_sensores contiene un conjunto de datos en el cual se define los parámetros de monitoreo de los diferentes sensores del sistema MTI. Como se muestra en la tabla 18.

Tabla 18.
Diccionario de datos tabla valores_sensores

Nombre	tb_valores_sensores			
Descripción	Almacena los parámetros para monitorear los sensores ambientales y fuga de refrigerante del MTI.			
Primary Kev	id_umbral			
Kev	**ColumnName**	**Data Type**	**NotNullI**	**Descripción**
PK	id_umbral	Serial	Yes	Identificador único de registro de parámetros de monitoreo del MTI.
	temp	Float	Yes	Representa el valor umbral de monitoreo del sensor de temperatura.
	Hume	Float	Yes	Representa el valor umbral de monitoreo del sensor de humedad.
	ent_volt_f1	Float	Yes	Representa el valor umbral de monitoreo del voltaje de entrada de la fase 1 del UPS
	ent_volt_f2	Float	Yes	Representa el valor umbral de monitoreo del voltaje de entrada de la fase 2 del UPS
	ent_volt_f3	Float	Yes	Representa el valor umbral de monitoreo del voltaje de entrada de la fase 3 del UPS
	sal_volt_f1	Float	Yes	Representa el valor umbral de monitoreo del voltaje de salida de la fase 1 del UPS
	sal_volt_f2	Float	Yes	Representa el valor umbral de monitoreo del voltaje de salida de la fase 2 del UPS
	sal_volt_f3	Float	Yes	Representa el valor umbral de monitoreo del voltaje de salida de la fase 3 del UPS
	volt_batt	Float	Yes	Representa el valor umbral de monitoreo del voltaje del banco de baterías del UPS.
	cap_batt	Float	Yes	Representa el valor umbral de monitoreo de carga del banco de baterías del UPS.

Nota. Descripción de la tabla tb_valores_sensores de la base de datos.

Elaborado por: David Oña y Alexis Vaca

La tabla tb_eventos contiene un conjunto datos en el cual se registra los incidentes ocurridos durante el monitoreo de las variables ambientales y UPS del Data Center. Como se muestra en la tabla 19.

Tabla 19.
Diccionario de datos tabla eventos

Nombre	tb_eventos			
Descripción	Almacena los diferentes id de los incidentes ocurridos durante el monitoreo de las variables ambientales y UPS del Data Center			
Primary Kev	Usuario			
Kev	**ColumnName**	**Data Type**	**NotNullI**	**Descripción**
PK	id_evento	Serial	Yes	Identificador único del evento registrado.
FK	id_var_amb	Integer	Yes	Identificador del id del incidente registrado en el monitoreo de las Variables Ambientales.
FK	id_ups_dct	Integer	Yes	Identificador del id del incidente registrado en el monitoreo del UPS del Data Center

Nota. Descripción de la tabla tb_eventos de la base de datos.

Elaborado por: David Oña y Alexis Vaca

3.8.6. Diseño de la Arquitectura

Finalizado el diseño de la base de datos, se continuará con la arquitectura del sistema como se muestra en la figura 40, para lo cual se hará uso de los casos de uso debido a que son más fáciles de elaborar e interpretar.

Arquitectura servidor web

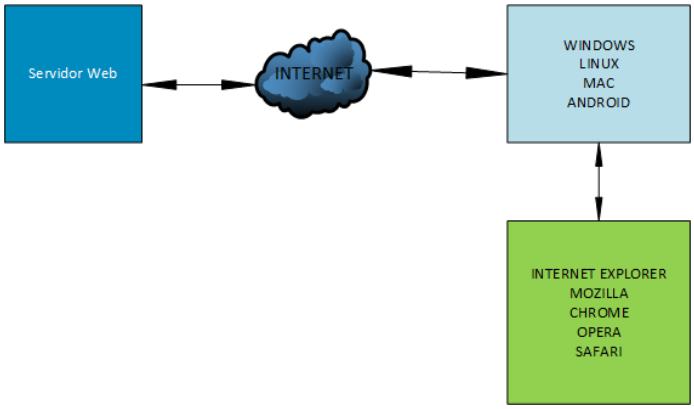


Figura 40.
Elaborado por: David Oña y Alexis Vaca

3.8.6.1. Casos de uso

Es una herramienta que sirve para representar la forma como los usuarios de un sistema interactúan con el sistema. La aplicación principal de los casos de uso es en el proceso de análisis y diseño, el caso de uso tiene 3 reglas básicas:

- Cada caso de uso tiene un actor.
- Cada caso de uso es un iniciador, es decir, un actor.
- Cada caso de uso lleva a un resultado final.

3.8.6.2. Actores del sistema

- Usuario: actor con restricciones sobre las funcionalidades del sistema.
- Administrador: actor sin restricciones sobre las funcionalidades del sistema.

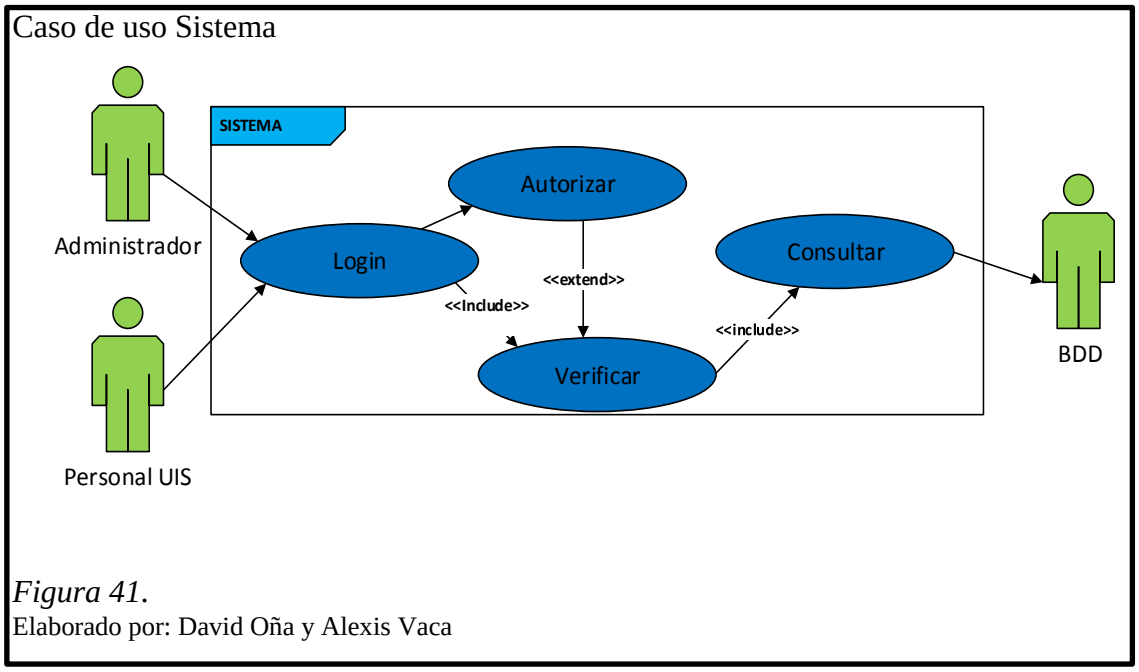


Figura 41.
Elaborado por: David Oña y Alexis Vaca

Caso de uso usuario

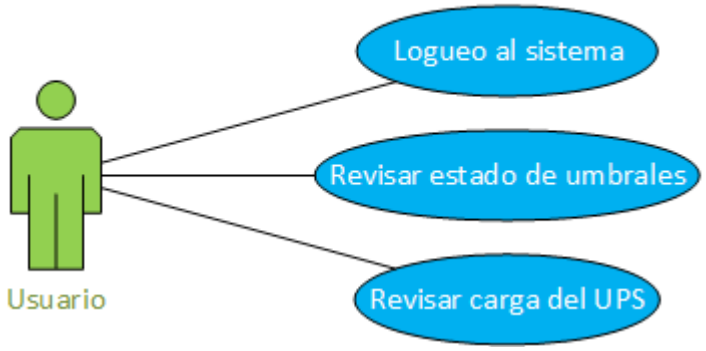


Figura 42.
Elaborado por: David Oña y Alexis Vaca

Caso de uso administrador

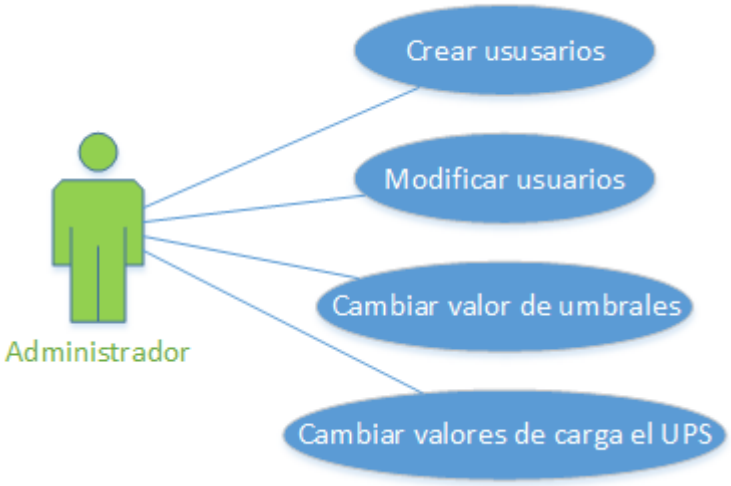


Figura 43.
Elaborado por: David Oña y Alexis Vaca

3.8.7. Interfaces de la aplicación web

3.8.7.1. Librería panamahitek_arduino

La librería panamahitek_arduino permite el envío y recepción de datos entre Arduino y Java.

La tabla 20 muestra los diferentes métodos y parámetros que contiene la librería

Tabla 20.
Librería PanamaHitek_Arduino

MÉTODO	DESCRIPCIÓN
ArduinoRX(stríng nombre del puerto, int time out, int baud rate, SerialPortEventListener evento)	Este método se utiliza para iniciar la conexión de Java con Arduino solamente para la recepción de datos. En el nombre de puerto se coloca el puerto COM donde esté conectado Arduino, el time out es el tiempo de espera (por defecto 2000), el baudrate que usa Arduino IDE (por defecto 9600) y el serial SerialPortEventListener debe ser una variable declarada antes de utilizar este método.
ArduinoTX(string nombre del puerto, int time out, int baud rate)	Este método se utiliza para iniciar la conexión de Java con Arduino solamente para la transmisión de datos.
ArduinoRXTX(string nombre del puerto, int time out, int baud rate, SerialPortEventListener evento)	Este método se utiliza para iniciar la conexión de Java con Arduino para la transmisión y recepción de datos.
sendData(String data)	Método utilizado para enviar datos a Arduino. Los datos se deben enviar como cadena de texto (string).
receiveData()	Devuelve un dato recibido a través del puerto serie en formato ASCII por lo que se debe traducir de decimal a caracter.
MessageAvailable()	Devuelve un valor boolean que nos indica si hay algún mensaje disponible para imprimir. El cual es enviado desde Arduino utilizando Serial.println().

Nota. Descripción de la librería PanamaHitek_Arduino
Elaborado por: David Oña y Alexis Vaca

3.8.7.2. Librería Java mail

La librería Java Mail desarrollada por SUN permite el envío de correos electrónicos.

La tabla 21 detalla sus principales clases y métodos.

Tabla 21.
Librería Java Mail

Clases	
Properties	La clase es la encargada de almacenar las propiedades de la conexión que vamos a establecer con el servidor de correo Saliente SMTP.
Session	Es la clase encargada de manejar la sesión de usuario
MimeMessage	Forma el mensaje que se enviara
Transport	Define los parámetros del protocolo de transporte. Inicializa la variable obteniendo el tipo de protocolo de transporte de la clase session.
Metodos	
Put	Mediante éste método asignaremos las propiedades que necesitamos, como son: o Servidor SMTP. o Puerto SMTP. o E-mail emisor del mensaje. o Usuario de acceso al servidor SMTP. O Valor booleano de contraseña requerida.
Get	Obtención de los parámetros anteriores ya guardados.
GetDefaultInstance	Introduce la variable de la clase Properties y se creará una sesión para dichas propiedades.
GetTransport	El método indica el protocolo de transporte a utilizar (por defecto smtp).
SetFrom	Recibe como parámetro la dirección del emisor del mensaje de tipo InternetAddress.
AddRecipient	Recibe 2 parámetros. Message.RecipientType (TO,CC,BCC). Como segundo parámetro una variable de la clase InternetAddress con la dirección del receptor.
SetSubject	Introduce el asunto del mensaje como único parámetro en forma de String.
SetText	Introduce el texto del mensaje en forma de String.
Connect	Se encarga de establecer la conexión con el servidor, introduciendo el nombre de usuario y contraseña (si es requerida).
SendMessage	Envía el mensaje que hemos creado anteriormente a los destinatarios especificados.
Close	Cierra la conexión.

Nota. Descripción de la librería JavaMail.
Elaborado por: David Oña y Alexis Vaca

3.8.8. Interfaz de la pantalla principal

La figura 44 muestra el diseño de la interfaz de la pantalla principal, dónde el usuario puede navegar por las diferentes secciones del portal.

Diseño de la interfaz de la pantalla principal

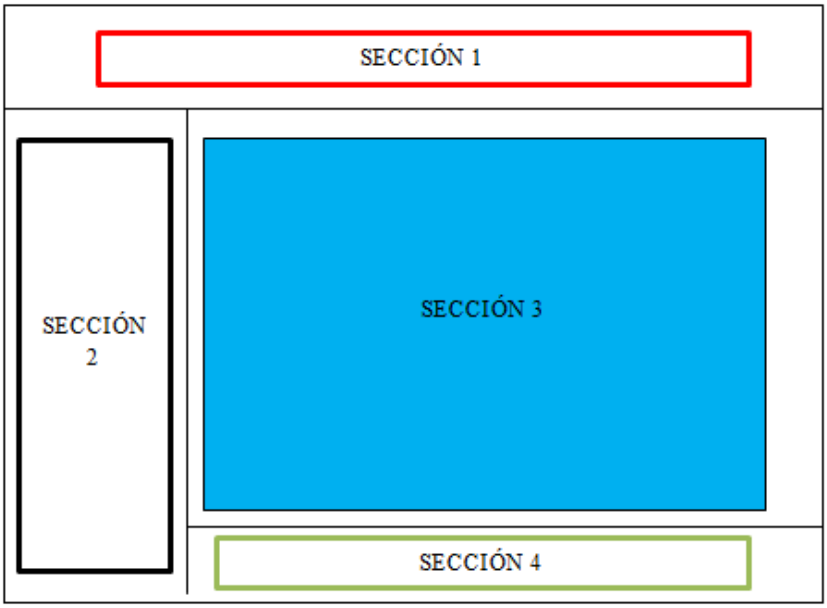


Figura 44.
Elaborado por: David Oña y Alexis Vaca

Descripción:

- Sección 1.

  Esta sección generará y visualizará por cada pantalla el logo y nombre de la aplicación web. También permite acceder a la iteración del sistema a través del evento de login.

- Sección 2.

  Esta sección generará el menú con sus diferentes configuraciones para la navegación de la aplicación.

- Sección 3.

  Esta sección generará la visualización de los estados de los umbrales (humedad, temperatura, monitor del UPS Y sensor de agua).

- Sección 4.

  Esta sección generará la visualización de los estados de cargas del UPS.

En la figura 45 se muestra el diseño final de la página principal donde el usuario visualizara y navegara por el portal.

Página Principal

z

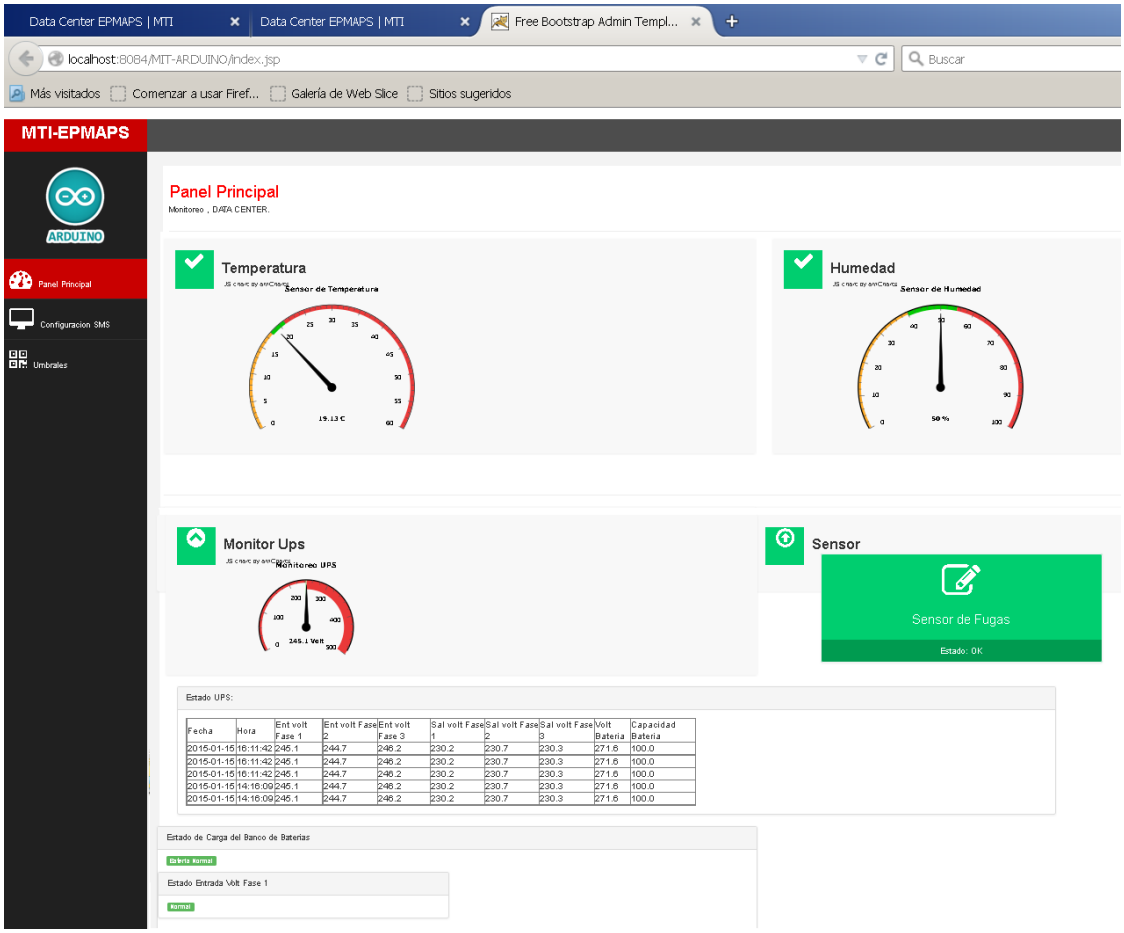


Figura 45.
Elaborado por: David Oña y Alexis Vaca

El Data Center de la EPMAPS tiene como valores de las variables ambientales los siguientes parámetros:

- 24 horas de refrigeración continua.
- Temperatura del Data Center entre 18° C a 20° C
- Humedad relativa Data Center del 50%

Las escalas entre valores normales y críticos se tomaron en base a la recopilación de datos explicados en el Capítulo 1.

Los rangos de valores de las bandas de los medidores análogos de la pantalla principal se definen en la tabla 22

Tabla 22.

Rangos de valores

Nro.	Nombre variables	Color Banda	Intervalos Valores(T/H)	Estado
Banda 1	band 1	Naranja	0° - 17° C/ 0% - 39 %	Alerta
Banda 2	band 2	Verde	18° - 20° C/40% - 55 %	Normal
Banda 3	band 3	Roja	> 20° C/ >56%	Peligro

Nota. T/H= temperatura / humedad

Elaborado por: David Oña y Alexis Vaca

3.8.9. Sensor de temperatura

Medidor análogo de temperatura

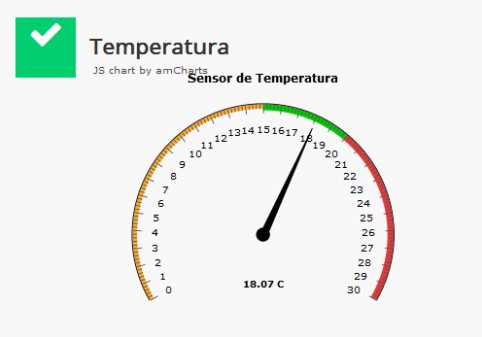


Figura 46.

Elaborado por: David Oña y Alexis Vaca

Como se observa en la figura 46 los valores de temperatura en verde son los permitidos en un Data Center, Si el valor del umbral de temperatura definido en la variable double_temp es mayor al determinado en la tabla tb_umbrales de la base de datos, es

decir pasa los valores a rojo significa que se ha producido un aumento en la temperatura en el data center, por lo que se realiza una consulta a la base de datos se verifica los números de celular de los usuarios y a través del método sendData(String data) se envía los datos a Arduino, el mensaje de “FALLO TEMP” más el valor actual de temperatura registrado por el sensor LM35 y el número del destinatario para ser enviado por SMS. En la figura 47 Se observa el código que permite realizar esta acción.

Código Java medidor análogo

```
////////
if (double_temp >= tempe_umb) { // no mayor porque sigue enviando mensajes hasta que se

    String sql2 = "Select usuario, num_cel from tb_num";
    ResultSet rs2 = null;

    try {

        rs2 = conexion.Consulta(sql2);
        while (rs2.next()) {

            if (cont <= cont_num_cel) {
                // cadena[cont]=rs2.getString(2);
                System.out.println(rs2.getString(2));
                Arduino.SendData(rs2.getString(2) + ".FALLA TEMP:" + temp);

                Thread.sleep(4200);

                // System.out.println("CONTADOR ENVIANDO NUMERO"+ cont );
                cont++;
            }

        }
        // System.out.println("FINALIZADO");
        Arduino.SendData("BAJANDO:");
        cont = 0;
    } catch (Exception ex) {
        Logger.getLogger(Cls_principal.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Figura 47.
Elaborado por: David Oña y Alexis Vaca

3.8.10. Sensor humedad

Medidor análogo de humedad

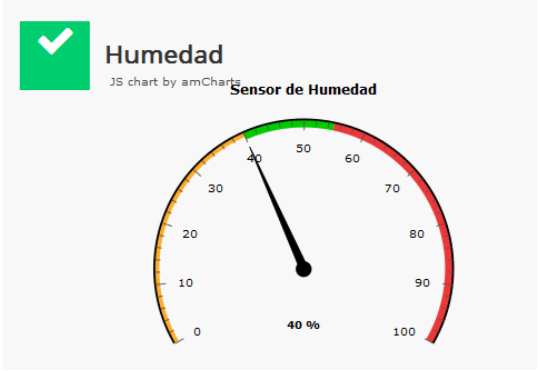


Figura 48.
Elaborado por: David Oña y Alexis Vaca

Como se observa en la figura 48 los valores de humedad en verde son los permitidos en un Data Center, si el valor del umbral de humedad relativa definido en la variable double_humedad es mayor al determinado en la tabla tb_umbrales de la base de datos es decir pasa los valores a rojo significa que se ha producido un aumento en la humedad relativa del Data Center, por lo que se realiza una consulta a la base de datos se verifica los números de celular de los usuarios y a través del método sendData(String data) se envía los datos a Arduino, el mensaje de "FALLO HUMEDAD" más el valor actual de temperatura registrado por el sensor digital DHT11 y el número del destinatario para ser enviado por SMS.

En la figura 49 se observa el código que permite realizar esta acción.

Código Java medidor análogo de humedad

```
if (double_Humedad > hume_umb) { //

    String sql2 = "Select usuario, num_cel from tb_num";
    ResultSet rs2 = null;

    try {

        rs2 = conexion.Consulta(sql2);
        while (rs2.next()) {

            if (cont <= cont_num_cel) {
                // cadena[cont]=rs2.getString(2);
                System.out.println(rs2.getString(2));
                Arduino.SendData(rs2.getString(2) + ".FALLO HUMEDAD:" + hume_umb);
                Thread.sleep(4200);
      //   Thread.sleep(14*60*10);

                // System.out.println("CONTADOR ENVIANDO NUMERO"+ cont );
                cont++;
            }

        }
        //   System.out.println("FINALIZADO");
        Arduino.SendData("BAJANDO:");
        cont = 0;
    } catch (Exception ex) {
        Logger.getLogger(Cls_principal.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Figura 49.
Elaborado por: David Oña y Alexis Vaca

3.8.11. Sensor de fugas de agua

Gráfico de estado del sensor de fugas de agua

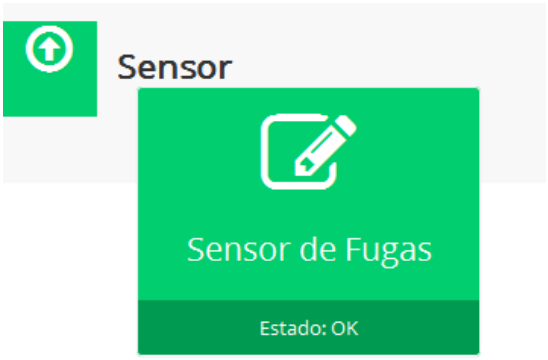


Figura 50.
Elaborado por: David Oña y Alexis Vaca

El principio básico para detectar las fugas del refrigerante es capturar la luz emitida por un led el cual es reflejada en una fotocelda que está conectada con resistencia de 10k en el pin 0 análogo de Arduino, el cual lee valores entre 0 y 1023 que son interpretados como disminución o aumento de presencia de luz.

Como se observa en la figura 50 el color verde indica que no hay fugas de agua en el Data Center, Si el umbral definido en la tabla tb_umbrales de la base de datos es mayor a 50 significa que se ha detectado una fuga de refrigerante, es decir cambia a color rojo el icono de la aplicación, por lo que se realiza una consulta a la base de datos se verifica los números de celular de los usuarios y a través del método Thread.sleep (tiempo de espera) que hace que el hilo que se está ejecutando se detenga por 4.2 milisegundos, tiempo necesario para enviar a través del método sendData(String data) utilizado para enviar datos a Arduino, el mensaje de fuga de refrigerante detectada y el número del destinatario para ser enviado por SMS.

En la figura 51 se observa el código que permite realizar esta acción.

Parámetros sensor de agua

```
//////// UMBRAL FUGA DE AGUA
if (double_Fagua > 50) { // colapsa el sistema si se envian silultaneamente 2 datos

    String sql2 = "Select usuario, num_cel from tb_num";
    ResultSet rs2 = null;

    try {

        rs2 = conexion.Consulta(sql2);
        while (rs2.next()) {

            if (cont <= cont_num_cel) {
                // cadena[cont]=rs2.getString(2);
                System.out.println(rs2.getString(2));
                Arduino.SendData(rs2.getString(2) + ".FUGA REFRIGERANTE DETECTADA");

                Thread.sleep(4200);

                //System.out.println("CONTADOR ENVIANDO NUMERO"+ cont );
                cont++;
            }

        }
```

Figura 51.

Elaborado por: David Oña y Alexis Vaca

3.8.12. Monitor UPS

Medidor análogo del UPS

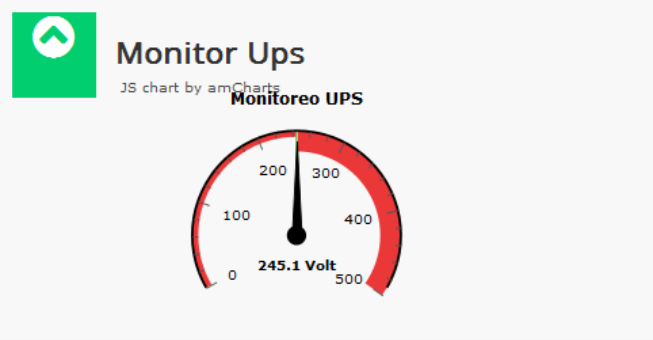


Figura 52.
Elaborado por: David Oña y Alexis Vaca

La clase Scanner de Java provee métodos para leer valores de entrada de varios tipos y está localizada en el paquete Java.util. Los valores de entrada pueden venir de varias fuentes, incluyendo valores que entren por el teclado o datos almacenados en un archivo.

Cuando se introducen caracteres por teclado o se lee un archivo almacenado en el disco duro, el objeto Scanner toma en su totalidad la cadena introducida y la divide en elementos llamados tokens.
El carácter predeterminado que sirve de separador de tokens es el espacio en blanco.

Por ejemplo, si introducimos por consola (Lectura de Datos con arduino), Scanner divide la cadena en los siguientes tokens como se muestra en la tabla 23.

Tabla 23.
Tokens método Scanner

Palabra	Tokens
Lectura de Datos con Arduino	Lectura
	de
	Datos
	con
	Arduino

Nota. División de palabras con los tokens.
Elaborado por: David Oña y Alexis Vaca

Los métodos que proporciona la clase Scanner se puede acceder a los tokens y trabajar con ellos en el programa. Para utilizar esa clase tenemos que crear primero un objeto de ella para poder invocar sus métodos. La siguiente declaración crea un objeto de la clase Scanner que lee valores de entrada del teclado.

Scanner teclado = new Scanner(System.in);

El propósito de pasar a System.in como argumento es conectar o establecer una relación entre el objeto tipo Scanner, con nombre teclado en la declaración anterior, y el objeto System.in, que representa el sistema estándar de entrada de información en Java. Si no se indica lo contrario, el teclado es, por omisión, el sistema estándar de entrada de información en Java.

Luego que se tenga un objeto de la clase Scanner asociado al sistema estándar de entrada System.in, se llama, por ejemplo, su método nextlnt para entrar un valor del tipo int. Para entrar otros valores de otros tipos de datos primitivos, se usan los métodos correspondientes como nextByte o nextDouble. En la tabla 24 se muestra la descripción de los métodos que tienen los tokens.

Tabla 24.

Método Scanner

Metodo	Descripción
nextXxx()	Devuelve el siguiente token como un tipo básico. Xxx es el tipo. Por ejemplo, nextInt() para leer un entero, nextDouble para leer un double, etc.
next()	Devuelve el siguiente token como un String.
nextLine()	Devuelve la línea entera como un String. Elimina el final \n del buffer
hasNext()	Devuelve un boolean. Indica si existe o no un siguiente token para leer.
hasNextXxx()	Devuelve un boolean. Indica si existe o no un siguiente token del tipo especificado en Xxx, por ejemplo hasNextDouble()
useDelimiter(String)	Establece un nuevo delimitador de token.

Nota. Descripción del método Scanner

Elaborado por: David Oña y Alexis Vaca

Los logs que genera automáticamente el UPS en un intervalo de 5 minutos se guarda en un archivo con extensión .csv como se muestra en la Figura 53 el cual es un tipo de documento en formato abierto que permite representar datos en forma de tabla, en las que las columnas se separan por comas (o punto y coma en donde la coma es el separador) y las filas por saltos de línea.

Logs UPS

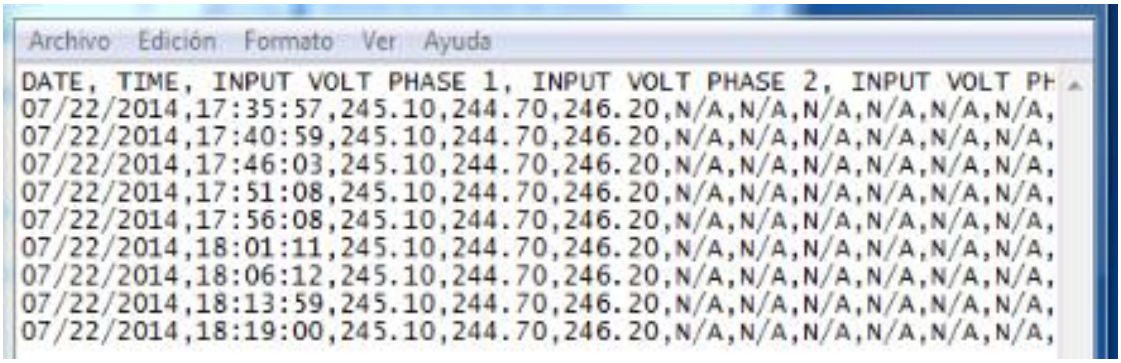


Figura 53.

Elaborado por: David Oña y Alexis Vaca

En la figura 54 se muestra el código de la clase lectura_ups() que contiene la ruta del archivo .csv antes mencionado.

Método lectura_ups

```
    public String lectura_ups() {

        int aux[] = new int[2];
        final String nomFich = "c:/lectura/UpsData.csv";

        Scanner in = null;

        try {

// abre el fichero
            in = new Scanner(new FileReader(nomFich));

// configura el formato de números
            in.useLocale(Locale.ENGLISH);

// lee el fichero palabra a palabra
            int pos = 0;

            String cadena[] = new String[50000]; // Asigna tamaño al vector

            while (in.hasNext()) {

// lee primera palabra
                String palabra = in.next();

                cadena[pos] = palabra;

 //if(palabra.contains("20140705    140151") ){
//System.out.println("Posicon:"+pos);
//}
                pos++;
```

Figura 54.

Elaborado por: David Oña y Alexis Vaca

TextScanner crea un objeto de escáner en el archivo. El escáner rompe el contenido del archivo en tokens utilizando un patrón de delimitación, Por defecto, el patrón de delimitación es un espacio en blanco. TextScanner continuación, llama al método hasNext () en el escáner. Este método devuelve (verdadero) si existe otro token en la entrada del analizador, que es el caso hasta que llega al final del archivo. El método next () devuelve una cadena que representa el siguiente token. Así que hasta que llegue al final del archivo, TextScanner imprime la cadena devuelta por next() en una línea separada como se muestra en la figura 55.

Log UPS archivo plano

Figura 55:

Elaborado por: David Oña y Alexis Vaca

En la figura 56 se muestra como el UPS NewWave guarda los logs en un archivo plano y se selecciona las variables más importantes para su respectivo monitoreo, a continuación se guarda los caracteres de las posiciones requeridas en vectores.

CharArray

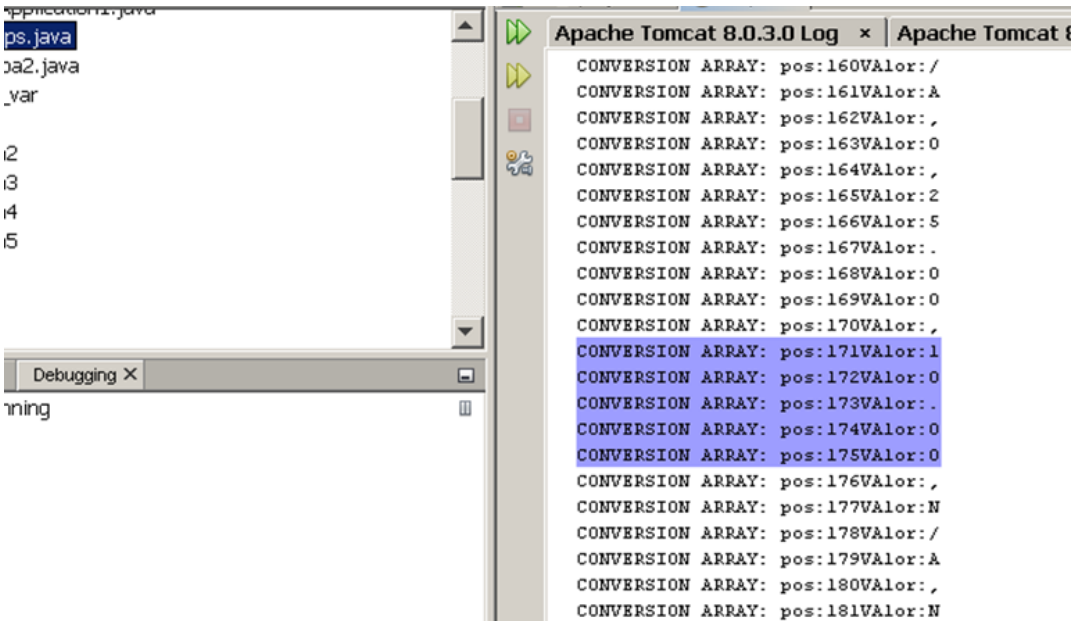


Figura 56.

Elaborado por: David Oña y Alexis Vaca

También se usa el método String.valueOf como se muestra en la figura 57 el cual convierte a cadenas los tipos de datos pasados como parámetro. En este caso se utiliza para convertir un arreglo de caracteres en una cadena. Cada log tiene un tamaño mayor a 203 caracteres, si se produjera una variación del tamaño se enviara un mensaje al celular y correo electrónico, una notificación en la cual se indica que se perdió la sincronización con el UPS.

En las variables de tipo String como "fecha_ups" se asigna las diferentes posiciones de caracteres definidas en cada registro del log que guarda automáticamente el UPS.

Método String.valueOf

```
(charArray.length >= 203) { // si es menor de 200 es porque se perdio la conexion con el ups , ya que no coinciden las posciones de .

 fecha_ups = String.valueOf(charArray[3] + "" + charArray[4] + "" + charArray[5] + "" + charArray[0] + "" + charArray[1] + "" + char.
 hora_ups = String.valueOf(charArray[11] + "" + charArray[12] + "" + charArray[13] + "" + charArray[14] + "" + charArray[15] + "" + (
 ent_volt_f1 = String.valueOf(charArray[20] + "" + charArray[21] + "" + charArray[22] + "" + charArray[23] + "" + charArray[24] + ""
 ent_volt_f2 = String.valueOf(charArray[27] + "" + charArray[28] + "" + charArray[29] + "" + charArray[30] + "" + charArray[31] + ""
 ent_volt_f3 = String.valueOf(charArray[34] + "" + charArray[35] + "" + charArray[36] + "" + charArray[37] + "" + charArray[38] + ""
 sal_volt_f1 = String.valueOf(charArray[83] + "" + charArray[84] + "" + charArray[85] + "" + charArray[86] + "" + charArray[87] + ""
 sal_volt_f2 = String.valueOf(charArray[90] + "" + charArray[91] + "" + charArray[92] + "" + charArray[93] + "" + charArray[94] + ""
 sal_volt_f3 = String.valueOf(charArray[97] + "" + charArray[98] + "" + charArray[99] + "" + charArray[100] + "" + charArray[101] + '
 volt_batt = String.valueOf(charArray[152] + "" + charArray[153] + "" + charArray[154] + "" + charArray[155] + "" + charArray[156] +
 cap_batt = String.valueOf(charArray[171] + "" + charArray[172] + "" + charArray[173] + "" + charArray[174] + "" + charArray[175]);/
```

Figura 57.

Elaborado por: David Oña y Alexis Vaca

Si se produce variación en los valores asignados en los vectores se guarda automáticamente el registro en la tabla tb_ups_dtc como se muestra en la figura 58

Consulta a la tabla tb_ups_dtc

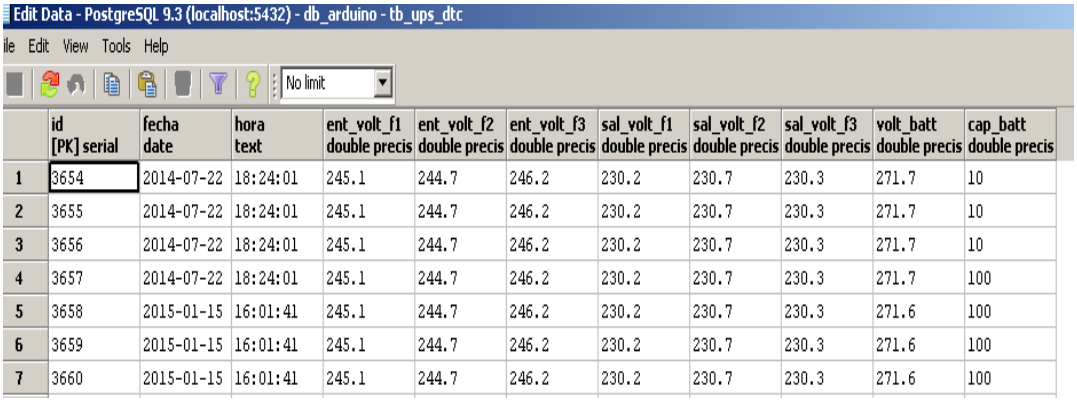
```
if (dble_ent_volt_f1 < 245 || dble_ent_volt_f2 < 244 || dble_ent_volt_f3 < 246 || dble_sal_volt_f1 < 230 || dble_sal_volt_f2 < 230
        || dble_sal_volt_f3 < 230 || dble_volt_batt < 271.70 || dble_cap_batt < 100.00) {

    String sql1 = "INSERT INTO tb_ups_dtc (fecha,hora,ent_volt_f1,ent_volt_f2,ent_volt_f3,sal_volt_f1,sal_volt_f2,sal_volt_f3,volt_batt,cap_batt)
            + "VALUES ( '" + fecha_ups + "', '" + hora_ups + "', '" + dble_ent_volt_f1 + "','" + dble_ent_volt_f2 + "','" + dble_ent_volt_f3
            + "','" + dble_sal_volt_f1 + "','" + dble_sal_volt_f2 + "','" + dble_sal_volt_f3 + "','" + dble_volt_batt + "','" + dble_cap_batt + "'
    ResultSet rs = null;
    Cls_Conexion conexion = new Cls_Conexion();
    rs = conexion.Consulta(sql1);
```

Figura 58.

Elaborado por: David Oña y Alexis Vaca

La figura 59 muestra como PostgreSQL guarda los datos en la tabla tb_ups_dtc

Tabla tb_ups_dtc



	id [PK] serial	fecha date	hora text	ent_volt_f1 double precis	ent_volt_f2 double precis	ent_volt_f3 double precis	sal_volt_f1 double precis	sal_volt_f2 double precis	sal_volt_f3 double precis	volt_batt double precis	cap_batt double precis
1	3654	2014-07-22	18:24:01	245.1	244.7	246.2	230.2	230.7	230.3	271.7	10
2	3655	2014-07-22	18:24:01	245.1	244.7	246.2	230.2	230.7	230.3	271.7	10
3	3656	2014-07-22	18:24:01	245.1	244.7	246.2	230.2	230.7	230.3	271.7	10
4	3657	2014-07-22	18:24:01	245.1	244.7	246.2	230.2	230.7	230.3	271.7	100
5	3658	2015-01-15	16:01:41	245.1	244.7	246.2	230.2	230.7	230.3	271.6	100
6	3659	2015-01-15	16:01:41	245.1	244.7	246.2	230.2	230.7	230.3	271.6	100
7	3660	2015-01-15	16:01:41	245.1	244.7	246.2	230.2	230.7	230.3	271.6	100

Figura 59.

Elaborado por: David Oña y Alexis Vaca

3.8.13. Interfaz pantalla registro de usuarios.

En la figura 60 se muestra el diseño de la interfaz de la página de ingreso de un nuevo usuario donde se hace la revisión del mecanismo de autenticación y el usuario navegará por las diferentes secciones de la página.

Diseño de interfaz página de ingreso de usuarios

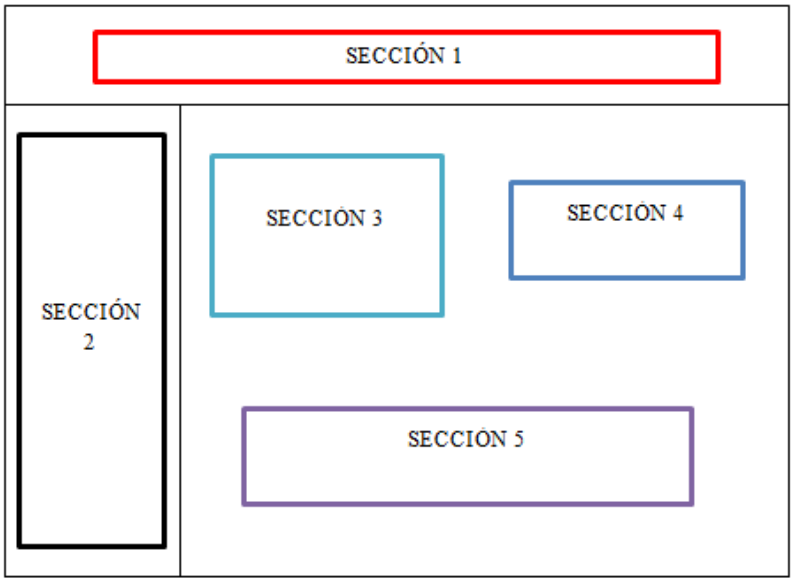


Figura 60.

Elaborado por: David Oña y Alexis Vaca

Descripción:

- Sección 1 y sección 2.

  Estas secciones cumplen las mismas funciones del diseño de interface del formulario.

- Sección 3.

  Esta sección permitirá realizar el ingreso de un nuevo usuario (nombre de usuario, email y número celular).

- Sección 4.

  Esta sección generará y visualizará los usuarios ya ingresados que constan en la base de datos.

- Sección 5.

  Esta sección generará la eliminación de un usuario mediante el id asignado automáticamente.

En la figura 61 se muestra el diseño final de la página de ingreso de un nuevo usuario, donde se podrá visualizar y navegar por el portal.

Página de ingreso de usuarios

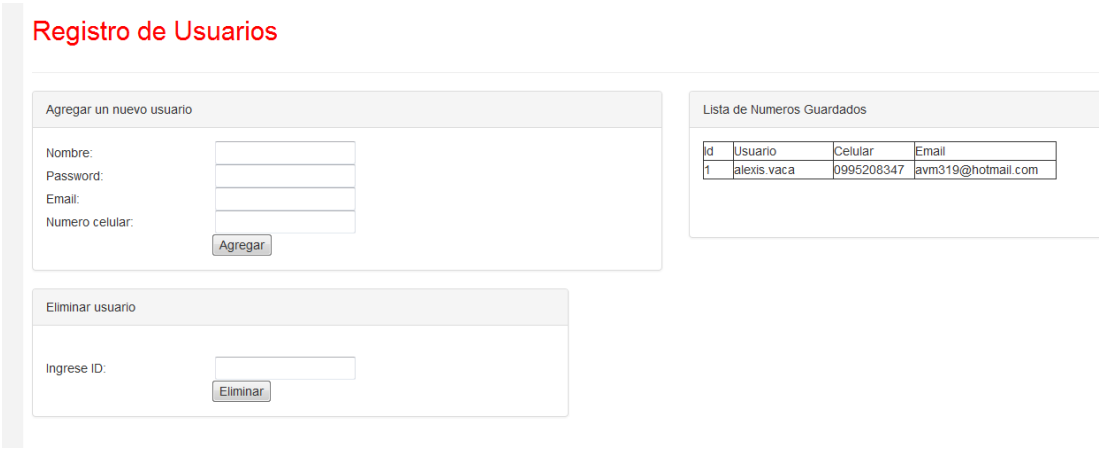


Figura 61.
Elaborado por: David Oña y Alexis Vaca

En la figura 62 se muestra el código del registro de usuarios

Código Java ingreso de usuarios

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>

        <%



            String txt_nombre = request.getParameter("txt_nombre");
               String txt_email = request.getParameter("txt_email");
           String txt_num_cel = request.getParameter("txt_num_cel");


            if(txt_nombre.length()>3 & txt_num_cel.length()==12){
Cls_consulta pro1=new Cls_consulta();
String resultado1=pro1.Guardar_Num(txt_nombre, txt_num_cel, txt_email);
out.print(resultado1);
            }


        %>


        <% response.sendRedirect("conf_sms.jsp"); %>
```

Figura 62.
Elaborado por: David Oña y Alexis Vaca

3.8.14. Diseño de interfaz pantalla configuración umbrales

En la figura 63 se muestra el diseño la interfaz de la página de configuración de los umbrales y cargas del UPS, el usuario navegará por las diferentes secciones de la página.

Diseño de interfaz página de configuración de umbrales

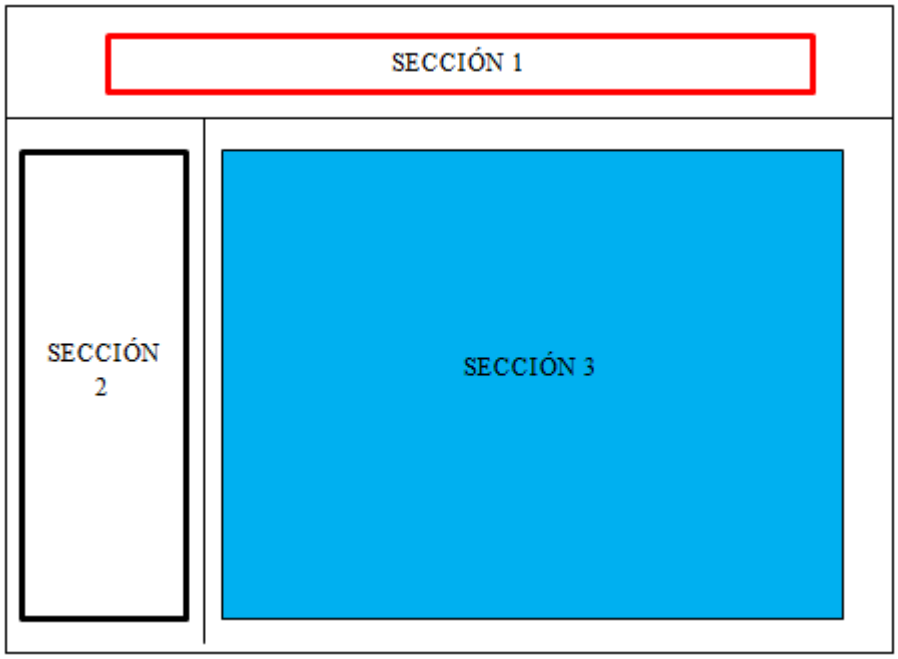


Figura 63.
Elaborado por: David Oña y Alexis Vaca

Descripción:

- Sección 1 y sección 2.

  Estas secciones cumplen las mismas funciones del diseño de interfaz del formulario.

- Sección 3.

  Esta sección generará la configuración de los umbrales y estados de carga del UPS (valor actual y modificar valor).

En la figura 64 se muestra el diseño final de la página de configuración de umbrales y estados de cargas del UPS donde el usuario podrá visualizar y navegar por el portal.

Página de configuración de umbrales

Figura 64.
Elaborado por: David Oña y Alexis Vaca

En la figura 65 se muestra el código de configuración de los umbrales y los estados de carga del UPS.

Código Java configuración de umbrales

```
        <%
        double txt_temp = Double.parseDouble(request.getParameter("txt_temp"));
        double txt_hum = Double.parseDouble(request.getParameter("txt_hum"));
        double txt_ent_volt_fase1 = Double.parseDouble(request.getParameter("txt_ent_volt_fase1"));
       double txt_ent_volt_fase2 = Double.parseDouble(request.getParameter("txt_ent_volt_fase2"));
        double txt_ent_volt_fase3 = Double.parseDouble(request.getParameter("txt_ent_volt_fase3"));

              double txt_sal_volt_fase1 = Double.parseDouble(request.getParameter("txt_sal_volt_fase1"));
              double txt_sal_volt_fase2 = Double.parseDouble(request.getParameter("txt_sal_volt_fase2"));
              double txt_sal_volt_fase3 = Double.parseDouble(request.getParameter("txt_sal_volt_fase3"));


                double txt_tension_batt = Double.parseDouble(request.getParameter("txt_tension_batt"));
        double txt_cappacidad_batt = Double.parseDouble(request.getParameter("txt_cappacidad_batt"));


Cls_consulta pro2=new Cls_consulta();
String resultado2=pro2.Guardar_Umbrales(txt_temp, txt_hum, txt_ent_volt_fase1, txt_ent_volt_fase2,
        txt_ent_volt_fase3, txt_sal_volt_fase1, txt_sal_volt_fase2, txt_sal_volt_fase3,
        txt_tension_batt, txt_cappacidad_batt);
out.print(resultado2);

        %>
```

Figura 65.
Elaborado por: David Oña y Alexis Vaca

3.8.14.1. Mapa navegacional de la aplicación web

En la figura 66 se muestra el mapa navegacional de la aplicación web.

Mapa navegacional aplicación web

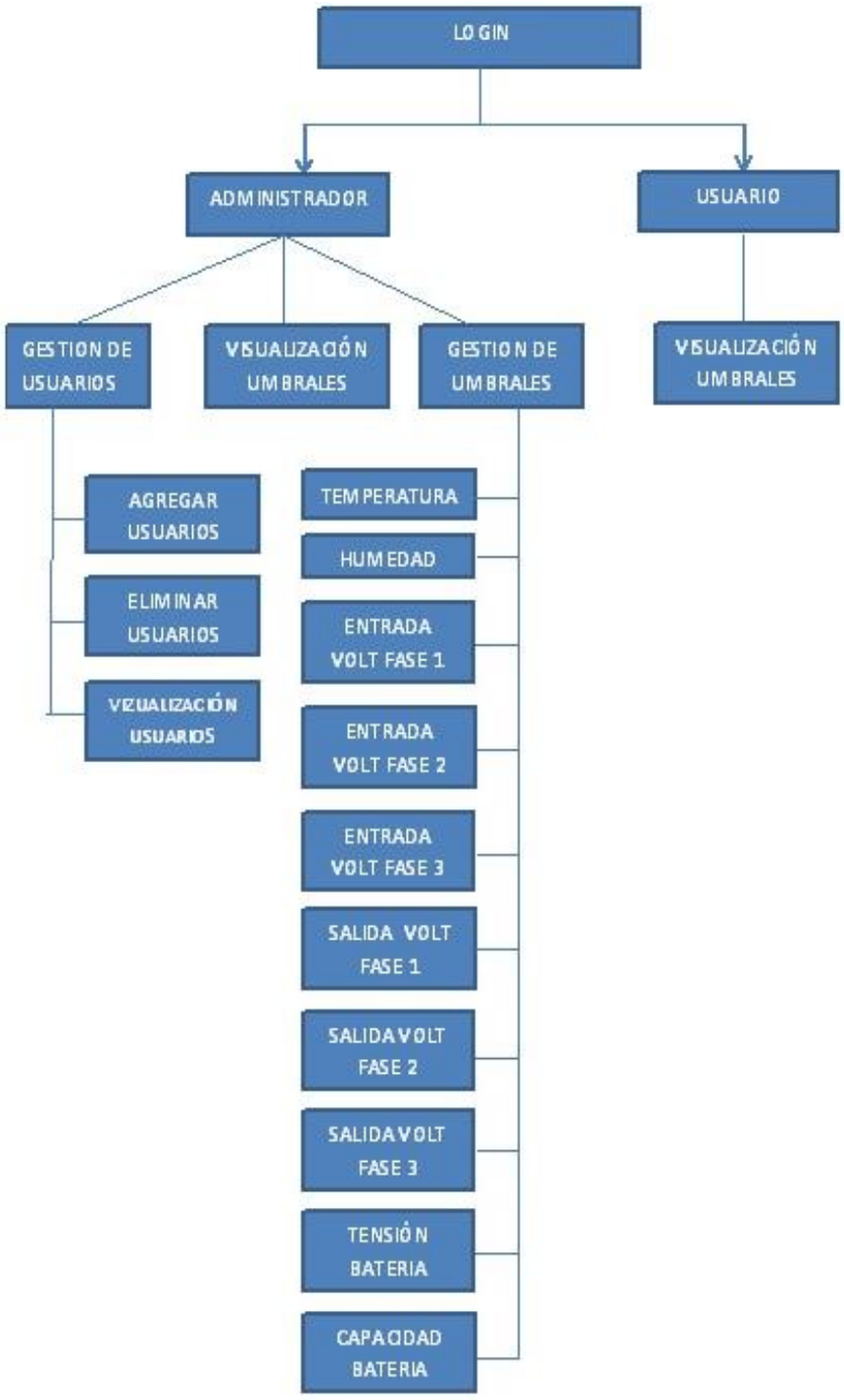


Figura 66.
Elaborado por: David Oña y Alexis Vaca

3.9. Aplicación Android

3.9.1. Archivo AndroidManifest.xml

Es un archivo .xml donde se especifica la configuración global de la aplicación Android y los componentes que la forman. En la tabla 25 se muestra la información del archivo AndroidManifest.xml

Tabla 25.
Arquitectura Android

Etiqueta	Atributo	Definición
<manifest>	Xmlns	Línea de código por defecto en una aplicación Android
	Package	Hace referencia al nombre del paquete de la aplicación
	VersionCode	Número de versión del código de la aplicación
	VersionName	Número de versión es el que se muestra en la Playstore
	MinSDKVersion	Versión mínima de SDK con la que va a funcionar la aplicación
SDK	TargetSDKVersion	Es la versión de la API de la app.
Application	AllowBackup	Permite hacer copia de seguridad de la aplicación y contenido.
	Icon	Icono de la aplicación
	Label	Nombre de la aplicación
	Theme	Estilos de nuestra aplicación
Activity	Name	Nombre de la clase Java que implementa el Activity
	Label	Nombre de la Activity que el usuario visualizará.
Intent-filter		Especifica lo qué está permitido hacer al Activity
Category		Indicamos que es la actividad principal, la que se va a lanzar cuando la aplicación se inicie

Nota. Descripción de los elementos de Android

Elaborado por: David Oña y Alexis Vaca

3.9.1.1. Interface principal.

Interfaz principal MTI Android

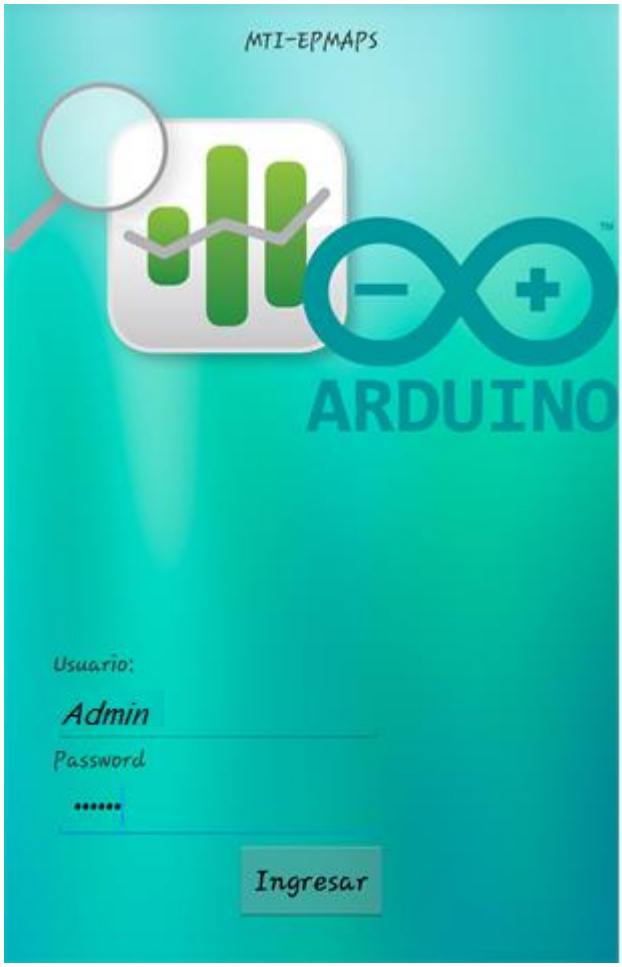


Figura 67.

Elaborado por: David Oña y Alexis Vaca

El Activity principal envía los parámetros ingresados desde el dispositivo móvil como es usuario y contraseña a un segundo Activity el cual validara si existe o no en la base de datos que encuentra en el servidor para que a partir de esta verificación se proceda a mostrar los valores actuales de los sensores de humedad temperatura y sensor de fuga de refrigerante. En la figura 68 Se muestra el código del Activity principal

Código Activity Principal

```
package com.androidya.proyecto010;

import android.app.Activity;

public class MainActivity extends Activity {
    private EditText et1;
    private EditText et2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        et1 = (EditText) findViewById(R.id.et1); //Cuadro de Texto para ingresar el Usuario
        et2 = (EditText) findViewById(R.id.et2); //Cuadro de Texto para ingresar la Contraseña
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }

    public void ejecutar(View view) {
        Intent i = new Intent(this, Actividad2.class); //parametros e1,e2 que se envian al activity2
        i.putExtra("direccion", et1.getText().toString()); // obtener texto de la variable e1= usuario
        i.putExtra("password", et2.getText().toString());// obtener texto de la variable e2= password
        startActivity(i);
    }

}
```

Figura 68.
Elaborado por: David Oña y Alexis Vaca

Método putExtra de la clase Intent. Tiene dos parámetros de tipo String, en el primero indicamos el nombre del dato y en el segundo el valor del dato.

3.9.2. Autenticación de la aplicación Android con el servidor

En la clase llamada Actividad2 se define una variable de tipo Bundle y se la inicializa llamando al método getExtras() de la clase Intent (la cual recupera el o los parámetros que envió la otra actividad (Activity). En la figura 69 se muestra el código de la actividad 2.

Código Actividad2

```
Thread sqlThread = new Thread() {
      public void run() {

   try {


      Class.forName("org.postgresql.Driver");
      // "jdbc:postgresql://IP:PUERTO/DB", "USER", "PASSWORD");
      Connection conn = DriverManager.getConnection(
                     "jdbc:postgresql://192.168.3.112:5432/db_arduino", "postgres", "P0werAndr0id$*789");
         //En el stsql se puede agregar cualquier consulta SQL deseada.
        // String stsql = "Select usuario,password from tb_login";
      Bundle bundle = getIntent().getExtras();

      String stsql = "Select usuario,password from tb_login where usuario='" + bundle.getString("direccion")
             + "' and password='" + bundle.getString("password") + "'";



        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(stsql);
       rs.next();
       System.out.println( rs.getString(1) );
       System.out.println( rs.getString(2) );
      // var_cons=rs.getString(1);
      // var_pasw=rs.getString(2);

       sqlThread2.start();


       conn.close();
catch (SQLException se) {

var_cons=("Disculpe esta tratando de ingresar a un sitio restringido. Error: " + se.toString());
```

Figura 69.

Elaborado por: David Oña y Alexis Vaca

Por medio de una consulta SQL a la base de datos se compara el usuario y la contraseña de ser validado el usuario la consulta ejecuta la tarea sqlThread2.start(); la cual se construye implementando el método run(), dentro del cual se realiza la tarea. Se llama al método start() del objeto sqlThread2 para comenzar la ejecución de la tarea en segundo plano, en la que se asigna los valores retornados por la consulta a variables definidas en la Actividad2.

En la figura 70 se muestra la conexión a la base de datos que es la misma que se utiliza para la aplicación web.

Código validación de usuario

```
Thread sqlThread2 = new Thread() {

    public void run() {

try {


  Class.forName("org.postgresql.Driver");
  // "jdbc:postgresql://IP:PUERTO/DB", "USER", "PASSWORD");
  Connection conn = DriverManager.getConnection(
                  "jdbc:postgresql://192.168.3.112:5432/db_arduino", "postgres", "P0werAndr0id$*789");
     //En el stsql se puede agregar cualquier consulta SQL deseada.
    // String stsql = "Select usuario,password from tb_login";
  Bundle bundle = getIntent().getExtras();

  String stsql = "Select fecha,hora,temp,hume,fagua from tb_arduino ORDER BY id DESC LIMIT 1 ";



      Statement st = conn.createStatement();
      ResultSet rs = st.executeQuery(stsql);
    rs.next();
    System.out.println( rs.getString(1) );
    System.out.println( rs.getString(2) );
    fecha=rs.getString(1);
    hora=rs.getString(2);
    temp=rs.getString(3);
    hume=rs.getString(4);
    fagua=rs.getString(5);


    conn.close();
} catch (SQLException se) {

var_cons=("Disculpe esta tratando de ingresar a un sitio restringido. Error: " + se.toString());
} catch (ClassNotFoundException e) {
var_cons=("oops! No se encuentra la clase. Error: " + e.getMessage());
}
```

Figura 70.

Elaborado por: David Oña y Alexis Vaca

3.9.3. Desarrollo list view

Una vista ListView visualiza una lista deslizable verticalmente de varios elementos como se muestra en la figura 71 donde cada elemento puede definirse como un Layout.

ListView

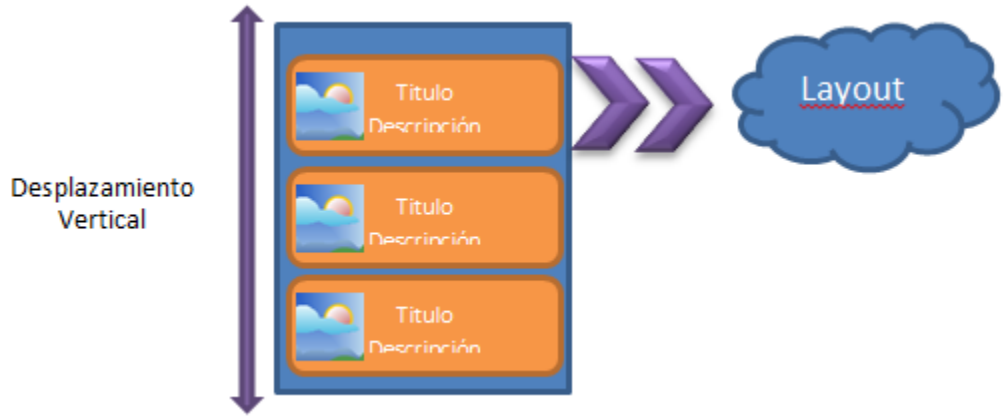


Figura 71.

Elaborado por: David Oña y Alexis Vaca

Se tiene dos Layouts:

- "entrada.xml". Es la unidad en la que se copia los datos en el interior del listado
- "listado.xml" Layout vacío el cual espera para contener la copia del diseño de la entrada.xml; y cada entrada con datos diferentes.

Layout

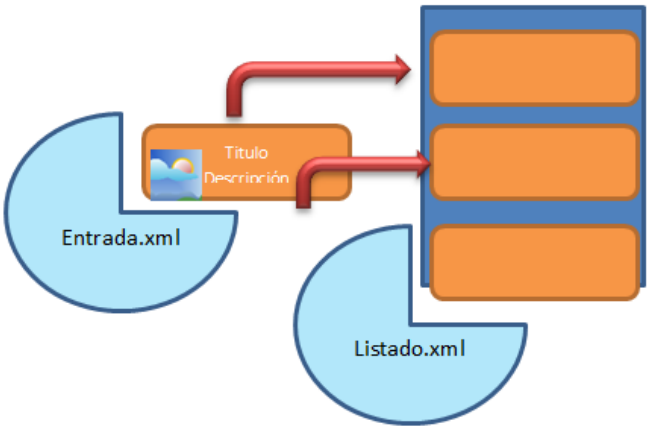


Figura 72.

Elaborado por: David Oña y Alexis Vaca

Diseño del Layout “**entrada.xml**”

Diseño de Layout

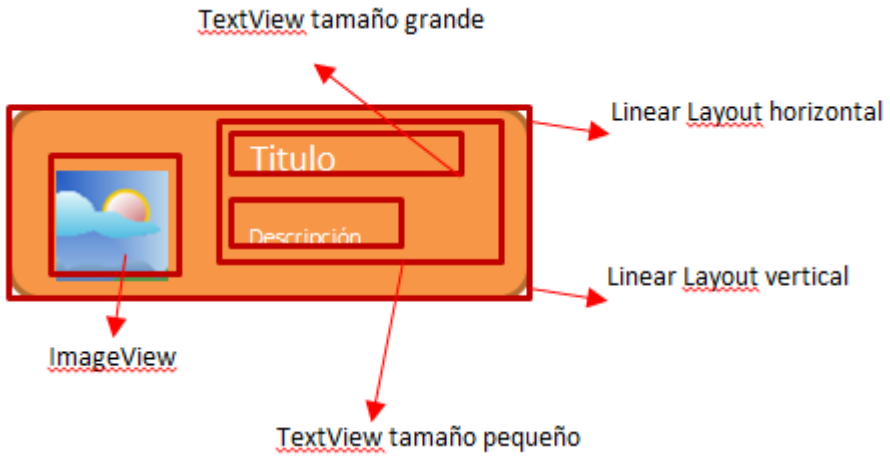


Figura 73.

Elaborado por: David Oña y Alexis Vaca

En la figura 74 se muestra el código Layout “**entrada.xml**”, en el cual el “id” se usa para luego referenciarlos y llenar cada una de las entradas con sus datos correspondientes.

Código Layout de entrada

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >

    <ImageView
        android:id="@+id/imageView_imagen"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:adjustViewBounds="true"
        android:scaleType="fitXY"
        android:contentDescription="Descripción del contenido de la imagen"
        android:src="@android:drawable/ic_menu_gallery" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/textView_superior"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Large Text"
            android:textAppearance="?android:attr/textAppearanceLarge" />

        <TextView
            android:id="@+id/textView_inferior"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Small Text"
            android:textAppearance="?android:attr/textAppearanceSmall" />

    </LinearLayout>

</LinearLayout>
```

Figura 74.

Elaborado por: David Oña y Alexis Vaca

En la figura 75 se muestra el código Layout "listado.xml", en el que se usa ListView, que es el contenedor de las entradas.

Código Layout de listado

```
<?xml version="1.0" encoding="utf-8"?>
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/ListView_listado"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
</ListView>
```

Figura 75.

Elaborado por: David Oña y Alexis Vaca

Los Datos se cargan y por medio de una Actividad "MainActivity.java" se construye el listado de forma dinámica, además se utilizara las clases “Lista_entrada.java”, la cual encapsula los datos de la entrada y “Lista_adaptador.java” es la que adapta las entradas a la lista. De cada entrada los parámetros: imagen, título y descripción serán datos para cada uno de los conjuntos de la entrada. Estos paquetes serán objetos de la clase “Lista_entrada.java”.

Clase **Lista_entrada.java**

Encapsulación de datos

Figura 76.

Elaborado por: David Oña y Alexis Vaca

Se crea los objetos de la clase “Lista_entrada.java”, cada uno almacenará sus datos (título, descripción e imagen). Estos grupos se llamar manejadores o en inglés “handler”.

handler

Figura 77.

Elaborado por: David Oña y Alexis Vaca

En la figura 78 se muestra el código que contiene la encapsulación de los siguientes datos:

- Título, con nombre de variable "textoEncima" para especificar en cuál de los dos TextViews de la vista tendrá que ser colocado.
- Descripción, con el nombre de variable "textoDebajo".
- Imagen, con el nombre de variable "idImagen", para referenciar con el id las imágenes.

Código encapsulación de datos

```
package com.androidya.proyecto010;

public class Lista_entrada {
    private int idImagen;
    private String textoEncima;
    private String textoDebajo;

    public Lista_entrada (int idImagen, String textoEncima, String textoDebajo) {
        this.idImagen = idImagen;
        this.textoEncima = textoEncima;
        this.textoDebajo = textoDebajo;
    }

    public String get_textoEncima() {
        return textoEncima;
    }

    public String get_textoDebajo() {
        return textoDebajo;
    }

    public int get_idImagen() {
        return idImagen;
    }
}
```

Figura 78.

Elaborado por: David Oña y Alexis Vaca

En la figura 79 se indica el código que se utiliza para encapsular el objeto "Lista_entrada":

Encapsulamiento del objeto

```
new Lista_entrada(R.drawable. imagen, "TÍTULO", "Descripción");
```

Figura 79.

Elaborado por: David Oña y Alexis Vaca

Para tener varios objetos se debe utilizar un ArrayList que se integra en la clase "MainActivity3.Java" en la cual añadiremos información extraída desde la base de datos de los diferentes sensores como el de temperatura, humedad, fuga de refrigerante.

En la figura 80 se muestra el código de configuración de los arreglos.

Código ArrayList

```
  Statement st = conn.createStatement();
      ResultSet rs = st.executeQuery(stsql2);
      Vector vec_temp = new Vector(4);
      Vector vec_hume = new Vector(4);
      Vector vec_fagua = new Vector(4);

while (rs.next ()) {
    double temp_decimal = Double.parseDouble(rs.getString(3));
    vec_temp.add("Fecha:"+rs.getString(1)+"Hora:"+rs.getString(2) +"Temp:"+temp_decimal);
    vec_hume.add("Fecha:"+rs.getString(1)+"Hora:"+rs.getString(2) +"Hume:"+rs.getString(4)+"%");
    vec_fagua.add("Fecha:"+rs.getString(1)+"Hora:"+rs.getString(2) +"Estado:"+rs.getString(5));

  }
datos.add(new Lista_entrada(R.drawable.im_temperatura, "Temperatura:"+ temp+"°C","Fecha:"+ fecha +" Hora:"+ hora+"" +
       "                                                                    "+"\n"+
          "5 Ultimos Registros de Temperatura:"+ vec_temp.toString()));

   datos.add(new Lista_entrada(R.drawable.im_humedad, "Humedad:"+hume+"%","Fecha:"+ fecha +" Hora:"+ hora+"" +
           "                                                                    "+"\n"+
           "5 Ultimos Registros de Humedad:"+ vec_hume.toString()));

   datos.add(new Lista_entrada(R.drawable.im_refrigerante, "Fuga Refrigerante:"+Fagua,"Fecha:"+ fecha +" Hora:"+ hora+""
           "                                                                    "+"\n"+
           "5 Ultimos Registros de Fuga Refrigerante:"+ vec_fagua.toString()));
```

Figura 80.

Elaborado por: David Oña y Alexis Vaca

3.9.4. Diagrama de actividades

En la figura 81 se muestra el diagrama de actividades de la aplicación MTI Android y su interacción.

Diagrama de actividades MTI

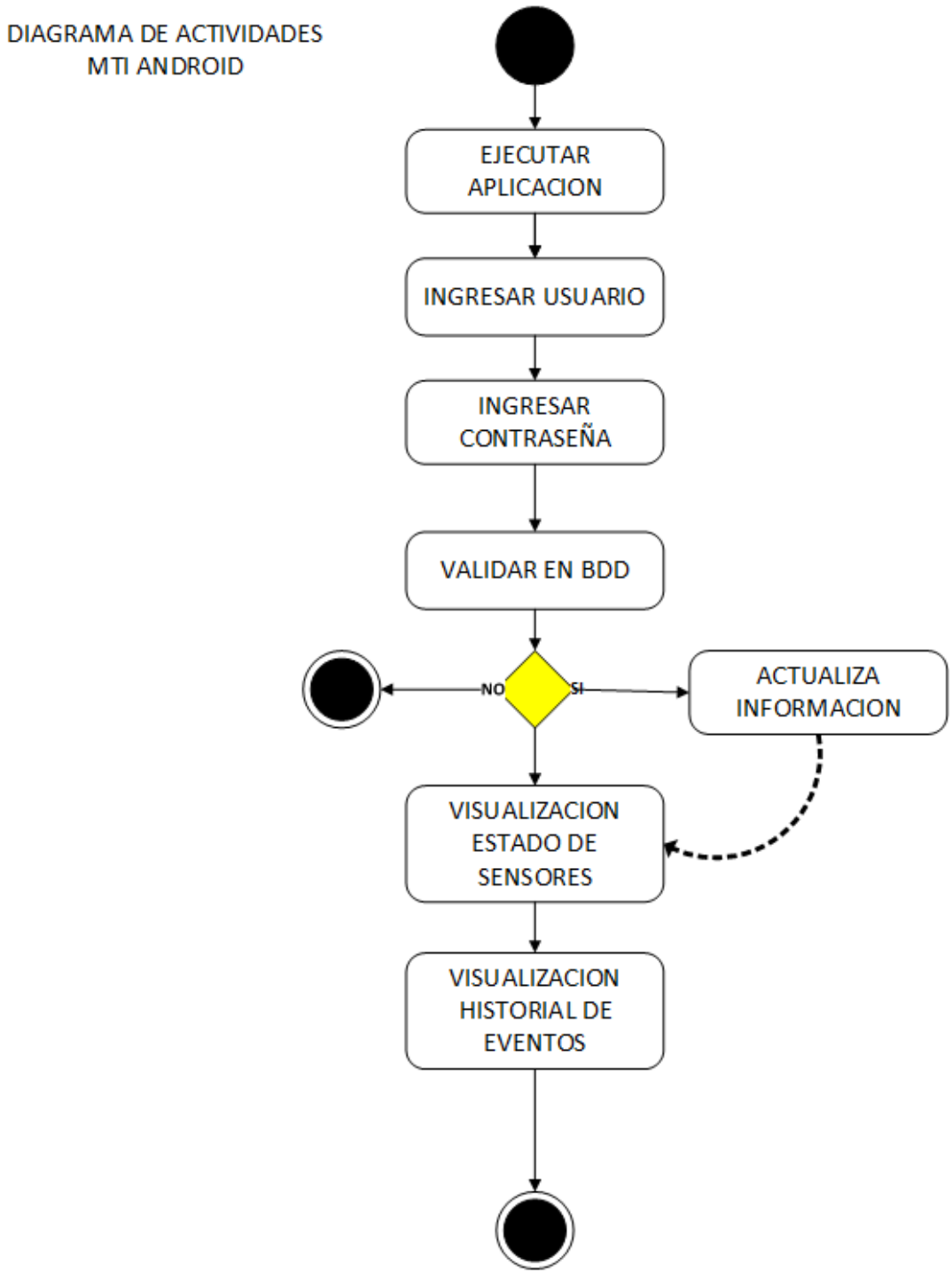


Figura 81.
Elaborado por: David Oña y Alexis Vaca

3.10. Pruebas de rendimiento del sensor de temperatura LM35 Y DHT11

Para realizar las pruebas de los sensores y determinar cuál es el más preciso para la aplicación MTI se procedió a someterlos a cambios de temperatura drásticos en intervalos de tiempo de 4 segundos con un tiempo total de la prueba de 5 minutos en temperaturas normales, altas y bajas como se muestra en la tabla 26.

Tabla 26.

Pruebas criticas de sensores

Tiempo	Sensor Análogo (LM35)	Sensor Digital (DHT11)		Tiempo	Sensor Análogo (LM35)	Sensor Digital (DHT11)		Tiempo	Sensor Análogo (LM35)	Sensor Digital (DHT11)	
22:00:00	20.51	20.00	TEMPERATURA NORMAL	22:01:48	16.60	20.00	TEMPERATURA BAJA	22:03:36	16.60	17.00	TEMPERATURA ALTA
22:00:04	21.00	20.00		22:01:52	18.07	19.00		22:03:40	21.48	17.00	
22:00:08	20.02	20.00		22:01:56	17.09	19.00		22:03:44	21.97	17.00	
22:00:12	20.51	20.00		22:02:00	16.11	19.00		22:03:48	22.46	17.00	
22:00:16	20.51	20.00		22:02:04	15.63	19.00		22:03:52	24.90	17.00	
22:00:20	20.51	20.00		22:02:08	15.58	18.00		22:03:56	26.37	17.00	
22:00:24	20.51	20.00		22:02:12	14.65	18.00		22:04:00	27.83	17.00	
22:00:28	20.51	20.00		22:02:16	15.14	18.00		22:04:04	28.32	17.00	
22:00:32	20.51	20.00		22:02:20	15.63	18.00		22:04:08	29.30	17.00	
22:00:36	20.02	20.00		22:02:24	15.63	18.00		22:04:12	29.30	17.00	
22:00:40	19.53	20.00		22:02:28	14.65	18.00		22:04:16	28.32	17.00	
22:00:44	19.53	20.00		22:02:32	14.65	18.00		22:04:20	28.81	17.00	
22:00:48	20.51	20.00		22:02:36	13.67	17.00		22:04:24	28.81	17.00	
22:00:52	20.51	20.00		22:02:40	12.70	17.00		22:04:28	27.83	17.00	
22:00:56	20.51	20.00		22:02:44	13.67	17.00		22:04:32	26.86	17.00	
22:01:00	20.51	20.00		22:02:48	12.70	17.00		22:04:36	27.34	17.00	
22:01:04	20.51	20.00		22:02:52	11.72	17.00		22:04:40	26.86	17.00	
22:01:08	20.51	20.00		22:02:56	10.25	17.00		22:04:44	26.86	17.00	
22:01:12	21.00	20.00		22:03:00	18.55	17.00		22:04:48	26.37	17.00	
22:01:16	20.51	20.00		22:03:04	10.74	17.00		22:04:52	26.37	17.00	
22:01:20	20.02	20.00		22:03:08	10.25	17.00		22:04:56	25.88	17.00	
22:01:24	20.51	20.00		22:03:12	10.25	17.00		22:05:00	25.88	17.00	
22:01:28	20.02	20.00		22:03:16	9.77	17.00		22:05:04	25.88	17.00	
22:01:32	19.53	20.00		22:03:20	9.28	17.00		22:05:08	25.39	17.00	
22:01:36	20.02	20.00		22:03:24	9.28	17.00		22:05:12	25.39	17.00	
22:01:40	20.51	20.00		22:03:28	6.84	17.00		22:05:16	24.41	17.00	
22:01:44	20.02	20.00		22:03:32	5.37	17.00		22:05:20	24.90	17.00	

Nota. Cambios bruscos de temperatura en un determinado tiempo.

Elaborado por: David Oña y Alexis Vaca

La figura 82 muestra la relación del sensor digital DHT11 vs el sensor análogo LM35 en el tiempo, en el cual se evidencia que el sensor LM35 tiene una mejor respuesta ante cambios críticos de temperatura.

Gráfico estadístico

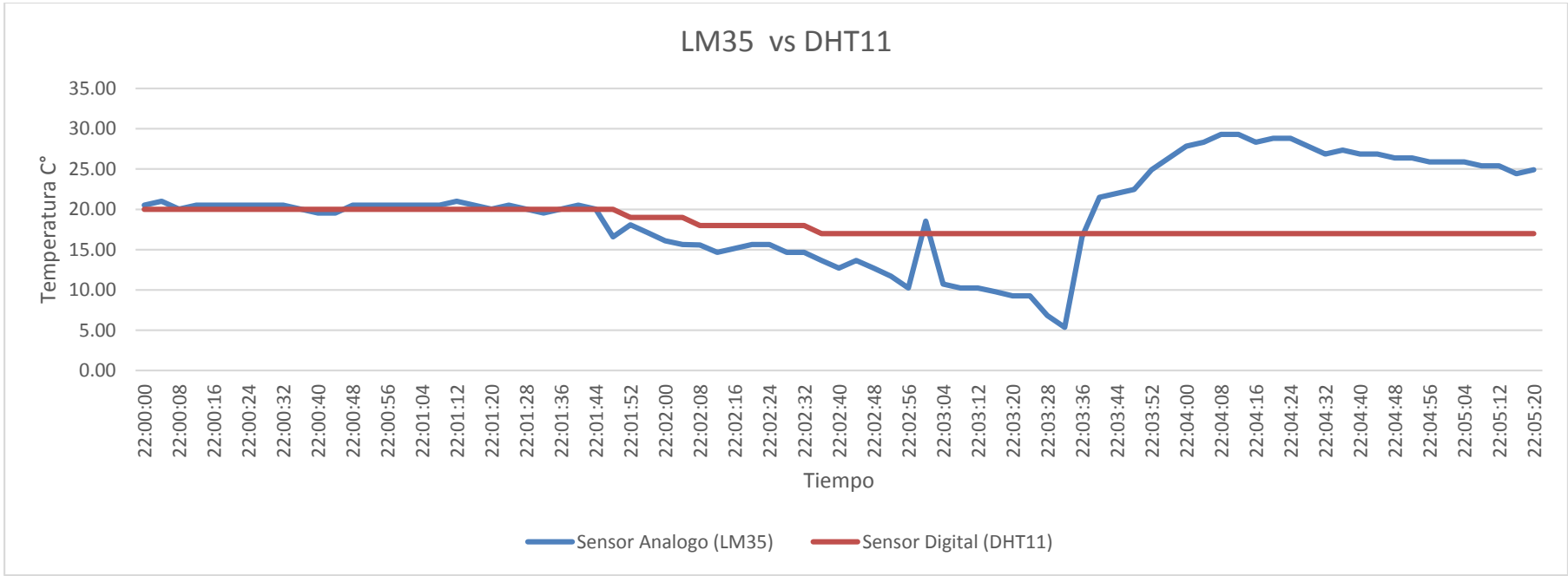


Figura 82.
Elaborado por: David Oña y Alexis Vaca

3.11. Pruebas en el Data Center de la EPMAPS

Para realizar la pruebas del funcionamiento total del MTI se escogió 3 días de monitoreo, lo primero es exporta un backup de la base de datos en el que se han guardado por cada uno de los eventos producidos en el puerto serie de ArduinoUno estos datos se pasan al programa Excel para interpretarlo y graficarlo.

Se tienen tres gráficas del comportamiento una por cada día escogido en las cuales se producen oscilaciones y variaciones del modelo ideal el cual muestra la humedad y temperatura linealmente.

Gráfico estadístico temperatura, humedad

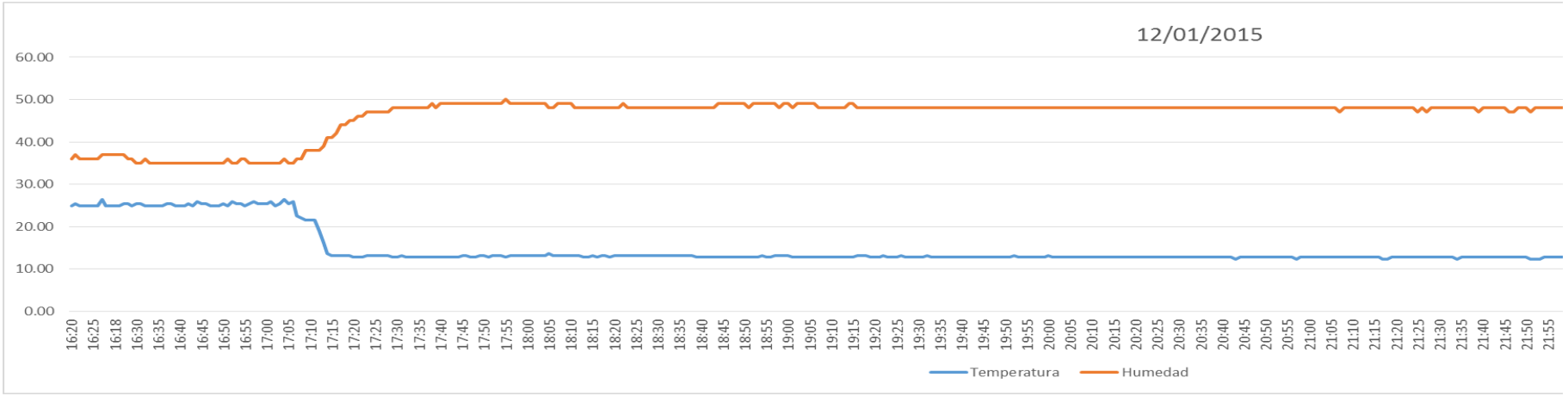


Figura 83.
Elaborado por: David Oña y Alexis Vaca

Gráfico estadístico temperatura, humedad

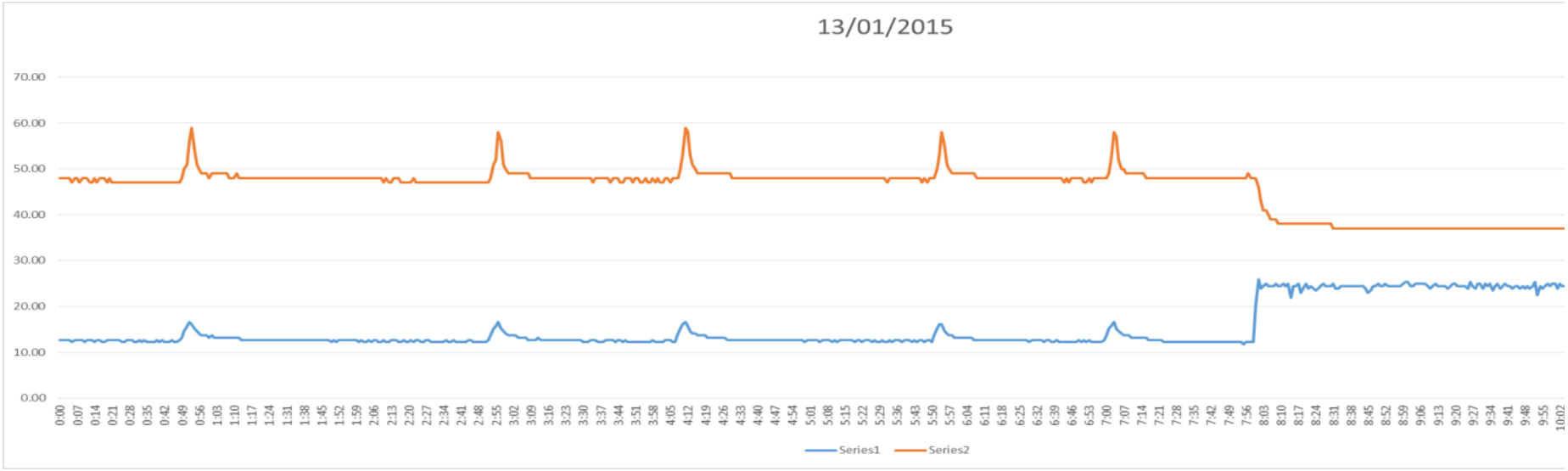


Figura 84.
Elaborado por: David Oña y Alexis Vaca

Gráfico estadístico temperatura, humedad

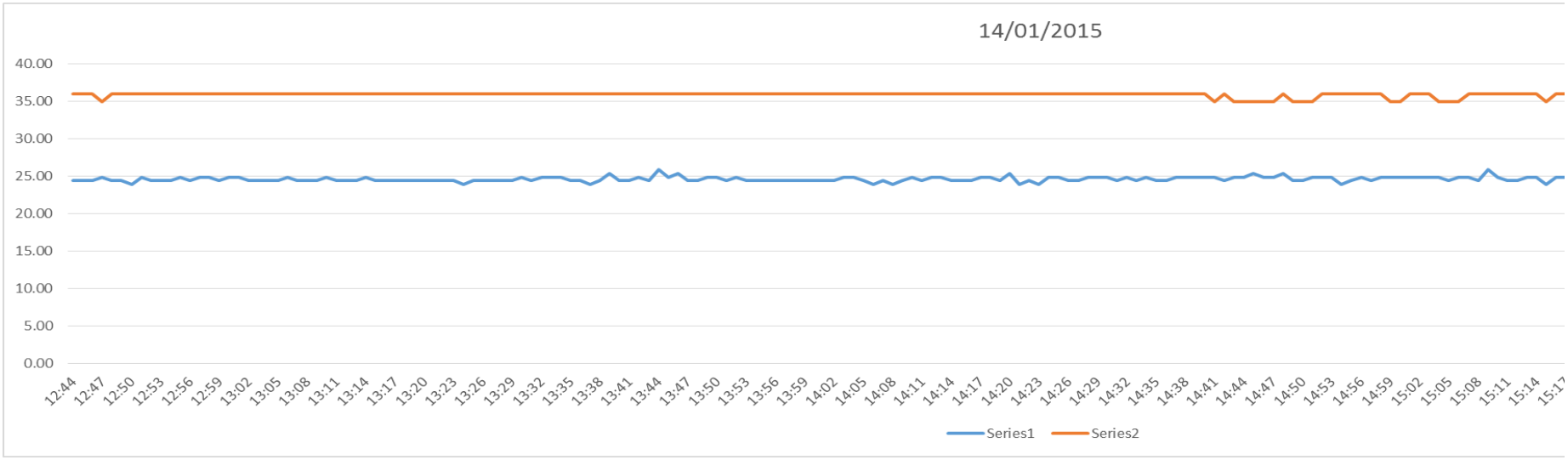


Figura 85.
Elaborado por: David Oña y Alexis Vaca

En las figuras 83, 84 y 85 se muestra las gráficas de humedad y temperatura donde se observa los diferentes eventos ocurridos durante los 3 días de pruebas, estas pruebas que se realizaron con el módulo MTI sirvió para verificar que la EPMAPS contaba con un problema en el sistema de aire acondicionado el cual fue confirmado por el sistema MTI, y se demuestra que está funcionando correctamente en el Data Center.

3.12. Duración de la batería módulo backup

Batería de stamina 9V

Figura 86.
Elaborado por: David Oña y Alexis Vaca

La vida útil de una batería se calcula en base a la corriente dada en miliamperios por hora y se abrevia mAh. La vida útil se puede calcular a partir de la corriente nominal de entrada de la batería y la corriente de carga del circuito.

La cantidad máxima de corriente que es capaz de entregar la batería en una hora es de 595 mAh, como se muestra en la tabla 27

Tabla 27.
Especificaciones técnicas batería stamina 9V

Marca / Modelo	Sony stamina Plus
Precio	$ 3,01
Voltaje	9,00 V
Tensión media durante la descarga	7.22 V
Capacidad de medida (descarga principal)	595 ± 1 % mAh
Energía	4,31 W/h

Nota. mAh= miliamperios por hora, V= voltaje
Fuente: Battery test.

Se procede a calcular la intensidad de corriente entregada por la batería de 9V al circuito backup de Arduino, por medio de un multímetro digital que nos muestra los datos en obtenidos en miliamperios (mA), a en la figura 87 se muestra el diagrama de

conexión para proceder con el cálculo correspondiente al tiempo de duración de la batería.

Diagrama de conexiones batería

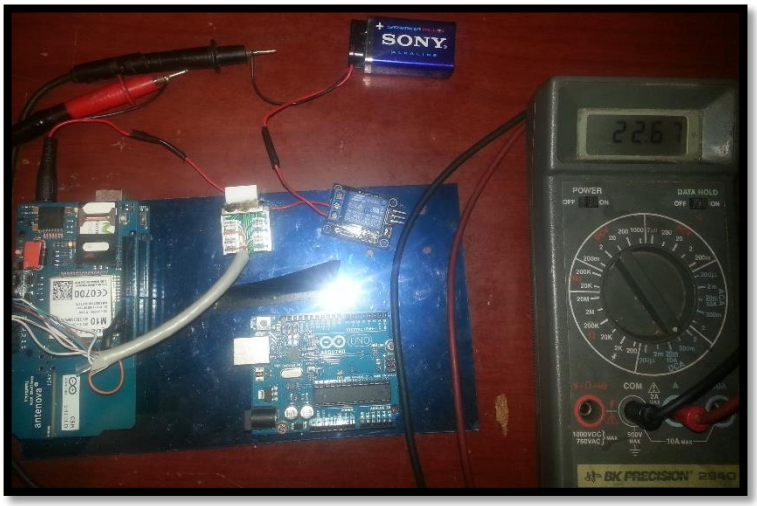


Figura 87.
Elaborado por: David Oña y Alexis Vaca

La capacidad de una batería se mide en mAh. Esta cifra nos da la duración de la batería, mediante la siguiente fórmula:

$$\text{Capacidad de la batería (mAh)} = Consumo\ del\ dispositivo\ (mA) * \text{ Vida útil de la batería } (h)$$

$$\text{Vida útil de la batería} = \frac{\text{Capacidad de la batería (mAh)}}{Consumo\ del\ dispositivo\ (mA)}$$

$$\text{Vida útil de la batería} = \frac{595\ \text{(mAh)}}{26.25\ (mA)}$$

$$\text{Vida útil de la batería} = 26.25\ horas$$

Con una duración aproximada de 26.25 horas de suministro de energía ininterrumpido para el modulo Back-up es suficiente para el envío de SMS a todo el personal de infraestructura y seguridad de la EPMAPS en el cual se informe "Falla total del

sistema" debido a un corte total del suministro de energía entregado por el UPS del Data Center, con lo cual se pueda tomar acciones urgentes ante el incidente presentado.

3.13. Pruebas de la aplicación web.

Una vez desarrollado los requisitos del Sprint Backlog junto con los casos de uso identificados anteriormente, se procede a hacer pruebas de funcionamiento de la aplicación web, se hace una serie de pruebas para observar el funcionamiento y los controles implementados e identificar las posibles fallas. Se realizaron las siguientes pruebas.

Tabla 28.
Prueba requerimiento 1

Prueba de aceptación aplicación web	
Número: 1	Requisito 1: Página inicial y de acceso a la aplicación.
Nombre: Prueba página inicial y de acceso a la aplicación	
Descripción: Se procede a abrir la página login.jsp y visualizar el ingreso de usuarios.	
Condiciones: El usuario debe registrarse para acceder.	
Pasos de ejecución: Se abre la interface del index.jsp. Se visualiza el ingreso de usuarios.	
Resultado: Visualización del index y del registro de usuarios de la página.	
Evaluación de la prueba: Excelente. Se cumplió con el resultado esperado.	

Nota. Pruebas de interacción del requerimiento 1

Elaborado por: David Oña y Alexis Vaca

Tabla 29.
Prueba requerimiento 2

Prueba de aceptación aplicación web	
Número: 2	Requisito 2: Control de acceso a la aplicación.
Nombre: Prueba del control de acceso a la aplicación.	
Descripción: Se ingresa el usuario y contraseña	
Condiciones: Ser usuario registrado.	
Pasos de ejecución: Se ingresa el usuario. Se ingresa la contraseña.	

Se presiona el botón entrar
Resultado: Visualización del sistema para usuarios registrados.
Evaluación de la prueba: Excelente. Se cumplió con el resultado esperado.

Nota. Pruebas de interacción del requerimiento 2
Elaborado por: David Oña y Alexis Vaca

Tabla 30.
Prueba requerimiento 3

Prueba de aceptación aplicación web	
Número: 3	Requisito 3: Prueba usuario no registrado
Nombre: Prueba de diseño de la página principal y menús.	
Descripción: Se ingresa a la aplicación web visualiza la página de logueo de usuarios y sale del sistema	
Condiciones: No puede acceder a nada.	
Pasos de ejecución: Ingresar a la aplicación web. Salir de la página.	
Resultado: visualización de la página de logueo	
Evaluación de la prueba: Excelente. Se cumplió con el resultado esperado.	

Nota. Pruebas de interacción del requerimiento 3
Elaborado por: David Oña y Alexis Vaca

Tabla 31.
Prueba requerimiento 4

Prueba de aceptación aplicación web	
Número: 4	Requisito 4: Diseño de la página principal y menús
Nombre: Prueba de diseño de la página principal y menús.	
Descripción: Se da clic en uno de los botones del menú de navegación, se abre una nueva página y se visualiza.	
Condiciones: La páginas del menú de navegación debe existir.	
Pasos de ejecución: Clic en uno de los botones del menú de navegación de la página principal. Aparece la página vinculada al botón del menú de navegación Se visualiza los datos de la pagina	
Resultado: Se dio clic en un botón del menú de navegación y se visualiza una nueva página.	
Evaluación de la prueba: Excelente. Se cumplió con el resultado esperado.	

Nota. Pruebas de interacción del requerimiento 4
Elaborado por: David Oña y Alexis Vaca

Tabla 32.
Prueba requerimiento 5

Prueba de aceptación aplicación web	
Número: 5	Requisito 5: Registro nuevo usuario
Nombre: Prueba de registro de nuevo usuario.	
Descripción: dentro de la página de usuarios se procede a llenar el formulario con los datos solicitados.	
Condiciones: El administrador crea un nuevo usuario.	
Pasos de ejecución: Clic en el botón de registro de usuarios en el menú de navegación. Ingresar el usuario Ingresar contraseña Ingresar Email Ingresar numero celular Presionar botón Agregar	
Resultado: Visualización del usuario ingresado correctamente.	
Evaluación de la prueba: Excelente. Se cumplió con el resultado esperado.	

Nota. Pruebas de interacción del requerimiento 5
Elaborado por: David Oña y Alexis Vaca

Finalmente se procede a realizar la última tabla que es parte de la fase de pruebas que corresponde a la prueba de configuración de umbrales, la cual es un poco más extensa debido a la cantidad de datos que el administrador debe llenar para realizar los cambios.

Tabla 33.

Prueba requerimiento 6

Prueba de aceptación aplicación web	
Número: 6	Requisito 6: Configuración de umbrales y estados de carga del UPS
Nombre: Prueba de configuración de umbrales.	
Descripción: dentro de la página de configuración de umbrales se procede a llenar el formulario con los datos solicitados.	
Condiciones: El administrador puede modificar el valor de los umbrales.	
Pasos de ejecución: Clic en el botón de umbrales en el menú de navegación. Ingresar valor de temperatura Ingresar valor de humedad Ingresar valor de entrada volt fase 1 Ingresar valor de entrada volt fase 2 Ingresar valor de entrada volt fase 3 Ingresar valor de salida volt fase 1 Ingresar valor de salida volt fase 2 Ingresar valor de salida volt fase 3 Tensión de la batería	

Capacidad de la batería Presionar el botón Realizar cambios
Resultado: Visualización de los umbrales realizados correctamente.
Evaluación de la prueba: Excelente. Se cumplió con el resultado esperado.

Nota. Pruebas de interacción del requerimiento 6
Elaborado por: David Oña y Alexis Vaca

3.14. Pruebas aplicación Android

Una vez desarrollada la aplicación Android que se la llamo MTI para Android se procede a hacer pruebas de funcionamiento, se hace una serie de pruebas para observar el funcionamiento y los controles implementados e identificar las posibles fallas. Se ejecuta la aplicación en el emulador y la aplicación Android con extensión .apk es instalada en un terminal móvil Samsung Galaxy S4 con las siguientes especificaciones técnicas mostradas en la tabla 30

Tabla 34.
Prueba Android requisito 1

Prueba de aceptación aplicación Android	
Número: 1	Requisito 1: Diseño de la Interface y login de usuario
Nombre: Prueba del diseño de la interface principal y login	
Descripción: Dentro de la aplicación navegar por la pantalla y loguearse	
Condiciones: ser usuario registrado	
Pasos de ejecución: Abrir la aplicación. Navegar por la pantalla. Ingresar usuario Ingresar contraseña Presionar el botón Ingresar	
Resultado: Visualización de la pantalla principal y login de usuario.	
Evaluación de la prueba: Excelente. Se cumplió con el resultado esperado.	

Nota. Pruebas de interacción del requerimiento 1

Elaborado por: David Oña y Alexis Vaca

Tabla 35.
Prueba Android requisito 2

Prueba de aceptación aplicación Android	
Número: 2	Requisito 2: Visualización de eventos en la aplicación
Nombre: Prueba de la visualización del estado de carga del UPS y umbrales	
Descripción: visualizar los valores de los umbrales y del estado de carga del UPS dentro del Data center	
Condiciones: ser usuario registrado	
Pasos de ejecución: Visualizar nueva pantalla. Visualizar estado de umbrales : Humedad Temperatura Visualizar estado de fuga de agua Visualizar estado de carga del UPS	
Resultado: Visualización de los estados de carga del UPS y los valores de los umbrales	
Evaluación de la prueba: Excelente. Se cumplió con el resultado esperado.	

Nota. Pruebas de interacción del requerimiento 2

Elaborado por: David Oña y Alexis Vaca

Tabla 36.
Prueba Android requisito 3

Prueba de aceptación aplicación Android	
Número: 3	Requisito 3: Visualización de las ultimas 10 lecturas de Arduino
Nombre: Prueba de la visualización de lecturas de Arduino	
Descripción: visualizar las ultimas 10 lecturas echas por Arduino	
Condiciones: ninguno	
Pasos de ejecución: Presionar un botón de los valores de Arduino. Visualizar las lecturas de Arduino Salir de la aplicación.	
Resultado: Visualización de las lecturas echas por Arduino	
Evaluación de la prueba: Excelente. Se cumplió con el resultado esperado.	

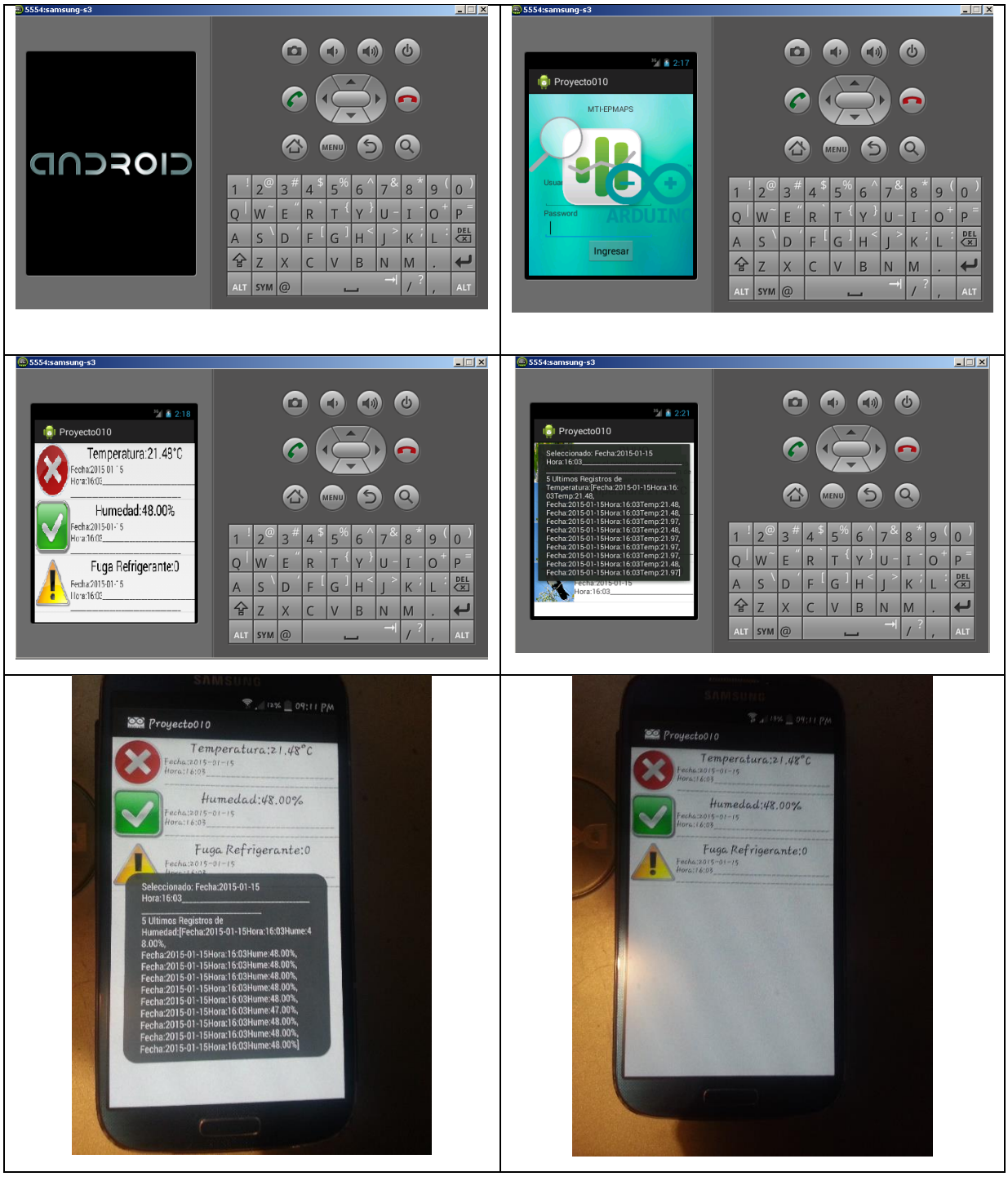
Nota. Pruebas de interacción del requerimiento 3

Elaborado por: David Oña y Alexis Vaca

En la tabla 37 se muestra las pruebas tanto en el simulador SDK como en el celular de marca Samsung Galaxy S4

Tabla 37.

Aplicación Android



Nota. Pantalla de pruebas aplicación Android.
Elaborado por: David Oña y Alexis Vaca

3.15. Pruebas de funcionamiento del MTI

Una vez finalizado el diseño del hardware y desarrollo del software del MTI, se procede a realizar pruebas de funcionamiento.

En estas pruebas se observa mensajes enviados por ArduinoUno a través de la consola de Java, envío de SMS, correo electrónico y visualización en la pantalla principal por medio de un navegador WEB, cuando fallan los diferentes sensores ambientales y el estado de carga del UPS.

3.16. Prueba de fuga de refrigerante

Cuando el sensor de fugas de refrigerante detecta liquido en la sonda, dispara automáticamente por medio de ArduinoUno un mensaje a la consola de Java el cual es procesado por el sistema MTI y enviado nuevamente a la placa ArduinoUno con un mensaje indicando los destinatarios y el nombre del sensor que esta fallando.

3.16.1. Consola Java, prueba fuga de refrigerante

en la figura 88 se muestra el monitoreo en tiempo real de los sensores en la consola de Java, y la falla del sensor de fuga de refirgerante

Consola Java prueba de sensor de refrigerante

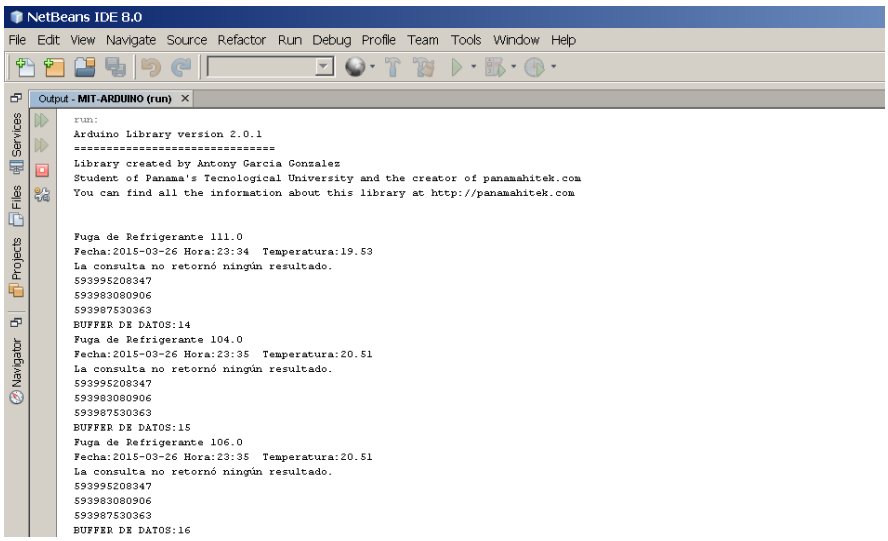


Figura 88.
Elaborado por: David Oña y Alexis Vaca

3.16.2. Envío de SMS a destinatarios, prueba fuga de refrigerante

En la figura 89 se muestra la recepción de los SMS a los diferentes destinatarios enviados a través del sistema MTI.

Recepción SMS

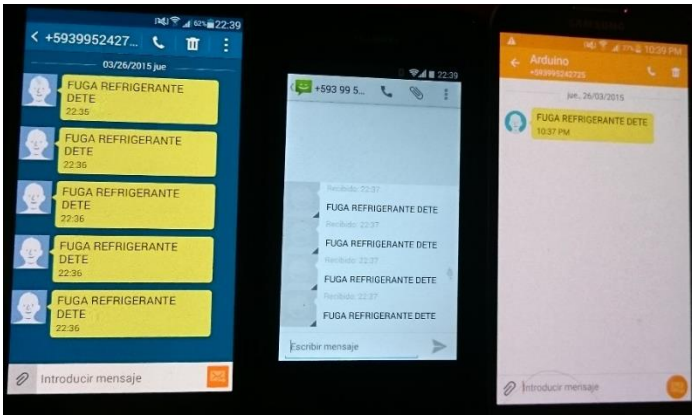


Figura 89.
Elaborado por: David Oña y Alexis Vaca

3.16.3. Correo electrónico, prueba fuga de refrigerante

En la figura 90 se muestra la recepción del correo electrónico a un destinatario indicando que se ha detectado una fuga en el refrigerante.

Recepción correo electrónico

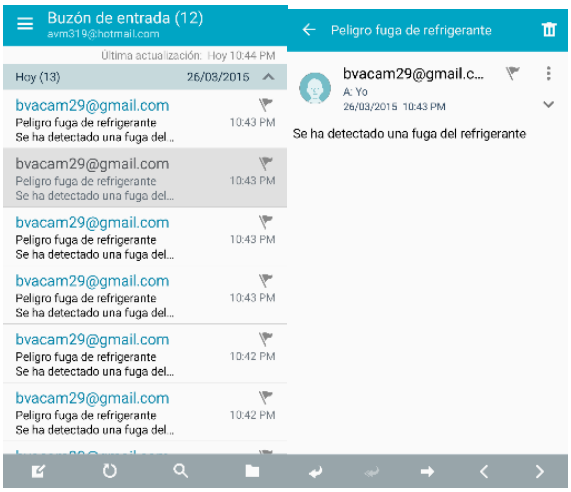


Figura 90.
Elaborado por: David Oña y Alexis Vaca

3.16.4. Interfaz MTI

En la figura 91 se visualiza la interfaz web del sistema MTI en el cual cambia el estado del sensor de fuga de refrigerante a un estado de error.

Interfaz web visualización sensor de fugas

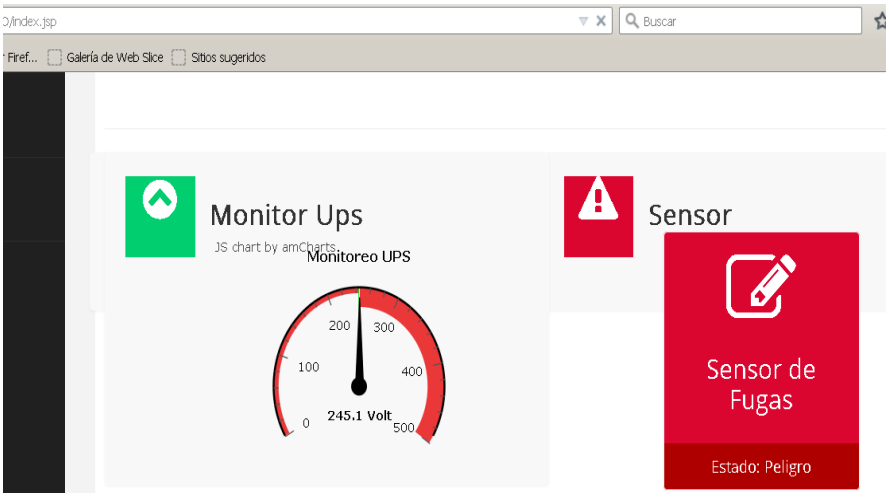


Figura 91.
Elaborado por: David Oña y Alexis Vaca

3.17. Prueba de humedad y temperatura

Como se muestra en la figura 92 el monitoreo en tiempo real de los sensores de temperatura y humedad y la falla de los mismos.

3.17.1. Consola de Java, prueba de humedad y temperatura

Consola Java prueba de sensor de humedad y temperatura

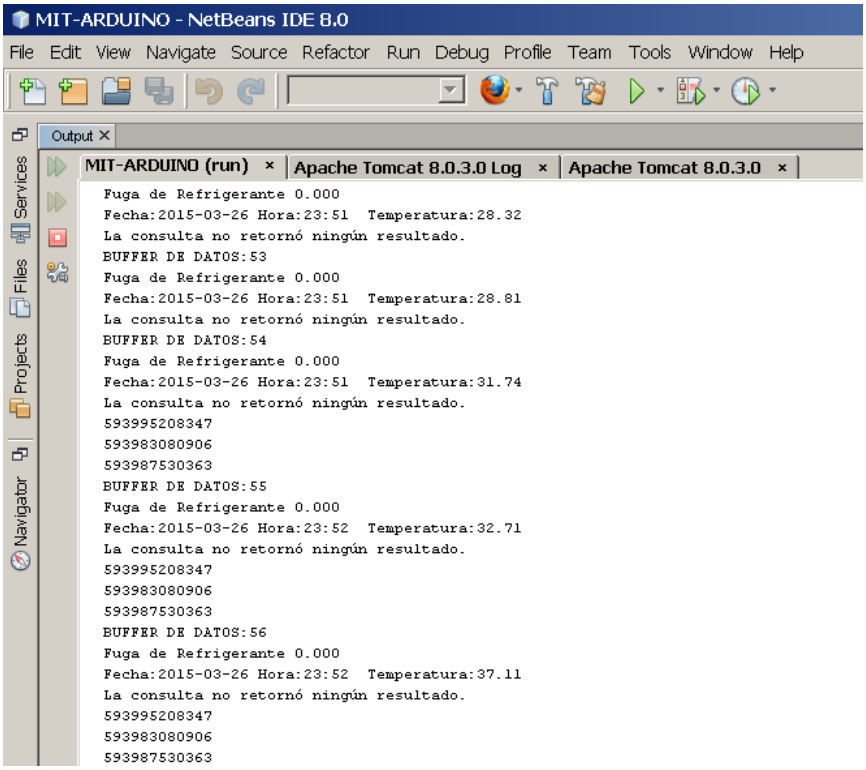


Figura 92.
Elaborado por: David Oña y Alexis Vaca

3.17.2. Envío de SMS, prueba de humedad y temperatura

En la figura 93 se muestra la recepción de los SMS a los diferentes destinatarios enviados a través del sistema MTI, en el cual se indica que la temperatura ha fallado.

Recepción SMS

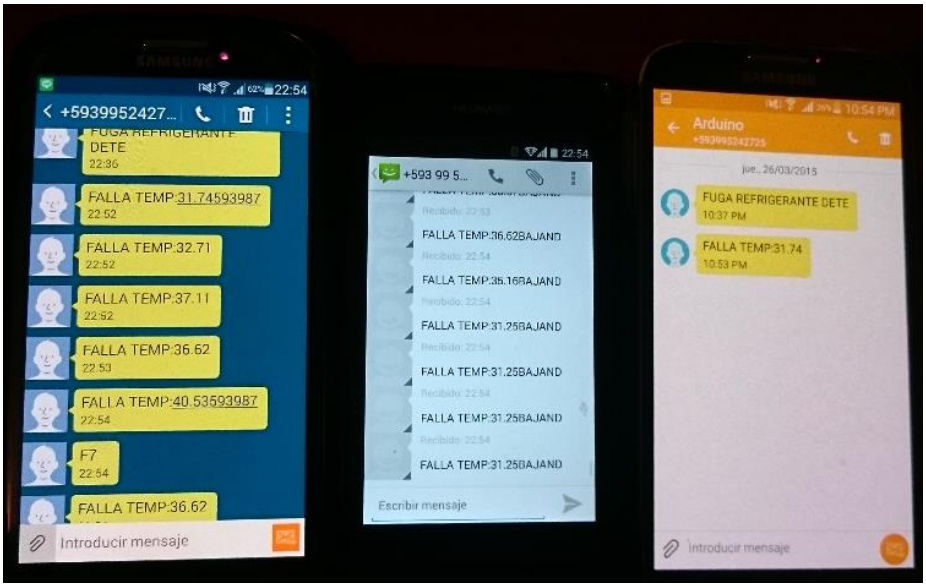


Figura 93.
Elaborado por: David Oña y Alexis Vaca

3.17.3. Correo electrónico, prueba de humedad y temperatura

En la figura 94 se muestra la recepción del correo electrónico a un destinatario indicando que se ha detectado una fuga en el refrigerante.

Recepción correo electrónico

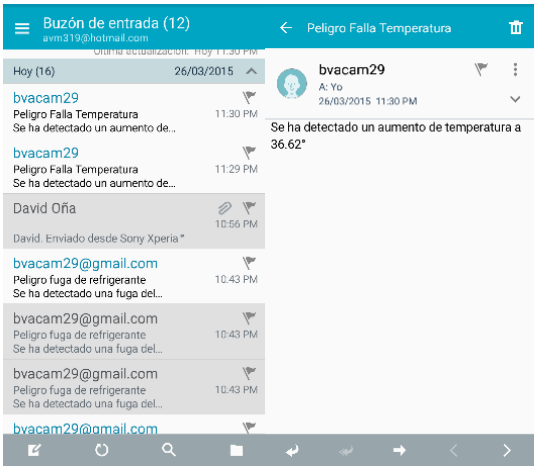


Figura 94.
Elaborado por: David Oña y Alexis Vaca

3.17.4. Interfaz MTI

En la figura 95 se visualiza la interfaz web del sistema MTI en el cual cambia el estado del sensor temperatura y humedad a un estado de error.

Interfaz web visualización sensor humedad y temperatura

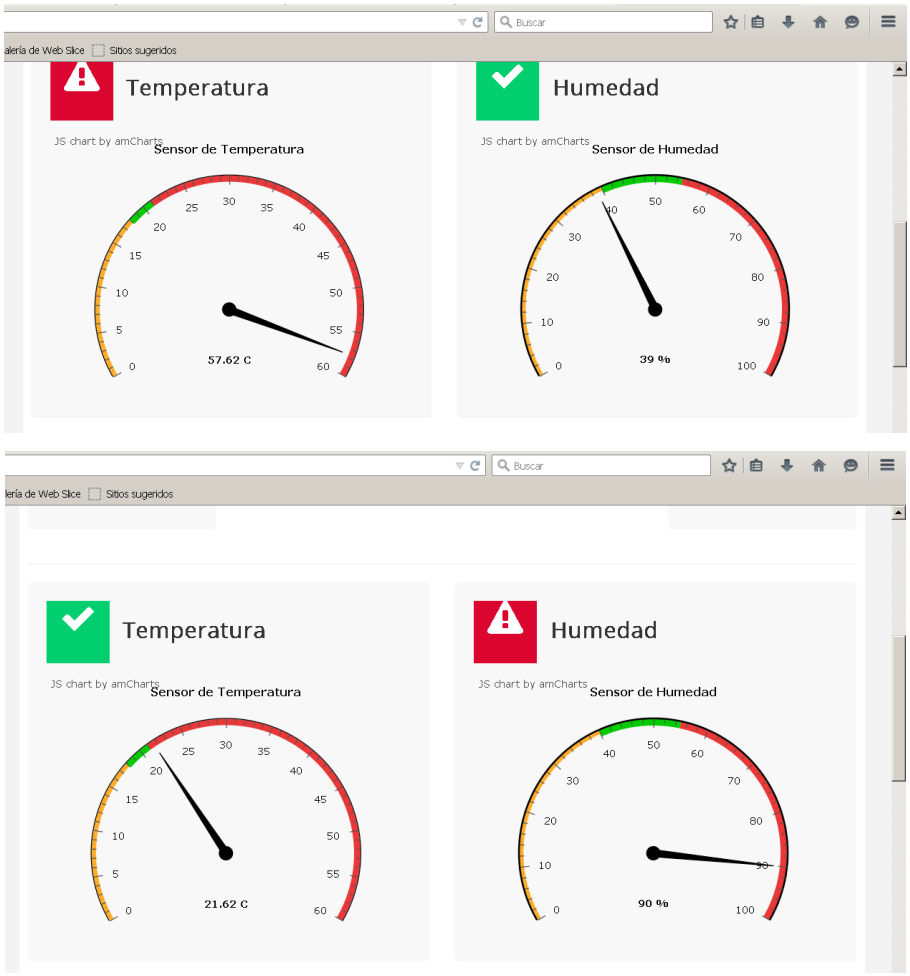


Figura 95.
Elaborado por: David Oña y Alexis Vaca

3.18. Prueba de incidentes del UPS

3.18.1. Consola Java, pruebas incidentes UPS

La figura 96 muestra el monitoreo y sincronización del UPS con un retardo de un minuto en durante la sincronización con los logs.

Consola Java sincronización UPS

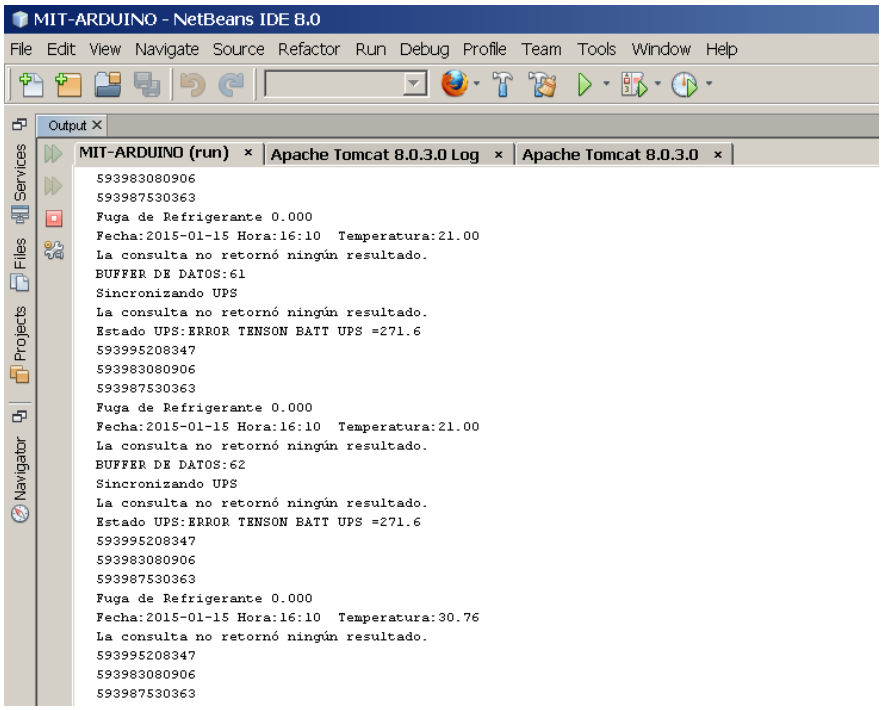


Figura 96.
Elaborado por: David Oña y Alexis Vaca

3.18.2. Envío de SMS, pruebas incidentes UPS

En la figura 97 se muestra la recepción de los SMS a los diferentes destinatarios enviados a través del sistema MTI, en el cual se indica que que la tensión de la batería.

Recepción SMS

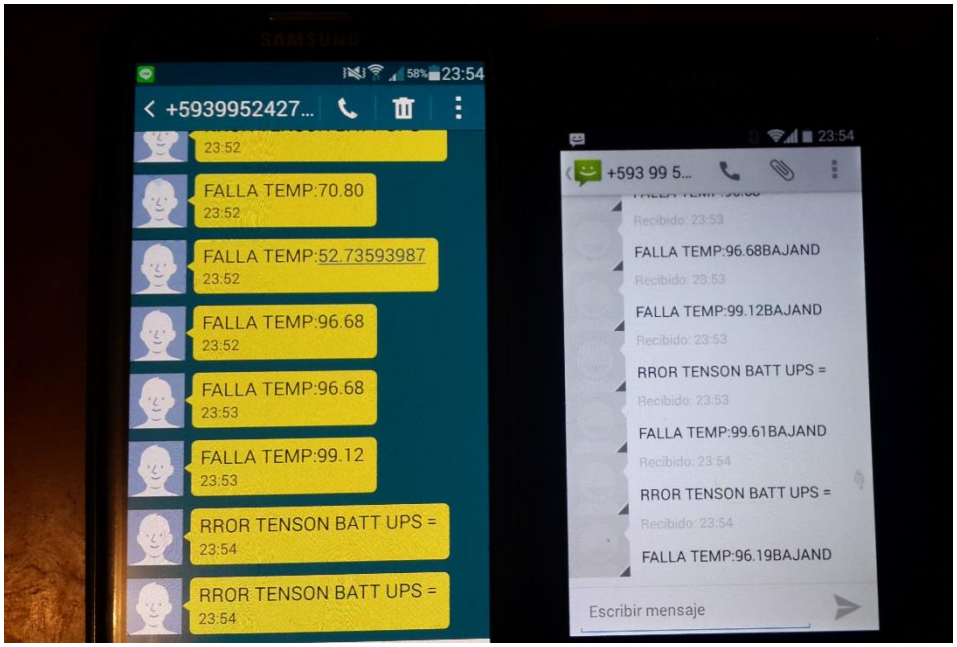


Figura 97.
Elaborado por: David Oña y Alexis Vaca

3.18.3. Correo electrónico, pruebas incidentes UPS

En la figura 98 se muestra la recepción del correo electrónico a un destinatario indicando que se el nivel de la batería de un 80%.

Recepción correo electrónico

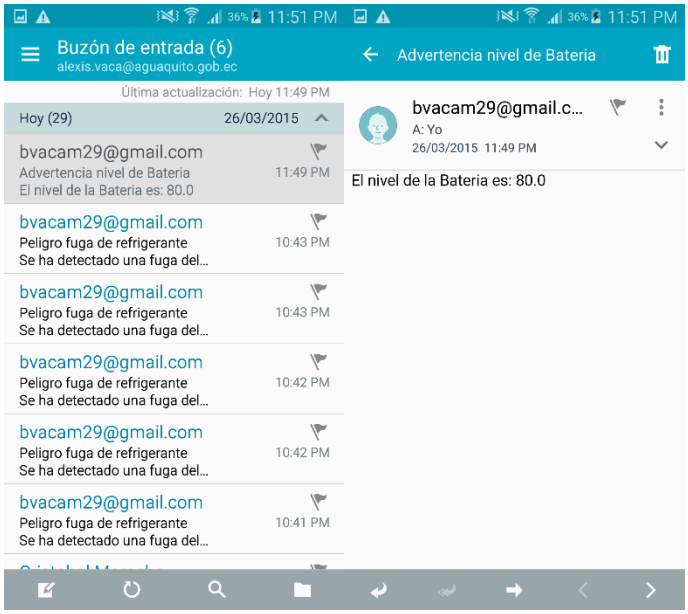


Figura 98.
Elaborado por: David Oña y Alexis Vaca

3.18.4. Interfaz MTI

En la figura 99 se visualiza la interfaz web del sistema MTI en el valor de la capacidad de la batería.

Interfaz web visualización estados del UPS

Estado UPS:

Fecha	Hora	Ent volt Fase 1	Ent volt Fase 2	Ent volt Fase 3	Sal volt Fase 1	Sal volt Fase 2	Sal volt Fase 3	Volt Bateria	Capacidad Bateria
2015-01-15	16:11:42	245.1	244.7	246.2	230.2	230.7	230.3	271.6	80.0
2015-01-15	16:11:42	245.1	244.7	246.2	230.2	230.7	230.3	271.6	100.0
2015-01-15	16:11:42	245.1	244.7	246.2	230.2	230.7	230.3	271.6	100.0
2015-01-15	14:16:09	245.1	244.7	246.2	230.2	230.7	230.3	271.6	100.0
2015-01-15	14:16:09	245.1	244.7	246.2	230.2	230.7	230.3	271.6	100.0

Estado de Carga del Banco de Baterias

Bateria Advertencia

Estado Entrada Volt Fase 1

Normal

Figura 99.
Elaborado por: David Oña y Alexis Vaca

CAPÍTULO 4

IMPLEMENTACIÓN DEL SISTEMAS DE MONITOREO

4.1. Análisis de requisitos

4.1.1. Hardware MTI

Tabla 38.
Requerimientos de hardware

Tarjetas	ArduinoUno GsmShield
Componentes eléctricos	Relay Fotocelda Resistencia
Conectores	Rj45 macho Rj45 hembra
Sensores	LM35 DHT11 DHT22
Batería	9V stamina

Nota. V= voltios.
Elaborado por: David Oña y Alexis Vaca

4.1.2. Servidor web

Tabla 39.
Requerimientos de software

Sistema Operativo	Windows 7 Pro Sp1
Ram	4 GB
Tarjeta Ethernet	10/100/1000
Disco Duro	500 GB

Nota. GB= gigabyte
Elaborado por: David Oña y Alexis Vaca

4.1.3. Aplicación Android

Tabla 40.

Requerimientos Android

Pantalla	5 pulgadas Full HD Super AMOLED (1920 x 1080), 441 ppi
Procesador	Quad-Core a 1,9 Ghz
Memoria RAM	2 gigas
Sistema Operativo	Android 4.4.2 (KitKat)
Conectividad	WiFi 802.11 a/b/g/n/ac GPS NFC Bluetooth® 4.0

Nota. GPS= sistema de posicionamiento global, NFC= comunicación de campo cercano.
Elaborado por: David Oña y Alexis Vaca

4.2. Implementación hardware

Una vez realizado el diseño y las respectivas pruebas se inicia la implementación del módulo ArduinoUno en la figura 100 se muestra el diagrama de la ubicación de los sensores en el Data Center de la EPMAPS.

Diagrama de ubicación Data Center

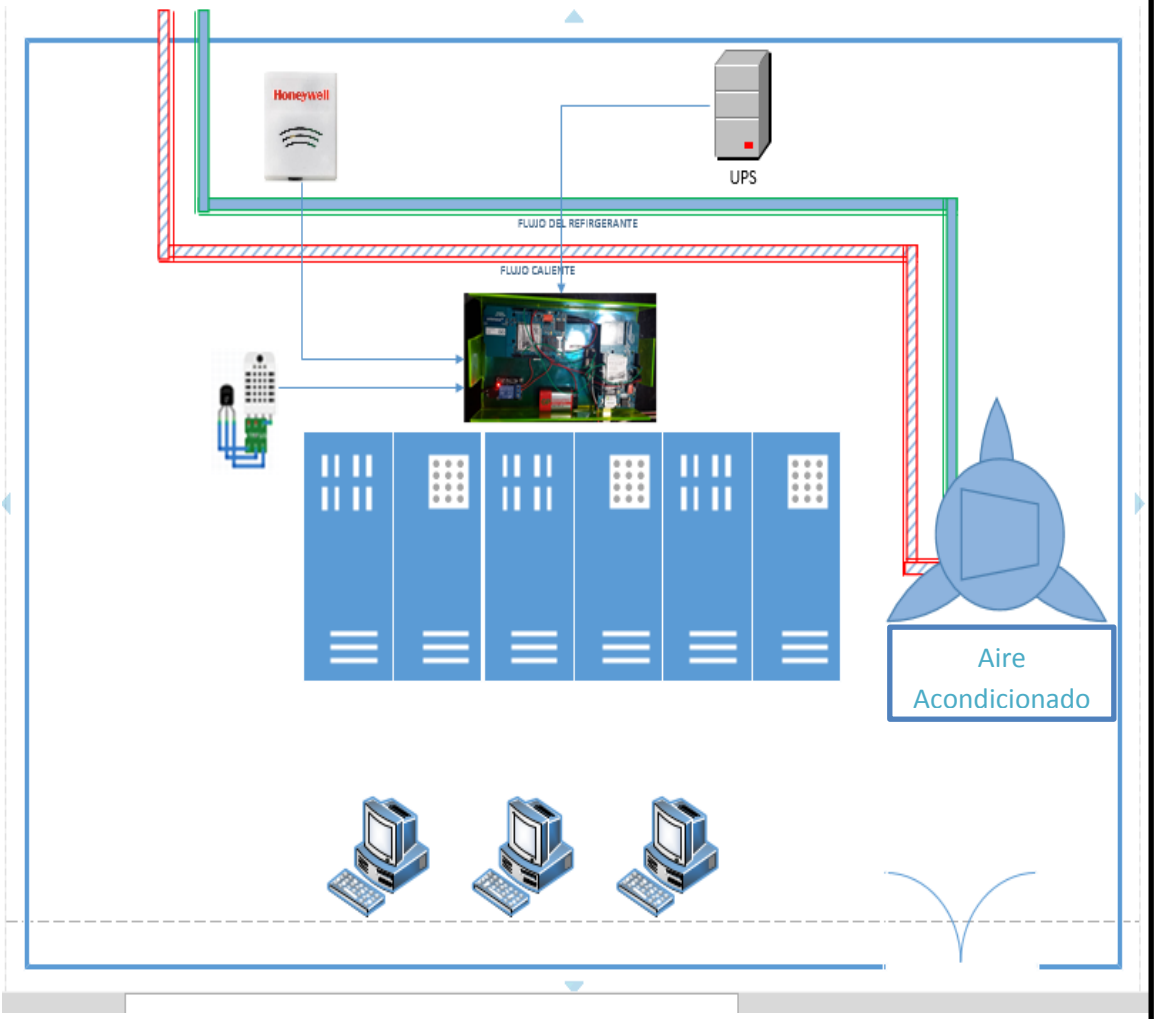


Figura 100.
Elaborado por: David Oña y Alexis Vaca

El primer paso es la ubicación módulo ArduinoUno este debe ir en una zona central, el Data Center de la EPMAPS consta de una zona de aire caliente y otra zona de aire frio. Como se observa en la figura 101 El modulo fue colocado en la parte central donde se trata de que la temperatura y humedad sea lo más exacta posible.

Ubicación MTI

Figura 101.
Elaborado por: David Oña y Alexis Vaca

Para colocar el sensor de fugas de agua es necesario levantar las divisiones del piso flotante y determinar dónde están las conexiones de las tuberías ya que es probable que en ese punto exista mayor riesgo de fugas, el cable se lo extiende junto a la tubería en la figura 102 Se muestra la ubicación del sensor.

Ubicación sensor de fugas de agua

Figura 102.
Elaborado por: David Oña y Alexis Vaca

Para el siguiente paso fue necesario ponchar un conector RJ45 hembra con la norma 568 B como se muestra en la figura 103 que van conectados hacia el modulo mediante cable UTP, estos jacks están instalados en el módulo ArduinoUno y permiten la movilidad de los sensores a lo largo del Data Center en caso de ser necesario.

Rj45 hembra

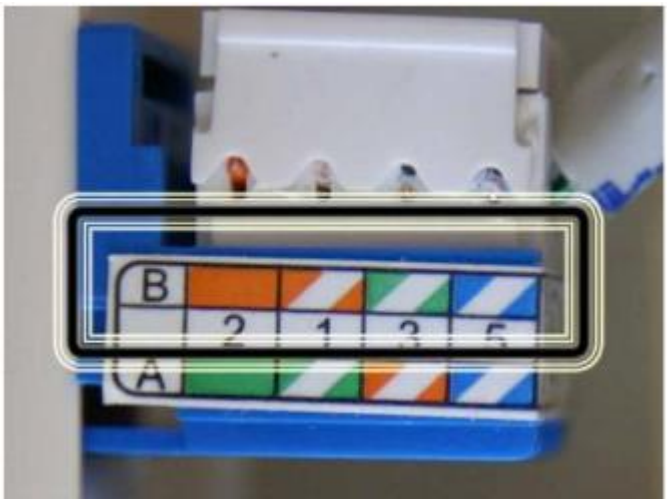


Figura 103.
Elaborado por: David Oña y Alexis Vaca

En la figura 104 se muestra un conector RJ45 ponchado de igual manera de la norma tipo B que irá conectado al módulo ArduinoUno y a los sensores de humedad y temperatura.

Cuyo código de colores de izquierda a derecha es:

- Naranja - Blanco
- Naranja
- Verde – Blanco
- Azul
- Azul – Blanco
- Verde
- Café – Blanco
- Café

Rj45

Figura 104.
Elaborado por: David Oña y Alexis Vaca

En la figura 105 se muestra como están conectados tanto el conector RJ45 hembra a los sensores de humedad y temperatura y como se debe conectar conector RJ45 para su correcto uso.

Conexión final

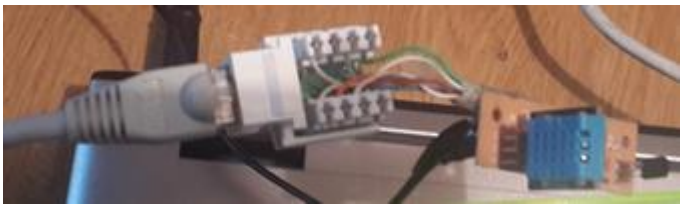


Figura 105.
Elaborado por: David Oña y Alexis Vaca

4.3. Implementación del servidor web

En la tabla 41 se detallan los programas necesarios para la implementación.

Tabla 41.
Programas

Java JDk (Java Development Kit)	El desarrollo de aplicaciones y applets de Java necesita herramientas de desarrollo como JDK. JDK incluye Java Runtime Environment, el compilador Java y las API de Java.
Apache Tomcat	Es una aplicación de software de código abierto de las tecnologías Java Servlet y JavaServer Pages. Las especificaciones Java Servlet y JavaServer Pages se desarrollan bajo Java Community Process .
PostgreSQL	Es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD (licencia de software libre permisiva) y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

Nota. Programas necesarios para la instalación del software MTI

Elaborado por: David Oña y Alexis Vaca

Las pruebas que se realizaron en el apartado 3.12 demuestran que esta implementación se realizó con éxito y actualmente se encuentra en producción.

CONCLUSIONES

Arduino es una plataforma en la que tenemos la posibilidad de interconectar todo de una manera flexible escalable, adaptable y totalmente controlada dirigidas a robótica, telecomunicaciones, automatización, etc.

El MTI logro proporcionar al Data Center de la EPMAPS. Una solución basada en el Monitoreo de Temperatura, Humedad Relativa, Estado de Carga del UPS, y envío de alertas por SMS y correo electrónico.

El envío de SMS y correos electrónicos del MTI depende únicamente de la cobertura de la operadoras móviles y un enlace de internet.

El módulo de monitoreo del UPS funciona perfectamente gracias a los algoritmos de lectura de los logs por medio de Java y el envío instantáneo de mensajes a Arduino GSMSheild gracias a las librerías que permiten el intercambio de información en tiempo real.

Una de las principales razones por la que se desarrolló la aplicación en Android para el monitoreo del Data Center, es que permita el acceso inmediato al sistema de monitoreo del MTI, desde cualquier celular o Tablet con sistema operativo Android 4.4.2 o superior permitiendo visualizar en tiempo real alertas y los diferentes estados de los sensores.

Debido a que los tiempos de respuesta son muy críticos en el monitoreo de las variables ambientales y carga de estado del UPS en un Data Center, se ha demostrado que el sensor análogo (LM35) de temperatura y el sensor digital de humedad (DTH11) son los más adecuados para la implementación del MTI como se observa en la tabla 26.

El módulo de backup es un elemento altamente importante en el MTI ya que entra en funcionamiento en menos de un minuto en el momento que se pierda el suministro de energía del UPS, enviando SMS alertando la falla total del sistema, permitiendo al personal de infraestructura y seguridad de tomar acciones urgentes. Ya que este sería el evento más crítico que pudiera suceder.

RECOMENDACIONES

Para futuros trabajos, se recomienda añadir nuevos sensores y un servidor web directamente en un módulo Arduino Ethernet Shield, permitiendo la autonomía del dispositivo.

Ejecutar el mantenimiento preventivo, de los módulos Arduino, con el propósito de precautelar la vida útil y el correcto funcionamiento de los mismos.

Se recomienda utilizar elementos electrónicos de precisión para implementar el circuito de sensores ya que los canales de voltaje y corriente trabajan con valores en el rango de milivoltios y cualquier interferencia puede generar valores errados de medición.

Se recomienda para el monitoreo de la temperatura el uso del sensor LM35 por su inmediata respuesta frente a cambios inesperados de temperatura.

Se recomienda el estudio de las tarjetas Arduino por su potencial de control de dispositivos electrónicos y por medio de sensores la recolección de datos del entorno, además de su programación que permite desarrollar un sin número de aplicaciones enfocada a todos los ámbitos de ingeniería.

LISTA DE REFERENCIAS

ABB. (2007, 01 01). *abb power and productivity for a better world.* Retrieved from abb power and productivity for a better world: http://www05.abb.com/global/scot/scot379.nsf/veritydisplay/914320f27fcd36e0c1257ddd003873db/$file/g8309_brochure_powerscale_es_150107.pdf

Adafruit. (2013, 1 1). *Adafruit*. Retrieved from Adafruit: http://www.adafruit.com/datasheets/DHT22.pdf

Agiles, P. (2015, 03 07). *Proyectos Agiles.* Retrieved from Proyectos Agiles: http://www.proyectosagiles.org/que-es-Scrum

Arduino. (2014, Enero 01). *Arduino.* Retrieved from Arduino: http://arduino.cc/en/Main/arduinoBoardUno

Cavsi. (2014, 01 01). *Cavsi.* Retrieved from Casi: http://www.cavsi.com/preguntasrespuestas/que-es-un-servidor-web/

Delfa, C. V. (2006, 01 01). *EL CORREO ELECTRÓNICO:*. Retrieved from EL CORREO ELECTRÓNICO:: http://biblioteca.ucm.es/tesis/fll/ucm-t29391.pdf

Espoch. (2013, 01 01). *Espoch.* Retrieved from Espoch: http://dspace.espoch.edu.ec/bitstream/123456789/1946/1/98T00016.pdf

Flores, O. (2013, Mayo 6). *academia.edu.* Retrieved from academia.edu: http://www.academia.edu/4340826/TESIS_BIO_CONTROL

Honeywell. (2014, 01 01). *Honeywell*. Retrieved from Honeywell: http://www.honeywellstore.com/store/products/honeywell-water-defense-leak-sensing-alarm.htm

HUIDOBRO MOYA, J. M. (2012). *COMUNICACIONES MÓVILES. SISTEMAS GSM, UMTS Y LTE.* RA-MA EDITORIAL.

Institute, U. (2013, 10 6). *Uptime Institute*. Retrieved from Uptime Institute: http://uptimeinstitute.com/images/stories/press_kits/UIPS_TiersSummary_100511.pdf

Julio Macías, M. S. (2002). Telefonistas. In M. S. Julio Macías, *telefonistas* (pp. 87,88,89). Sevilla: MAD, S.L.

Libre, E. A. (2014, 03 03). *El Androide Libre.* Retrieved from El Androide Libre: http://www.elandroidelibre.com/2014/05/linaro-anuncia-una-version-de-64-bits-para-dispositivos-arm-con-android-kitkat.html

LLamas, L. (2014, 04 16). *Ingeniería, informática y diseño.* Retrieved from Ingeniería, informática y diseño: http://www.luisllamas.es/2014/04/arduino-puerto-serie/

Moreno, E. G. (1999). *Automatización de procesos industriales.* Valencia: Servicio de Publicaciones.

ANEXOS

(Adjunto en CD)