

UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERÍA ELÉCTRICA



**Sistema de alerta temprana por riesgo de inundaciones
debidas a desbordamiento de los niveles de ríos.**

PRESENTADO POR:

JOSÉ RIGOBERTO OSEGUEDA MIRANDA

PARA OPTAR AL TÍTULO DE:

INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA, MARZO DE 2016

UNIVERSIDAD DE EL SALVADOR

RECTOR INTERINO :

LIC. JOSÉ LUIS ARGUETA ANTILLÓN

SECRETARIA GENERAL :

DRA. ANA LETICIA ZAVALA DE AMAYA

FACULTAD DE INGENIERÍA Y ARQUITECTURA

DECANO :

ING. FRANCISCO ANTONIO ALARCÓN SANDOVAL

SECRETARIO :

ING. JULIO ALBERTO PORTILLO

ESCUELA DE INGENIERÍA ELÉCTRICA

DIRECTOR :

ING. ARMANDO MARTÍNEZ CALDERÓN

UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERÍA ELÉCTRICA

Trabajo de Graduación previo a la opción al Grado de:

INGENIERO ELECTRICISTA

Título :

**Sistema de alerta temprana por riesgo de inundaciones
debidas a desbordamiento de los niveles de ríos.**

Presentado por :

JOSÉ RIGOBERTO OSEGUEDA MIRANDA

Trabajo de Graduación Aprobado por:

Docente Asesor :

MSc. e ING. JOSÉ WILBER CALDERÓN URRUTIA

San Salvador, marzo de 2016

Trabajo de Graduación Aprobado por:

Docente Asesor :

MSc. e ING. JOSÉ WILBER CALDERÓN URRUTIA

ACTA DE CONSTANCIA DE NOTA Y DEFENSA FINAL

En esta fecha, Jueves 10 de marzo de 2016, en la Sala de Lectura de la Escuela de Ingeniería Eléctrica, a las 8:00 a.m. horas, en presencia de las siguientes autoridades de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador:

1. Ing. Armando Martínez Calderón
Director

Firma: 

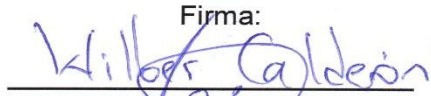


2. MSc. José Wilber Calderón Urrutia
Secretario

Firma: 

Y, con el Honorable Jurado de Evaluación integrado por las personas siguientes:

1- MSc. José Wilber Calderón Urrutia

Firma: 

2- MSc. Salvador de Jesús German



3- Ing. Walter Leopoldo Zelaya Chicas



Se efectuó la defensa final reglamentaria del Trabajo de Graduación:

Sistema de alerta temprana por riesgo de inundaciones debidas a desbordamiento de los niveles de ríos.

A cargo del Bachiller:

- José Rigoberto Osegueda Miranda

Habiendo obtenido en el presente Trabajo una nota promedio de la defensa final: 9.4

(NUEVE. CUATRO)

AGRADECIMIENTO

Agradezco a mi madre Marisa Miranda, por apoyarme en el camino hacia el éxito de la superación personal y profesional, a través de los años me brindo siempre su comprensión y el estímulo para alcanzar la meta.

Índice de contenido

Introducción	1
Objetivos	3
Alcances	4
Antecedentes	5
Justificación	6
Capítulo 1	7
1.1 Introducción a sistemas de alerta temprana	7
1.2 Definición de sistema de alerta temprana	7
1.3 Importancia de un sistema de alerta temprana	8
1.4 Aplicabilidad de un sistema de alerta temprana	8
1.4.1 Sistemas automatizados	9
1.4.2 Sistemas comunitarios	9
1.5 Dispositivos utilizados en un SAT	10
1.6 Elementos de un SAT	11
1.7 Funcionamiento de un SAT	12
Capítulo 2	16
2.1 Descripción del SAT	16
2.2 Funcionamiento del SAT	16
2.3 Raspberry Pi	18
2.3.1 Descripción de Raspberry Pi	19
2.3.2 Especificaciones técnicas	19
2.3.3 Sistemas operativos (elinux, s.f.)	21
2.3.4 Sistema operativo SAT	22
2.3.5 GPIOs	23
2.4 Pantalla táctil	25
2.4.1 Codificación	25
2.5 Sensor de temperatura - DS18B20	30

2.5.1 Descripción general	30
2.5.2 Características generales	30
2.5.3 Protocolo de comunicación	30
2.5.4 Valores máximos de operación	31
2.5.5 Configuración de pines	31
2.5.6 Características eléctricas	31
2.5.7 Codificación.....	32
2.6 Sensor de humedad DHT11	33
2.6.1 Características generales	33
2.6.2 Protocolo de comunicación	35
2.6.3 Configuración de pines	35
2.6.4 Características eléctricas	35
2.6.5 Codificación.....	36
2.7 Sensor de presión barométrica BMP180	37
2.7.1 Características generales	37
2.7.2 Protocolo de comunicación	37
2.7.3 Configuración de pines	38
2.7.4 Características eléctricas	38
2.7.5 Codificación.....	39
2.8 Sensor de distancia HC-SR04	40
2.8.1 Características generales	41
2.8.2 Protocolo de comunicación	42
2.8.3 Configuración de pines	42
2.8.4 Características eléctricas	42
2.8.5 Codificación.....	43
2.9 Sensor de precipitación WS 2300	45
2.9.1 Características generales	45
2.9.2 Protocolo de comunicación	46
2.9.3 Codificación.....	46
2.10 Placa 3G.....	47

2.10.1 Telefonía 3G	47
2.10.2 Descripción.....	48
2.10.3 Características módulo SIM5215E.....	48
2.10.4 Características módulo 3G.....	49
2.10.5 Protocolo de comunicación	50
2.10.6 Configuración de Pines	50
2.10.7 Características eléctricas	51
2.11 Cámara OV7670	52
2.11.1 Características generales	52
2.11.2 Características eléctricas	53
2.11.3 Descripción de pines	53
2.11.4 Codificación.....	54
2.12 Envío de alertas	56
2.12.1 Actualización de Sensores, GPS y estación SAT	56
2.12.2 Codificación ADC- modo medición de tensión.....	59
2.13 Sistema de posicionamiento global	60
2.13.1 Características.....	61
2.13.2 Especificaciones	61
2.13.3 Protocolo de comunicación	62
2.13.4 Codificación.....	63
2.13.5 NTPd	64
2.13.6 Configuración	65
2.14 Diseño PCB	66
2.15 Integración del sistema.	69
2.15.1 CRON	69
2.15.2 Formato del fichero crontab.....	69
2.15.3 Definición de horarios predefinidos.....	70
2.15.4 Configuración crontab	70
2.15.5 Interrelación de scripts	72
2.15.6 Diagrama de bloques del SAT.....	73

Capítulo 3	74
3.1 Ubuntu server	74
3.1.1 Introducción	74
3.1.2 Características Ubuntu Server	74
3.1.3 Requisitos Ubuntu Server	75
3.2 Django	75
3.2.1 Introducción	75
3.2.2 Descripción	76
3.2.3 Instalación	76
3.2.4 Aplicación en Django	77
3.2.5 Archivos de aplicación y configuración.	78
3.2.6 Configuración de Apache para uso con Django	95
3.3 MySQL server	97
3.3.1 Introducción	97
3.3.2 Características	97
3.3.3 Instalación	98
3.3.4 Creación de base de datos	99
3.4 HTML5	99
3.4.1 Características	99
3.5 Bootstrap	100
3.5.1 Características	100
3.6 CSS3	100
3.7 Google Maps	100
3.7.1 Codificación	101
Capítulo 4	106
4.1 Sitio WEB	106
4.2 Alertas enviadas por la estación	123
4.2.1 Tweets	123
4.2.2 Mensajes de texto	124
Conclusión	126

Bibliografía	127
Anexos	129
Anexo A: Cuadro desagregado de precios	129
Anexo B: REQUESTS	130
Anexo C Wvdial	131
Anexo D Protocolo de comunicación SPI	132
Anexo E Protocolo de comunicación I2C	133
Anexo F Protocolo de comunicación UART	135
Anexo G Sentencias NMEA	136
Anexo H comandos AT	136

Índice de ilustraciones

Ilustración 1. Raspberry Pi modelo B	19
Ilustración 2 Raspberry Pi, headers P1 y P2 (Kleback, s.f.).....	24
Ilustración 3. Pantalla táctil (Adafruit, s.f.).....	25
Ilustración 4. DS18B20	31
Ilustración 5. DHT11	35
Ilustración 6. BMP180	38
Ilustración 7. HC-SR04, Rango efectivo de detección (Electfreaks)	41
Ilustración 8. HC-SR04	42
Ilustración 9. WS2300 (Sparkfun).....	45
Ilustración 10. Módulo de comunicación 3G	49
Ilustración 11. Módulo 3G, descripción de pines.....	50
Ilustración 12. OV7670, cámara digital	52
Ilustración 13. GPS- neo 6m	61
Ilustración 14. PCB base estación SAT - lado de componentes	66
Ilustración 15. PCB base estación SAT - lado de pistas	67
Ilustración 16. Derecha driver de motor, izquierda fuente de alimentación	68
Ilustración 17. Circuito de acople convertidor ADC	68
Ilustración 18. Interrelación de scripts del sistema	72
Ilustración 19. Diagrama de bloques estación SAT	73
Ilustración 20. Página de home (inicio)	106
Ilustración 21. Página de registro de usuarios	107
Ilustración 22. Correo de confirmación de registro	107

Ilustración 23. Página de inicio de sesión	108
Ilustración 24. Página de recuperación de cuenta	109
Ilustración 25. Correo de recuperación de cuenta.....	109
Ilustración 26. Página de usuario registrado.....	110
Ilustración 27. Condiciones del SAT	111
Ilustración 28. Condiciones del servidor	112
Ilustración 29. página de información de dispositivos.....	113
Ilustración 30. Página de gráficas y estadísticas	114
Ilustración 31. Tacómetro de registro	115
Ilustración 32. Cuadro de promedio, máximos y mínimos	116
Ilustración 33. Página de teclas	116
Ilustración 34. Mapa de riesgo.....	117
Ilustración 35. Página de calendario de actividades	118
Ilustración 36. Página de imágenes.....	119
Ilustración 37. Página de videos.....	120
Ilustración 38. Página de información.....	121
Ilustración 39. Página de administracion del sitio.....	122
Ilustración 40. Alertas emitidas en red social Twitter.....	123
Ilustración 41. Alertas emitidas por mensaje de texto	124
Ilustración 42. Mensaje informativo	125

Índice de tablas

Tabla 1 Raspberry Pi, especificaciones técnicas. (elinux, s.f.).....	20
Tabla 2. DS18B20, valores máximos de operación (Maxim Integrated)	31
Tabla 3. DS18B20 Características eléctricas (Maxim Integrated)	32
Tabla 4. DHT11, características sensor de humedad (D-Robotics)	34
Tabla 5. DHT11 características sensor de temperatura (D-Robotics)	34
Tabla 6. DHT11 Características eléctricas (D-Robotics)	35
Tabla 7. BMP180, características eléctricas (Bosch Sensortec)	39
Tabla 8. HC-SR04, características de sensor (Electfreaks).....	41
Tabla 9. HC-SR04, Características eléctricas (Electfreaks)	42
Tabla 10. Módulo 3G, indicadores led (Electfreaks, s.f.).....	51
Tabla 11. Módulo 3G, características eléctricas (Electfreaks, s.f.).....	51
Tabla 12 OV7670, características eléctricas (Omnivision)	53
Tabla 13. OV7670, descripción de pines (Omnivision).....	53
Tabla 14. OV7670, Funcion de pines (Omnivision)	54
Tabla 15. GPS, Especificaciones técnicas (Ublox).....	62

Tabla 16. Crontab, horarios predefinidos (Ubuntu documentation, s.f.) 70
Tabla 17 Cuadro de precios..... 129

Índice de scripts

Script 1. Interfaz de pantalla de control..... 29
Script 2. Lectura sensor de temperatura 33
Script 3. Lectura sensor de humedad..... 36
Script 4. Lectura sensor de presión 40
Script 5. Lectura sensor de distancia..... 43
Script 6. Lectura de sensor de precipitación 46
Script 7. Captura fotográfica 55
Script 8. Captura de video 56
Script 9. Actualización al servidor 59
Script 10. Lectura de tensión en batería 60
Script 11. Lectura del GPS 64
script 12. ntp.conf 65
script 13. rc.local 65
Script 14. Formato crontab 69
Script 15. Archivo crontab..... 71
script 16. Archivo de configuración de Django 81
script 17. Modelos estación SAT 83
script 18. Definición de vistas..... 86
script 19. Plantilla de página home 92
script 20. Definición de URLs del sitio 94
script 21. apache2.conf 95
script 22. Configuración mapa de riesgo..... 104

Introducción

El sistema de alerta temprana es una estación que está provista de un conjunto de dispositivos electrónicos relacionados de tal forma que permiten la toma de decisiones sobre la base de las condiciones medioambientales registradas por dichos dispositivos.

La aplicación del sistema de alerta temprana tiene lugar en el entorno donde existe riesgo a la problemática sobre el desbordamiento de ríos, donde los habitantes de la región en cuestión se ven amenazados y es necesario dar solución en el corto plazo a este problema.

El problema del desbordamiento de los niveles de los ríos en El Salvador, es un hecho que ha dejado la pérdida de vidas humanas y daños materiales, es por esto que la implementación del sistema con las capacidades que la tecnología actual permite es de vital importancia para la prevención de este tipo de desastres.

El sistema de alerta temprana (SAT) consiste en un dispositivo con la capacidad de monitoreo en tiempo real de las condiciones medioambientales tales como la temperatura, presión barométrica, humedad relativa, cantidad de precipitación por metro cuadrado, y los niveles de altura del río en análisis y generar alertas específicas e inmediatas a cada habitante en riesgo, utilizando la red de telefonía 3G y las redes sociales. Además permite el monitoreo remoto en línea (web) mediante un servidor dedicado que almacena una base de datos que se actualizan constantemente y que sirven para generar estadísticas y registros que permitirían a las autoridades de gobierno prevenir con anticipación los desastres por desbordamiento de ríos observando las tendencias en los datos históricos generados.

El método de creación del SAT se basó en la programación de dispositivos de integración a gran escala ARMv11, que son la tendencia actual en el desarrollo de tecnología, el framework moderno Django para la creación de la página web y el sistema operativo Linux Ubuntu server 12.04 como servidor dedicado.

De esta forma la creación de un sistema de alerta temprana específico y moderno es importante porque permite a las autoridades la prevención de desastres por desbordamientos de ríos evitando así la pérdida de vidas humanas y aquellas pérdidas materiales subsecuentes.

Objetivos

General

1. Desarrollar un sistema electrónico con comunicación inalámbrica moderna de alerta temprana, para la prevención de pérdida de vidas humanas antes que los ríos o quebradas presenten niveles de agua peligrosos.

Específicos

1. Construir un sistema electrónico basado en software y hardware de libre distribución para el monitoreo de niveles de agua de ríos o quebradas, así también georeferenciando las condiciones medioambientales de la zona.
2. Utilizar la red de telefonía GSM/3G de El Salvador para el envío de alertas hacia terminales móviles o centrales de procesamiento de datos para generar un histórico de los eventos registrados.
3. Registrar las mediciones del nivel del agua de los ríos o quebradas para mantener informada a la población y autoridades en tiempo real, vía mensajes de texto, llamadas telefónicas y vía red social.
4. Realizar una página web en donde se pueda consultar en tiempo real la actividad de la estación de alerta temprana, así como poder descargar la base de datos históricos generados.

Alcances

Se busca realizar una estación de motorización telemétrica en tiempo real de los niveles del agua de ríos o quebradas, utilizando la red de telefonía actual proporcionada por los operadores existentes en el país y con una cobertura de señal ofertada por dichas compañías.

La estación reportará datos recopilados por los sensores de temperatura, presión barométrica, humedad relativa del aire, precipitación y aumento del nivel agua en ríos o en quebradas. Estos datos serán enviados utilizando los protocolos de comunicación TCP/UDP a servidores de bases datos. Luego del procesamiento de los datos la información será mostrada públicamente en una página web.

También posibilita el envío de mensajes de texto a terminales móviles o fijas en caso de presentarse condiciones de riesgo a través de alertas mediante el uso de una red social para aquellos usuarios en condición de riesgo que sigan dicha cuenta.

La administración de la estación de alerta temprana se establece a través de mensajes de texto de usuarios denominados como administradores del sistema y conexión vía servidor TCP/UDP a través de una página web propia de la estación de alerta temprana que se actualiza en tiempo real.

Antecedentes

El Salvador es un país que debido a su localización geográfica en el mundo presenta un clima tropical, está sujeto a condiciones de inundaciones provocadas por el desarrollo de huracanes y depresiones tropicales y las experiencias vividas en los recientes años demuestran que el país es susceptible a desastres debido a desbordamientos de ríos tales como el suscitado en la capital en la zona de la Málaga, entre otros.

Según el documento “SISTEMA DE ALERTA TEMPRANA POR INUNDACIONES EXPERIENCIA EN EL SALVADOR – SNET” nuestro país contaba con una cierta cantidad de estaciones de monitoreo convencionales de las precipitaciones pero luego de la guerra civil sufrió un deterioro de las estaciones de tal manera que el número total de las estaciones disminuyó reduciéndose drásticamente el sistema de monitoreo de las condiciones hidrometeorológicas del país.

Actualmente el país cuenta con una red hidrometeorológica básica de 28 estaciones: 16 estaciones con telemetría, 10 estaciones automáticas y 2 convencionales.

Justificación

En los países en vías de desarrollo, el crecimiento demográfico conlleva a la población de escasos recursos a migrar y a buscar sitios para desarrollar asentamientos en zonas de alta vulnerabilidad tales como las riberas de los ríos o quebradas, los barrancos y las faldas de volcanes. Por lo general, esta población no cuenta con información a su alcance para tomar las decisiones más adecuadas sobre las zonas geográficas en las cuales podrían asentarse sin ningún riesgo y por esto habitan en zonas en donde las inundaciones son frecuentes. Debido a esto, utilizar un sistema de alerta temprana podría resolver el problema de pérdidas de vidas humanas susceptibles a riesgo.

En base a que el país cuenta con una institución encargada del servicio nacional de estudio territoriales para la gestión de riesgos (SNET) y una red de estaciones telemétricas, automáticas, y convencionales, es decir El Salvador cuenta con todo un mecanismo para la operación y determinación de riesgos. Es posible la introducción de nuevas estaciones telemétricas de monitorización en tiempo real que sirvan de apoyo a la red actual y así poder ampliar el sistema de alerta temprana de El Salvador en la prevención de desastres por inundaciones. Es importante también la utilización de un sistema de alerta temprana vinculada a una red social, ya que en la actualidad existen muchas personas que cuentan con dispositivos móviles capaces de acceder a todo tipo de información que se suba a la red social. Por lo que la población en posible riesgo estará informada en tiempo real. En caso que la población sujeta a riesgo no tiene al alcance dicha tecnología en sus terminales móviles, la misma estación de alerta temprana envía mensajes de texto a celulares registrados, con esto la población no incurre en gastos.

Capítulo 1

Sistemas de Alerta Temprana (UNESCO - CEPREDENAC, 2012)

1.1 Introducción a sistemas de alerta temprana

La alerta temprana es uno de los principales elementos de la reducción de desastres. Los sistemas de alerta temprana evitan la pérdida de vidas y disminuye los impactos económicos sociales y materiales. Para que un sistema de alerta temprana sea eficaz, debe incluir activamente a las comunidades en riesgo, facilitar la educación sobre el uso y la concientización del público sobre tales riesgos, diseminar eficazmente mensajes y alertas en tiempo real y garantizar una preparación constante de las partes involucradas. Los sistemas de alerta temprana pueden brindar información para tratar de anticipar los eventos naturales que, en interacción con la vulnerabilidad, pueden desembocar en desastres.

1.2 Definición de sistema de alerta temprana

Un Sistema de Alerta Temprana¹ es aquel dispositivo complejo que alerta con antelación de la eventualidad de un acontecimiento natural o humano que puede causar un desastre, con el objetivo de evitarlo. Desde la terminología de la gestión de riesgos, la dimensión del desastre está en la función de la fuerza del evento natural y del nivel de vulnerabilidad de la población ante el mismo. Los Sistemas de Alerta Temprana recolectan y procesan datos e información, ofreciendo pronóstico o predicciones temporales sobre su acción y posibles efectos. Millones de personas en todo el mundo salvan sus vidas y sus medios de subsistencia gracias a la implementación de estos sistemas

¹ SAT

1.3 Importancia de un sistema de alerta temprana

La importancia de un SAT, radica en el hecho de permitir conocer anticipadamente y con cierto nivel de certeza, en que tiempo y espacio, una amenaza o evento adverso de tipo natural puede desencadenar situaciones potencialmente peligrosas. Por lo cual las alertas deben difundirse con suficiente anticipación es decir justo en el momento en que las condiciones sean potencialmente peligrosas.

El objetivo fundamental de un SAT es por lo tanto, evitar la posibilidad que se produzcan lesiones personales, pérdidas de vidas, daños a los bienes, mediante la emisión de alertas y al pronta aplicación de medidas de protección y reducción de riesgos. Los planes de gestión de riesgo o respuesta implementados en coordinación con los sistemas de alerta temprana son medidas indispensables para que una alerta sea efectiva.

1.4 Aplicabilidad de un sistema de alerta temprana.

Dado que los sistemas de alerta temprana incluyen sensores que monitorizan las condiciones medioambientales de riesgo. La aplicabilidad se da en lugares en donde se pueda vigilar, monitorizar y emitir alertas. Entre las amenazas o eventos más comunes a los cuales se aplican SAT se tienen las inundaciones, deslizamientos de tierra, huracanes, volcanes, tsunamis, incendios forestales, fenómeno del niño y la niña, entre otros.

En los Sistemas de Alerta Temprana para inundaciones, se puede identificar dos modalidades: los automatizados y los operados por las comunidades.

1.4.1 Sistemas automatizados

Los sistemas automatizados se basan en la observación y monitoreo mediante la utilización de redes telemétricas, estaciones de lluvia y niveles de los ríos, modelos hidrológicos computarizados, sensores remotos, y satélites; con lo cual se vigila la cantidad de lluvia, los niveles de los ríos, para pronosticar crecidas en forma precisa.

Estos sistemas tienen aplicación en cuencas hidrográficas grandes y se apoyan en organizaciones de tipo técnico-científico como los Centros Especializados en Hidrometeorología, Universidades, Sistema Nacional de Protección Civil, Gobiernos Locales y otros actores sociales.

1.4.2 Sistemas comunitarios

Estos sistemas tienen aplicación en cuencas hidrográficas medianas y pequeñas; son de fácil manejo, ya que sus instrumentos son básicos y no requieren de técnicos especializados; los recursos disponibles para su creación y funcionamiento son limitados; participan un conjunto de actores, en donde la comunidad organizada es el elemento fundamental, y cuya participación se ejerce en forma voluntaria. Con estos sistemas las comunidades identifican sus riesgos, aumentan sus capacidades para enfrentar emergencias y reducen la posibilidad de pérdidas de vidas y daños materiales. Por ello es indispensable su activa participación, en todos los aspectos del establecimiento y funcionamiento de los SAT, sean estos de tipo automatizado o comunitario, ya que ambos sistemas aportan y contribuyen al fortalecimiento de los procesos de desarrollo de las comunidades donde son implementados.

1.5 Dispositivos utilizados en un SAT

La utilización de dispositivos y sensores para un Sistema de Alerta Temprana depende de las características particulares de los eventos o amenazas, de su ubicación geográfica, y de los recursos disponibles.

En caso de sistemas automatizados se utilizan instrumentos sofisticados o tecnológicos como satélites, sensores remotos, redes telemétricas y otros que permiten transmitir información directa desde los equipos de medición hasta los centros de análisis y de toma de decisión. En cuanto a sistemas comunitarios se utilizan equipos de bajo costo y de fácil manejo. En los Sistemas de Alerta Temprana para inundaciones, se mide la cantidad de lluvia precipitada y el nivel de caudal de los ríos, para ello se utilizan dos instrumentos fundamentales

1. La medición de la cantidad de lluvia precipitada se mide mediante un instrumento llamado “PLUVIMETRO”, estos son recipientes, en algunos casos graduados, que permiten medir la cantidad de agua que cae durante un tiempo determinado, pueden ser automatizados o manuales. En los sistemas comunitarios, los voluntarios se encargan de la lectura, registro y transmisión de los datos obtenidos en estos instrumentos de medición.
2. La medición de los niveles de los ríos, también se puede efectuar mediante la utilización de instrumentos automatizados con sensores ubicados en tubos que se colocan en zonas donde se pueda determinar los cambios de nivel de agua, la información se registra y es procesada automática y directamente. También se utilizan las “REGLAS LIMNIMÉTRICAS”, su uso es generalizado por su bajo costo y fácil manejo, no requiere de personal especializado, sólo de una comunidad organizada y comprometida con su propia seguridad, este instrumento consiste en colocar dentro o fuera de los ríos, postes o reglas graduadas en centímetros, y pintadas en tres colores relacionados a las alertas (verde, amarillo y rojo); como alternativa se pueden pintar y graduar postes de las bases de puertos o embarcaderos, puentes, árboles, piedras, pisos u otros elementos del entorno que sirvan como regla y permita realizar una vigilancia adecuada de los cambios en los niveles de los ríos. Al igual que los pluviómetros, cuando la comunidad participa, los voluntarios se encargan de la lectura, registro y transmisión de los datos obtenidos en estas reglas.

1.6 Elementos de un SAT

Para la implementación de un Sistema de Alerta Temprana, se debe tener en cuenta una serie de elementos y de estructuras sectoriales e institucionales, además otros componentes que determinan su aplicación y éxito. Algunas organizaciones internacionales identifican cuatro elementos fundamentales que deben ser tomados en cuenta para la creación de un SAT:

1. **Existencia y conocimiento del riesgo:** Se debe identificar las amenazas y tener conocimiento de los riesgos, o eventos potencialmente peligrosos que puedan afectar a las poblaciones, infraestructuras y recursos expuestos al impacto de dichos fenómenos. Esto debe estar plasmado en un Mapa de Riesgo, ya que conociendo las amenazas, vulnerabilidades y los elementos expuestos a dichos fenómenos, podremos estimar la potencialidad del peligro y los daños que se puedan generar, para tomar medidas de Gestión de Riesgo como los Sistemas de Alerta Temprana.
2. **Respaldo técnico e institucional:** Se debe contar con el respaldo de instituciones científico- técnicas, y aquellas responsables de la Gestión del Riesgo a Desastres, para que el estudio, vigilancia, seguimiento y evaluación de una amenaza o evento adverso contenga una base científica. Es necesaria la participación de las autoridades locales e instituciones nacionales, que componen el Sistema Nacional de Protección Civil, las cuales tienen la responsabilidad de establecer operaciones y acciones relacionadas con la preparación y la respuesta en caso de materializarse dichos eventos. En el proceso de creación de un SAT, se debe contar con los recursos necesarios: técnicos, financieros y humanos.
3. **Difusión y comunicación:** Es clave la comunicación y la difusión de información, para motivar y concienciar a los habitantes de las comunidades y a sus autoridades locales, sobre la importancia del conocimiento de los riesgos, amenazas, vulnerabilida-

des, planes de emergencias y medidas de prevención y reducción de riesgos a desastres, como el Sistemas de Alerta Temprana, que incluye la transmisión de datos, emisión de alertas, alarmas y la coordinación de comunicaciones en situaciones de emergencia.

4. **Capacidad de respuesta:** Es necesario contar con la participación directa de las comunidades, las cuales deben estar organizadas y preparadas con sus Planes de Respuesta debidamente actualizados, para actuar en caso de emergencias. Los SAT forman parte de la preparación y aportan información para la toma de decisiones en materia de gestión del riesgo y el desarrollo de las comunidades. Esta preparación local requiere del apoyo y coordinación con entidades nacionales para una mayor efectividad de la respuesta y de las acciones integrales de reducción de riesgo a desastres.

1.7 Funcionamiento de un SAT

El funcionamiento consiste en la ejecución de los siguientes pasos: lectura y registro de la medición de los instrumentos sobre el evento monitoreado; transmisión de los datos registrados; procesamiento y análisis de los datos transmitidos; pronóstico de la situación; establecimiento del nivel y tipo de alerta; difusión del nivel de alerta; activación de un Plan de Emergencias o Evacuación

1. **Lectura y Registro:** Cuando el fenómeno monitoreado produce alguna alteración, activación o manifestación de peligro, es registrado por los instrumentos, y se procede a tomar las lecturas correspondientes, manteniendo una vigilancia pormenorizada, continua y permanente para conocer sus cambios y evolución. Estas lecturas pueden realizarse con equipos tecnológicos, simples o manuales, operados por instituciones especializadas, por las comunidades o entre ambos.

2. **Transmisión de Datos:** Luego que las lecturas han sido tomadas y registradas, son transmitidas inmediatamente, para que los encargados o especialistas efectúen los cálculos necesarios y se realicen los pronósticos respectivos, sobre la posible ocurrencia o no de un evento adverso o destructivo. Si se trata de una institución especializada, la transmisión puede hacerse automáticamente mediante equipos sofisticados o tecnológicos como satélites, teléfonos móviles, sistemas computarizados, etc.; si es un sistema comunitario, se utilizarán los equipos que estén a su alcance, y se ajusten a las condiciones, presupuesto o cultura. En la mayoría de los casos se utilizan sistemas de radiocomunicación, radio-emisoras, teléfonos o cualquier otro medio que permita en forma segura y rápida enviar los datos.
3. **Procesamiento y Análisis de Datos:** Los datos llegan a manos de expertos o encargados de procesarlos, quienes realizan sus cálculos y establecen si estos indican la posibilidad o no de manifestarse un evento adverso o destructivo. Los datos pueden ser analizados automáticamente con la utilización de equipos tecnológicos, como sistemas computarizados que realizan pronósticos.

Si estos son operados por la comunidad, deben llegar a las personas responsables de los comités locales, quienes procesaran la información, para que las autoridades analicen la situación y definan el nivel y tipo de alerta a declarar.

4. **Evaluación de la Situación y Definición de la Alerta:** Las instituciones encargadas o los miembros de los comités de Emergencias de las comunidades, evalúan la información o el resultado del análisis de los datos procesados y lo contrastan con un Mapa de Riesgo, determinando así el daño potencial, nivel y tipo de alerta que se debe declarar y emitir. Comúnmente se utilizan tres colores de alertas, en algunos países se utilizan cuatro, incorporando el color anaranjado, cada una con un significado y acciones definidas.

- Verde: indica que se debe estar atento al comportamiento y evolución del fenómeno o evento monitoreado, y de las alertas que se continúen emitiendo. Esta alerta debe dirigirse a los especialistas de las instituciones, los encargados del Plan de Emergencia y los habitantes de las comunidades en peligro
 - Amarilla: aumenta la alerta y los diferentes equipos e instituciones inician sus preparativos para ejecutar las acciones correspondientes, dirigidas a enfrentar el impacto del evento y sus consecuencias
 - Roja: significa que es inminente la llegada o materialización del evento, esta alerta es emitida a través de las instituciones responsables o entidades autorizadas, tanto nacionales como locales. Se activa el Plan de Emergencias y, en la mayoría de los casos, se ordenará la evacuación de los pobladores a zonas seguras o albergues, además otras acciones, según las condiciones en que se presenta el evento.
 - En algunos países se utilizan cuatro colores, en el cual el color naranja se sitúa después del color amarillo, representando la Alerta Naranja, que significa lo siguiente: Cuando se han concretado las condiciones necesarias para que se presente el fenómeno y sólo sea cuestión de minutos y horas para que se manifieste el fenómeno.
5. ***Difusión de la Alerta:*** Al contar con la alerta oficial debidamente definida, emitida y comprobada, se procede a notificarla a la población. La alerta debe ser clara y oportuna, garantizando la confianza de las comunidades o beneficiarios. La alerta se podrá difundir utilizando radios de comunicación, radio emisoras, teléfonos, radio parlantes, sirenas, banderas, campanas y cualquier otro instrumento que tenga el alcance, que permita informar rápidamente a la comunidad.

6. ***Activando el Plan de Emergencias o Respuesta:*** Sin este paso la alerta, no tendría sentido o ningún resultado, por lo tanto es imprescindible que todos los centros educativos y las comunidades cuenten con planes o actividades de preparación para respuesta.

Capítulo 2

Estación de alerta temprana

2.1 Descripción del SAT

La estación de alerta temprana es un sistema integrado que cuenta con 4 sensores que registran las condiciones medioambientales de temperatura, humedad², precipitación por lluvia, presión barométrica³. Un sensor que monitorea el nivel de altura del río. Un dispositivo GPS⁴ que sirve para referenciar geográficamente la estación de alerta temprana, un dispositivo de comunicación inalámbrica de tercera generación. Una cámara digital para visualizar las condiciones de la región en riesgo y una pantalla táctil para interacción con el operador. Todo controlado por una placa microprocesador. Además cuenta con una batería que sirve para la alimentación de la estación de alerta temprana y un voltímetro que periódicamente registra la tensión en la batería.

2.2 Funcionamiento del SAT

La estación de alerta temprana en su funcionamiento normal inicia arrancando su sistema operativo Raspbian y configurando todos los dispositivos con los que cuenta. Activa el GPS y luego se registra en la red de telefonía y comienza a realizar el monitoreo de las condiciones medioambientales que le permiten los sensores.

En caso de detectarse una condición de riesgo reportada por el sensor de nivel del río, la estación de alerta temprana establece la condición de riesgo y emite alertas por

² **Humedad relativa:** Relación entre la cantidad de vapor de agua que tiene una masa de aire y la máxima que podría tener.

³ **Presión barométrica:** Presión ejercida por la atmósfera en una altitud dada.

⁴ **GPS:** Sistema de posicionamiento global.

llamadas telefónicas y mensajes de texto a aquellas personas que han sido registradas en el directorio telefónico interno de la estación de alerta temprana. Al actualizar hacia el sistema de almacenamiento de datos y monitoreo web este último emite una publicación en la red social de Twitter sobre las condiciones de alerta registradas por la estación de alerta temprana.

El registro del nivel del río y los demás sensores se realiza cada minuto, las actualizaciones de nivel, temperatura, humedad, presión y precipitación se envían al servidor cada minuto para tener información actualizada de las condiciones registradas por la estación de alerta temprana, a excepción de la cámara digital que está programada para tomar fotografías del río a intervalos de 15 minutos y tomar videos de duración de 1 minuto para ser almacenados localmente y retirados cuando se realice una labor de mantenimiento.

Existen también registro y actualización de las condiciones propias de la estación de alerta temprana, se reporta la temperatura del procesador, la frecuencia del reloj, la carga de la CPU y el porcentaje de utilización de la memoria RAM.

La estación de alerta temprana también soporta el monitoreo por mensajes de texto a terminales móviles registrados en el directorio de la estación. Cada vez que la estación de alerta temprana recibe un mensaje de texto enviado por algún usuario registrado de la estación de alerta temprana este recibe un mensaje en un lapso de 5 minutos informándole sobre las condiciones que ha solicitado conocer. La estación de alerta temprana admite mensajes de texto que incluyen la palabra “temperatura”, “humedad”, “precipitación”, “nivel”, “presión”, “alerta” como único cuerpo del mensaje; En caso contrario indica que se desconoce la variable que ha solicitado conocer. Si alguna persona externa al directorio telefónico de la estación de alerta temprana requiere de información esta no será enviada dado que no se encuentra en la base de datos de la estación.

Mediante la interfaz que proporciona la pantalla táctil se puede ver las condiciones registradas por la estación de alerta temprana, la ubicación reportada por el GPS y es posible también el reiniciar la estación de alerta temprana o apagar por completo para realizar alguna labor de mantenimiento. Es posible también realizar llamadas de corta duración aproximadamente de 1 minuto.

A continuación se describe el hardware utilizado en la implantación de la estación de alerta temprana.

2.3 Raspberry Pi

El sistema de alerta temprana utiliza como unidad de procesamiento una placa de desarrollo de última tecnología que abarca la utilización de microprocesadores ARM en una integración a gran escala, diseñada para su utilización en sistemas embebidos⁵. La placa denominada Raspberry Pi fue desarrollada en Reino Unido por la Fundación Raspberry Pi,

Dentro de la evolución de la fundación se han ido desarrollando varios modelos de placas de desarrollo, actualmente existen 4 tipos: modelo A, modelo B y modelo B+ y el modelo 2 B. La estación de alerta temprana utiliza el modelo B que este cumple con las especificaciones de hardware necesarias para la implementación de la estación.

⁵ **Sistema embebido:** Sistema de computación diseñado para realizar algunas funciones dedicadas, en un sistema en tiempo real.

2.3.1 Descripción de Raspberry Pi



ILUSTRACIÓN 1. RASPBERRY PI MODELO B

El diseño incluye un SOC⁶ Broadcom BCM 2835, que contiene un procesador central (CPU) ARM1176JZF-S a 700 MHz el firmware incluye modos Turbo para hacer overclock de hasta 1 GHz, un procesador gráfico (GPU) Video core IV, y 512 MB de memoria RAM. El diseño no incluye un disco duro ni unidad de estado sólido, ya que usa una tarjeta SD para el almacenamiento permanente, tampoco incluye fuente de alimentación ni carcasa.

2.3.2 Especificaciones técnicas

Las especificaciones de los modelos varían en el hardware de soporte a excepción de la última versión, que ha experimentado un cambio de microprocesador.

⁶ **SOC**: System on chip. Un solo chip contiene, CPU, GPU y RAM.

	Modelo A	Modelo B	Modelo B+	Modelo 2 B
SoC:	Broadcom BCM2835 (CPU + GPU + DSP + SDRAM + puerto USB)			Broadcom BCM2836 (CPU + GPU + DSP + SDRAM + puerto USB)
CPU:	ARM 1176JZF-S a 700 MHz			900 MHz quad-core ARM Cortex A7
Juego de instrucciones:	RISC de 32 bits			
GPU:	Broadcom VideoCore IV, 60 OpenGL ES 2.0, MPEG-2 y VC-1 (con licencia), 58 1080p30 H.264/MPEG-4 AVC3			
Memoria (SDRAM):	256 MiB (compartidos con la GPU)	512 MiB (compartidos con la GPU)		1 GB (compartidos con la GPU)
Puertos USB 2.0:	1	2 (vía hub USB integrado) ⁵³	4	
Entradas de vídeo:	Conector MIPI CSI que permite instalar un módulo de cámara desarrollado por la RPF			
Salidas de vídeo:	Conector RCA (PAL y NTSC), HDMI (rev1.3 y 1.4), Interfaz DSI para panel LCD ⁶³			
Salidas de audio:	Conector de 3.5 mm, HDMI			
Almacenamiento integrado:	SD/MMC/ ranura para SDIO		MicroSD	
Conectividad de red:	Ninguna	10/100 Ethernet(RJ-45) via hub USB		
Periféricos de bajo nivel:	8xGPIO, SPI, I ² C, UART			17 x GPIO y un bus HAT ID
Consumo energético:	500mA, (2.5W)	700mA, (3.5 W)	600 mA, (3.0 W)	800 mA, (4.0 W)
Fuente de alimentación:	5V vía Micro USB / GPIO header			
Dimensiones:	85.60mm × 53.98mm ⁶⁵ (3.370 × 2.125 inch)			
Sistemas soportados:	GNU/Linux: Raspbian, Pidora, Arch Linux ARM, Slackware Linux. RISC OS			

TABLA 1 RASPBERRY PI, ESPECIFICACIONES TÉCNICAS. (ELINUX, S.F.)

2.3.3 Sistemas operativos (elinux, s.f.)

La Raspberry Pi usa sistemas operativos basados en el núcleo Linux. Por ejemplo Raspbian, una distribución derivada de Debian que está optimizada para el hardware de Raspberry Pi. Slackware ARM. Los 512 MiB de memoria RAM disponible en la Raspberry Pi, cubren los necesarios 64 MiB de RAM para arrancar estas distribuciones en sistemas ARM⁷

La lista de sistemas operativos que funcionan, se han portado, o están en proceso de ser portados a Raspberry Pi:

Sistemas operativos completos:

- AROS
- Linux
- Android98
- Arch Linux ARM
- Debian Wheezy Soft-Float, versión de Debian sin soporte para coma flotante por hardware
- Firefox OS
- Gentoo Linux99
- Google Chromium OS
- Kali Linux
- Open webOS100
- PiBang Linux,101 distribución Linux derivada de Raspbian con diferente escritorio y aplicaciones
- Pidora, versión Fedora Remix optimizada102
- QtonPi, distribución Linux con un framework de aplicaciones multiplataforma basado en Qt framework
- Raspbian,103 versión de Debian Wheezy para ARMv6 con soporte para coma flotante por hardware
- Slackware ARM, también conocida como ARMedslack
- Ubuntu MATE
- Plan 9 from Bell Labs104 105

⁷ ARM: Arquitectura RISC de 32 bits y recientemente 64 bits

- RISC OS 52
- Unix
- FreeBSD106
- NetBSD107 108
- Windows 10
- Windows CE

Distribuciones ligeras multipropósito:

- Minibian, distribución ligera basada en Raspbian.
- Moebius, distribución ligera ARM HF basada en Debian que usa el repositorio de Raspbian y que cabe en una tarjeta SD de 1GB, usa pocos servicios y está optimizada para usar poca memoria
- Squeezed Arm Puppy, una versión de Puppy Linux (Puppi) para ARMv6 (sap6)
- específicamente para Raspberry Pi109
- Distribuciones ligeras de único propósito:
- Instant WebKiosk, sistema operativo con solo un navegador
- IPFire
- Micro Elastix, solución de código abierto para comunicaciones unificadas110
- OpenELEC
- OSMC
- Raspbmc
- Xbian

2.3.4 Sistema operativo SAT

La estación de alerta temprana tiene como sistema operativo la versión Raspbian Wheezy 7.8 con la versión de kernel 4.1.6. Raspbian es un sistema operativo libre basado en Debian optimizado para el hardware de Raspberry Pi.

Un sistema operativo es el conjunto de programas básicos y utilidades que hacen que funcione la Raspberry Pi y por lo tanto la estación de alerta temprana.

Raspbian no está afiliada con la Fundación Raspberry Pi. Raspbian fue creado por un equipo pequeño y dedicado de desarrolladores que son fanáticos del hardware de

Raspberry Pi, los objetivos educativos de la Fundación Raspberry Pi y, por supuesto, el proyecto Debian.

Raspbian Wheezy provee soporte para la comunicación con dispositivos de bajo nivel, comunicación SPI, I2C, PWM mediante bibliotecas externas de python y un kernel rediseñado por la compañía Adafruit para el control de una pantalla LCD SPI táctil resistiva de 3.5”.

2.3.5 GPIOs⁸

La Raspberry Pi modelo B tiene 2 header de expansión para comunicación con hardware de bajo nivel, como los sensores con que cuenta la estación de alerta temprana, el primer header está compuesto de 26 pines denominado P1 de los cuales 8 son GPIO de propósito general, 2 pines para comunicación con dispositivos que utilizan en protocolo de comunicación i2c, 5 pines para dispositivos seriales que utilizan el protocolo de comunicación SPI. 2 pines para el protocolo de comunicación serial UART y 3 pines de alimentación que proporcionan los niveles de tensión de 5.0 v. 3.3 v y tierra para dispositivos externos. Algunos de los pines de este header comparten función como pines de propósito general y como de hardware específico; dependiendo de la configuración que se asigna en el software así pueden ser de propósito general aunque hayan sido definidos por hardware como pines dedicados.

⁸ **GPIO**: General purpose input output. Es un pin generico cuyo comportamiento se puede programar

			P1					
<50mA	3V3		1	2		5V		
BCM GPIO00/02	SDA0/1	8	3	4		5V		
BCM GPIO01/03	SCL0/1	9	5	6		GND		
BCM GPIO04		7	7	8	15	TX	BCM GPIO14	
	GND		9	10	16	RX	BCM GPIO15	
BCM GPIO17		0	11	12	1	PWM0	BCM GPIO18	
BCM GPIO21/27		2	13	14		GND		
BCM GPIO22		3	15	16	4		BCM GPIO23	
<50mA	3v3		17	18	5		BCM GPIO24	
BCM GPIO10	SPIMOSI	12	19	20		GND		
BCM GPIO9	SPIMOSO	13	21	22	6		BCM GPIO25	
BCM GPIO11	SPI SCLK	14	23	24	10	SPI CE0 N	BCM GPIO08	
	GND		25	26	11	SPI CE1 N	BCM GPIO07	
			P5					
<50mA	3V3		2	1		5V		
BCM GPIO29	SCL0	18	4	3	17	SDA0	BCM GPIO28	
BCM GPIO31		20	6	5	19		BCM GPIO30	
	GND		8	7		GND		

ILUSTRACIÓN 2 RASPBERRY PI, HEADERS P1 Y P2 (KLEBACK, S.F.)

El segundo header consta de 8 pines de los cuales 4 son para alimentación de dispositivos externos. Los 4 restantes son GPIO de propósito general

Características eléctricas GPIO

Los niveles de tensión son de 3,3 V y no son tolerantes a 5 V. No hay protección contra la sobre tensión en el header

El consumo máximo permitido actual del pin de 3,3 V es de 50 mA. El consumo de corriente desde el pin 5 V es la corriente de entrada USB (por lo general 1 A) menos cualquier consumo de corriente del resto de la placa. Para el modelo B 1000 mA - 700 mA del sistema, permite una máxima corriente de 300 mA para el pin de 5V.

2.4 Pantalla táctil

La estación de alerta temprana cuenta con una pantalla táctil de 3.5 pulgadas que le permite al operador revisar las condiciones medioambientales que reporta la estación de alerta temprana, además de permitir realizar ciertas operaciones tales como el reinicio y apagado de la estación, realizar llamadas telefónicas y tomar fotografías o videos en el sitio de la estación.

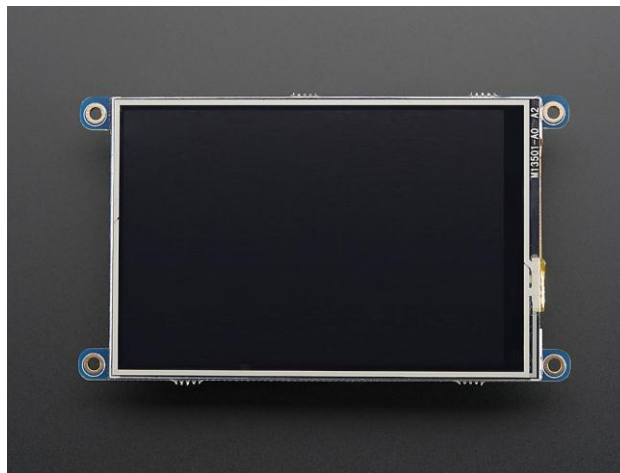


ILUSTRACIÓN 3. PANTALLA TÁCTIL (ADAFRUIT, S.F.)

La pantalla utiliza el protocolo de comunicación SPI lo que permite una interfaz de conexión con 7 pines GPIO.

La configuración y comunicación con la pantalla esta soportada por el sistema operativo, que ha añadido al kernel el código necesario para utilizar la pantalla como si de un monitor se tratara.

2.4.1 Codificación

El control de la estación de alerta temprana se hace mediante la pantalla táctil. Se muestra a continuación fragmento del código empleado. Se utiliza la biblioteca de python Tkinter para la generación de la interfaz gráfica y el soporte touch lo realiza el sistema operativo.

```

.
class Ventana(Frame):
    def __init__(self, padre): # Creación de un frame principal de fondo
        Frame.__init__(self, padre, background="#333")
        self.padre = padre # Instancia de ventana inicial
        self.fecha = StringVar() # Variable de tipo cadena para almacenar la
fecha
.
.
.
        self.estado = StringVar() # Variable de tipo cadena para almacenar el
estado del sistema
        self.estado.set("EN ESPERA") # Asignación (seteo) de variable de estado
del sistema
        self.error = StringVar() # Variable de tipo cadena para el almacenaje
de mensajes o códigos de error de la estación
        self.coordenadas = "" # Variable tipo cadena que almacena las coordena-
das geodésicas de la estación de alerta temprana
        self.datos = "" # Variable de tipo cadena para almacenar los datos que
se manejan en la estación
.
.
.
        self.inicializarInterface() # Llamado a la función que realiza el
dibujado de la interface básica

    def inicializarInterface(self):
        self.padre.title("SAT") # título de la ventana padre
        self.padre.overrideRedirect(1) # se elimina la decoración de ventanas
        self.padre.geometry('480x320') # geometría de la ventana padre
        self.padre.resizable(0,0) # Sin redimensión de ventana padre
        self.pack(fill=BOTH, expand=1) # Ajuste de frame principal

##### FRAMES PARA ESTRUCTURA DE APLICACION
#####

# frame que agrupa los botones de monitoreo de sensores
self.frameBotones = Frame(self, background="#222")
self.frameBotones.pack(fill=BOTH, side=RIGHT)

# frame que agrupa el label de fecha
self.frameFecha = Frame(self, background="#222")
self.frameFecha.pack(fill=BOTH, side=TOP)
.

```

```

.
.
##### WIDGETS DE APLICACION
#####
#####
    # Label que muestra la fecha y hora
    self.labelSensor = Label(self.frameFecha, font="RADIOLAND 12", tex-
tvar=self.fecha, bg="#222", fg="#8FF")
    self.labelSensor.pack(side=RIGHT)
.
.
.
    # Canvas de muestra de datos de los sensores
    self.canvasDatos = Canvas(self, bg="#333")
    self.nombre = self.canvasDatos.create_text(10,0, anchor=NW, font="RADI-
OLAND 15", text=self.sensor, fill="#FFF")
    self.lectura = self.canvasDatos.create_text(10,55, anchor=NW, font="RA-
DIOLAND 50", text=self.datos, fill="#FFF")
    self.unidades = self.canvasDatos.create_text(10,117, anchor=NW,
font="RADIOLAND 15", text=self.unidad, fill="#FFF")
    self.gps = self.canvasDatos.create_text(10,160, anchor=NW, font="RADIO-
LAND 13", text=self.coordenadas, fill="#F88")
    self.hardwareEstado = self.canvasDatos.create_text(200,160, anchor=NW,
font="RADIOLAND 13", text=self.hardware, fill="#F88")
    self.canvasDatos.pack(fill=BOTH, expand=1)
.
.
.
    # Actualizador de fecha, hora, batería, señal
    def timer(self):

        self.fecha.set(strftime('%A %d %B %Y %X'))
        if self.llamadaActiva > 0:
            self.output.set('\r'+self.cadena+"\nRestante: "+str(self.llamadaAc-
tiva))
            self.llamadaActiva = self.llamadaActiva -1
        else:
            self.mensajes.set("\nActividad del Sistema")
            self.output.set("\nRAM: %s\t SWAP: %s\nDISK: %s" %( psutil.vir-
tual_memory().percent, psutil.swap_memory().percent, psutil.disk_us-
age('/').percent))
.
.
.
        with open('internal_temp', 'r') as archivo:
            temporal = archivo.read()

```



```

        while temporal == '':
            temporal = archivo.read()
            temporal = float(temporal)/1000.0
            interna = str(temporal)
        .
        .
        .

        self.after(500, self.timer)

# lectura de temperatura
def mostrarTemperatura(self):
    system("sudo ./SAT/buzzer.py -v 1 -t 50 &")
    with open('temperatura', 'r') as archivo:
        self.datos = archivo.read()
        while self.datos == '':
            self.datos = archivo.read()
        self.canvasDatos.itemconfigure(self.nombre, text="\rTEMPERATURA")
        self.canvasDatos.itemconfigure(self.unidades, text="GRADOS CELSIUS")
        self.canvasDatos.itemconfigure(self.lectura, text=self.da-
tos.split()[0])
        .
        .
        .

def apagar(self):
    system("sudo ./SAT/buzzer.py -v 1 -t 50 &")
    if self.unlock:
        self.unlock = 0
        outfd = open('archivo_out', 'w+')
        errfd = open('archivo_err', 'w+')

        self.mensajes.set("\nApagando Sistema")

        call(['sudo`.` shutdown', '-h', 'now'], stdout=outfd, stderr=errfd)

        outfd.close()
        errfd.close()

        fd = open('archivo_out', 'r')
        output = fd.read()
        fd.close()

        fd = open('archivo_err', 'r')
        err = fd.read()
        fd.close()

```

```

        self.output.set('\n'+output)
        self.output.set('\n'+err)
    else:
        if not self.lockkeypad:
            self.teclado("password")
    .
    .
    .
def main():
    root = Tk()
    app = Ventana(root)
    root.mainloop()

if __name__ == '__main__':
    main()

```

SCRIPT 1. INTERFAZ DE PANTALLA DE CONTROL

El script implementa una clase denominada ventana que contiene todas las variables utilizadas para el almacenamiento de los datos temporales que utiliza para mostrar al usuario. El script recoge de los archivos generados por los sensores y demás scripts con los que cuenta la estación de alerta temprana y los muestra en un widget tipo canvas. La interfaz cuenta con dos pantallas de operación la primera es de presentación de datos al operador y la segunda es de mantenimiento.

En la pantalla de presentación, solo se muestran los datos de los sensores de la estación de alerta temprana. Mediante el panel de botones lateral se puede cambiar al tipo de sensor que se desea observar.

En la pantalla de mantenimiento se realizan operación de apagado o reinicio de la estación para realizar algún procedimiento de actualización, verificación o limpieza. La pantalla de mantenimiento también permite tomar fotografías y videos mediante la cámara de la estación además permite realizar llamadas telefónicas de corta duración esto con el propósito de realizar llamadas por parte de los usuarios hacia las autoridades para reportar alguna condición o anomalía en la estación.

2.5 Sensor de temperatura - DS18B20

El sistema de alerta temprana realiza el monitoreo de la temperatura mediante un sensor de temperatura digital este es el encargado de entregar el valor de temperatura ambiente.

2.5.1 Descripción general

El termómetro digital DS18B20 proporciona 9 bits a 12 bits para mediciones de temperatura Celsius y tiene una función de alarma de temperatura superior e inferior no volátiles programadas. El DS18B20 se comunica a través de un bus 1-Wire, que por definición requiere una sola línea de datos (y tierra) para la comunicación con un microprocesador central.

Cada DS18B20 tiene un código de 64 bits de serie único, que permite que múltiples DS18B20 funcionen en el mismo bus 1-Wire. Por lo tanto, es fácil de usar un microprocesador para controlar muchos DS18B20s distribuidos sobre un área grande. Sin embargo la estación de alerta temprana no necesita más de uno para realizar las mediciones adecuadamente.

2.5.2 Características generales

- Interfaz 1-Wire[®] requiere un solo pin para la comunicación.
- Rango de temperatura -55 ° C a + 125 ° C. (-67 ° F a + 257 ° F)
- ± 0,5 ° C Precisión de -10 ° C a + 85 ° C.
- Resolución programable 9 Bits a 12 Bits.

2.5.3 Protocolo de comunicación

El sensor de temperatura DS18B20 utiliza un protocolo de comunicación serie desarrollado por Dallas semiconductor. Está basado en un bus, un maestro y varios esclavos de una sola línea de datos en la que se alimentan y que necesita una referencia a tierra común a todos los dispositivos.

2.5.4 Valores máximos de operación

Parámetro	Valor
Rango de tensión en cualquier pin con respecto a tierra	-0,5 V a +6,0 V
Rango de temperatura de funcionamiento	-55°C a +125°C
Rango de temperatura de almacenamiento	-55°C a +125°C

TABLA 2. DS18B20, VALORES MÁXIMOS DE OPERACIÓN (MAXIM INTEGRATED)

2.5.5 Configuración de pines

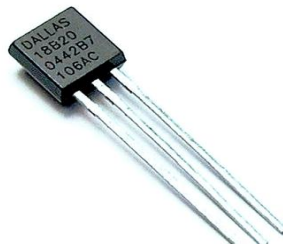


ILUSTRACIÓN 4. DS18B20

1. Tierra
2. Datos de entrada/salida. Drenaje abierto, pin de interfaz 1-Wire.
3. VDD pin de alimentación.

2.5.6 Características eléctricas

Parámetro	Símbolo	Condición	Min	Tip	Max	Unidades
Tensión de alimentación	VDD	Tensión local VDD	+3.0		+5.5	V

Tensión de alimentación Pullup	VPU	Tensión local VDD	+3.0	+5.5	V
Error de termómetro	TERR	-10°C to +85°C	±0.5		°C
		-55°C to +125°C	±2		
Entrada lógica baja	VIL	Tensión local VDD	-0.3	+0.8	V
Entrada lógica alta	VIH		+2.2	VDD+0.3	V
Corriente de sumidero	IL	VI/O = 0.4V	4.0		mA
Corriente Standby	IDDS		750	1000	nA

TABLA 3. DS18B20 CARACTERÍSTICAS ELÉCTRICAS (MAXIM INTEGRATED)

2.5.7 Codificación

Para realizar la lectura de la temperatura mediante el sensor de temperatura DS18B20 existe un script que permite la comunicación entre el sensor y el microprocesador, dicha codificación, realiza la captura de la temperatura en grados Celsius y a la vez almacena el valor en un archivo de disco para posterior envío hacia el servidor y tratamiento de los datos.

La estación de alerta temprana utiliza la biblioteca `w1thermsensor` que implementa un módulo de python para la lectura de temperatura de sensores 1-wire. La biblioteca proporciona soporte para los siguientes dispositivos: DS18S20, DS1822, DS18B20, DS28EA00, DS1825/MAX31850K. La instalación se hace mediante los repositorios del sistema para el paquete de python: `python-w1thermsensor`

La biblioteca `w1thermsensor` requiere de paquetes adicionales para funcionar, los paquetes son `moc`, `six`, `coverage`, `sure`, `nose`. Estos paquetes se encuentran en los repositorios de Linux para plataformas ARM.

```

.
.
from w1thermsensor import W1ThermSensor # Manejo de sensor de temperatura
.
.
class temperatura:
    def __init__(self): # inicializar el sensor
        self.sensor = W1ThermSensor()

```

```

def obtenerCelsius(self): # obtener la temperatura Celsius
    self.celsius = '{0:0.2f}'.format(self.sensor.get_temperature())
.
.
if __name__=="__main__": # programa principal
    ds18b20 = temperatura()

```

SCRIPT 2. LECTURA SENSOR DE TEMPERATURA

La codificación implementa programación orientada a objetos mediante la creación de una clase denominada temperatura, esta clase se encarga de iniciar el dispositivo utilizando la biblioteca `w1thermsensor` que por definición utiliza el pin 4 del GPIO BCM para comunicarse con el dispositivo. Se han creado 3 funciones para la obtención de la temperatura en las escalas de temperatura más comunes así mismo se aplica un formato para la presentación de datos con dos decimales. La temperatura se almacena en un archivo de texto plano en el disco para posterior registro.

2.6 Sensor de humedad DHT11

La humedad ambiente es registrada por el sensor de humedad relativa porcentual, este sensor tiene la capacidad de registrar tanto la humedad ambiente como la temperatura pero en la estación de alerta temprana se utiliza para la medición de la humedad. Este cuenta con un sensor de temperatura y humedad complejo con una señal de salida digital calibrada. Este sensor incluye un componente de medición de humedad de tipo resistivo y un componente de medición de temperatura NTC, y se conecta a un microcontrolador de alto rendimiento de 8 bits, que ofrece una excelente calidad, respuesta rápida, la capacidad anti interferencia y rentabilidad.

2.6.1 Características generales

- Calibrado de fábrica.
- Tensión 3.5v-5.5v DC
- Humedad relativa entre 20%-95%. Margen error 5%.
- Temperatura entre 0 y 50 °C. Margen error 2%.
- Resolución de temperatura y humedad en rangos de 1 (no proporciona decimales).

Parámetro	Condición	Mínimo	Típico	Máximo
Resolución	Medición	1% RH	1% RH 8 bits	1% RH
Perceptibilidad			+/- 1% RH	
Exactitud	0 °C ~ 50 °C			+/- 5% RH
Rango de medición	0 °C 25 °C 50 °C	30 % RH 20 % RH 20 % RH		90 % RH 90 % RH 80 % RH
Tiempo de respuesta	63 % RH y 25 °C	6 segundos	10 segundos	15 segundos
Histéresis			+/- 1 % RH	
Estabilidad a largo plazo			+/- 1 % RH/año	

TABLA 4. DHT11, CARACTERÍSTICAS SENSOR DE HUMEDAD (D-ROBOTICS)

Parámetro	Condición	Mínimo	Típico	Máximo
Resolución		1 °C 8 bits	1 °C 8 bits	1 °C 8 bits
Repetitividad			+/- 1°C	
Exactitud		+/- 1 °C		+/- 2 °C
Rango de medición		0 °C		50 °C
Tiempo de respuesta	63 % RH	6 segundos		30 segundos

TABLA 5. DHT11 CARACTERÍSTICAS SENSOR DE TEMPERATURA (D-ROBOTICS)

2.6.2 Protocolo de comunicación

El sensor de humedad relativa utiliza un protocolo de comunicación serie denominado Single-Wire Two-Way, que es un formato de datos de un solo bus. La sincronización entre el microprocesador y el sensor DHT11 es un proceso de comunicación de aproximadamente 4 ms.

2.6.3 Configuración de pines

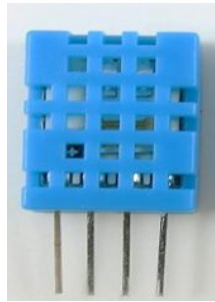


ILUSTRACIÓN 5. DHT11

1. VCC pin de alimentación
2. Datos de entrada salida
3. Sin conexión
4. Tierra

2.6.4 Características eléctricas

Parámetro	Condición	Min	Tip	Max	Unidades
Tensión de alimentación	DC	3.0	5.0		V
Corriente de alimentación	Medición	0.5			mA
	Promedio	0.2		1.0	mA
	Standby	100			uA
Periodo de muestreo		1.0			s

TABLA 6. DHT11 CARACTERÍSTICAS ELÉCTRICAS (D-ROBOTICS)

2.6.5 Codificación

Para la lectura de la humedad relativa la estación de alerta temprana utiliza una biblioteca denominada Adafruit_DHT de código libre. Debido al protocolo de comunicación del sensor DHT11, la librería de python hace uso del enclavamiento GPIO a través de C. La biblioteca permite soporte para sensores modelo DHT 11, DHT 22 y AM2302

```

.
.
import Adafruit_DHT # Manejo de sensor de humedad relativa
.
.
class humedad:
    def __init__(self): # inicializar el sensor
        self.sensor = Adafruit_DHT.DHT11
        self.pin = 17

    def obtenerHumedad(self): # obtener la humedad relativa
        self.humedad, self.temperatura = Adafruit_DHT.read_retry(self.sensor, self.pin)
.
.
.
if __name__ == "__main__": # programa principal
    dht11 = humedad()
.
.

```

SCRIPT 3. LECTURA SENSOR DE HUMEDAD

La codificación implementa programación orientada a objetos mediante la creación de una clase denominada humedad, esta se encarga de iniciar el sensor de humedad y definir el pin 17 GPIO BCM para la comunicación con el sensor. Se ha creado una función para obtener la humedad relativa porcentual con un formato para el registro de 2 posiciones decimales. El valor obtenido se almacena en un archivo de texto plano para posterior procesamiento y registro de datos.

2.7 Sensor de presión barométrica BMP180

Para el monitoreo de la presión barométrica la estación de alerta temprana cuenta con un sensor BMP180 este dispositivo utiliza el protocolo de comunicación I2C para ser conectado directamente a la Raspberry Pi. Este dispositivo puede mostrar tanto la presión barométrica así como la temperatura y esta compensado internamente con los coeficientes almacenados en EEPROM.

El BMP180 consiste en un sensor piezo-resistivo, un convertidor analógico a digital, una unidad de control con EEPROM y una interfaz I2C serie. El BMP180 proporciona el valor no compensado de la presión y la temperatura. La EEPROM tiene almacenado 176 bits de los datos de calibración individuales. Esto se utiliza para compensar el offset, dependiendo de la temperatura y otros parámetros del sensor.

2.7.1 Características generales

- Rango de presión: 300 ~ 1100 hPa
- Tensión de operación: 1.8 ~ 3.6 Vdc
- Baja potencia: 5uA a 1 muestra/segundo
- Bajo ruido: 0.06 hPa (0.5m)
- Comunicación: I2C
- Completamente calibrado

2.7.2 Protocolo de comunicación

El sensor de presión barométrica utiliza comunicación I2C. Este un protocolo de comunicación serie diseñado por Philips que se utiliza esencialmente entre dispositivos que pertenecen al mismo circuito.

2.7.3 Configuración de pines

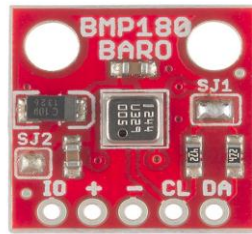


ILUSTRACIÓN 6. BMP180

1. IO- Tensión de alimentación baja
2. Tensión de alimentación
3. Tierra
4. Reloj
5. Datos

2.7.4 Características eléctricas

Parámetro	Símbolo	Condición	Min	Tip	Max	Unidad
Temperatura de operación	TA	Funcionamiento	-40		+85	°C
Tensión de alimentación	VDD	Rizado máximo de 50mVpp	1.8	2.5	3.6	V
Corriente de alimentación	IDDSTD	Modo estándar		5		uA
Corriente pico	Ipeak	Conversión		650	1000	UA
Corriente de standby	IDDSBM	@ 25 °C		0.1	4	UA
Exactitud relativa de presión		950 ~ 1050 hPa @ 25°C		+/- 0.12		hPa
				+/- 1.0		m
Exactitud absoluta de presión		300 ~ 1100 hPa 0 ~ 65 °C	-4.0	1.0	+2.0	hPa

Parámetro	Símbolo	Condición	Min	Tip	Max	Unidad
Resolución	Presión		0.01			hPa
	temperatura		0.1			°C
Exactitud absoluta de temperatura		0 ~ 65 °C	-2.0	+/-1.0	+2.0	°C
Tiempo de conversión de presión	tc_p_std	Modo estándar		5	7.5	ms
Tiempo de conversión de temperatura	tc_temp	Modo estándar		3	4.5	Ms
Frecuencia de reloj serie	fscl				3.4	Mhz
Estabilidad a largo plazo	0 ~ 65 °C	12 meses		+/- 1.0		hPa

TABLA 7. BMP180, CARACTERÍSTICAS ELÉCTRICAS (BOSCH SENSORTEC)

2.7.5 Codificación

Para la lectura de la presión barométrica la estación de alerta temprana utiliza la biblioteca Adafruit_BMP de. La biblioteca hace uso del soporte nativo de la placa Raspberry Pi y por lo tanto emplea el uso de los paquetes python-smbus e i2c-tools que se instalan de los repositorios. La configuración de estos paquetes se realiza en el archivo “/etc/modules”. Al final del archivo es necesario incluir los módulos “i2c-bcm2708” e “i2c-dev” para la carga de estos, en el sistema. La biblioteca provee soporte para dispositivos modelo BMP085 y modelo BMP180.

```

.
.
import Adafruit_BMP.BMP085 as BMP # Manejo de sensor de presión barométrica
.
.
class presión:
    def __init__(self): # inicializar el sensor
        self.sensor = BMP.BMP085()
.
.

```

```
def obtenerPresion(self): # obtener la presión en Pascales
    self.presion = '{0:0.2f}'.format(self.sensor.read_pressure())
.
.

if __name__=="__main__": # programa principal
    bmp180 = presion()
.
```

SCRIPT 4. LECTURA SENSOR DE PRESIÓN

Se ha creado una clase denominada presión que realiza la iniciación del sensor, además la codificación incluye 4 funciones para la obtención de los datos proporcionados por el sensor para temperatura, presión, altitud y presión al nivel del mar respectivamente. El dato que se envía hacia la página web de monitoreo y de almacenamiento de la base de datos es la presión.

2.8 Sensor de distancia HC-SR04

La determinación del nivel que ha subido en metros el río o quebrada se realiza con un sensor de distancia de ultrasonido, el cálculo se realiza tomando en cuenta el nivel de referencia del río o quebrada como una distancia en metros hacia la estación de alerta temprana y a partir de ese punto si la distancia medida de la estación hacia el río aumenta significa que el nivel o altura del río ha bajado y en caso contrario si la distancia que separa la estación de alerta temprana con el río en dirección vertical disminuye significa que el río ha subido su nivel o altura en metros.

El sensor de ultrasonido utiliza una frecuencia de 40 KHz para emitir un tono que viaja hasta la superficie del río y retorna hacia el sensor la distancia se calcula en base a la velocidad del sonido y el tiempo en que el tono regresa, es decir el tiempo de vuelo dividido por 2.

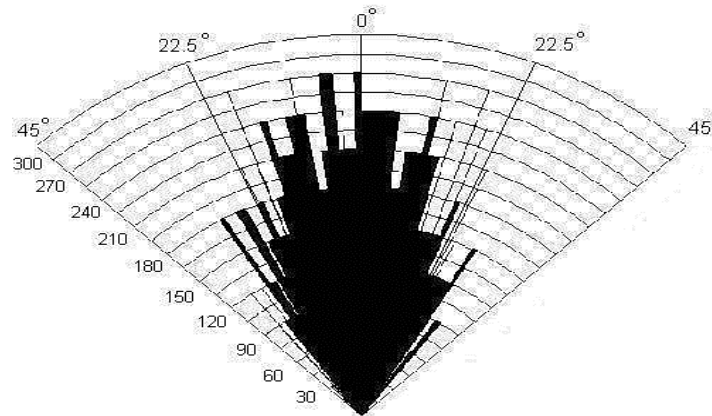


ILUSTRACIÓN 7. HC-SR04, RANGO EFECTIVO DE DETECCIÓN (ELECTFREAKS)

2.8.1 Características generales

Parámetro	Valor
Frecuencia:	40 KHz.
Distancia Mínima:	2 cm.
Distancia Máxima:	400 cm.
Sensibilidad:	Detecta un palo de escoba a 3 m.
Pulso de Disparo	10 uS min. TTL
Pulso de Eco:	100 uS - 18 mS
Retardo entre pulsos:	10 mS Mínimo
Tamaño:	43 x 20 x 17 mm
Peso:	10 gr.
Angulo efectivo	30 °
Angulo de medición	15 °

TABLA 8. HC-SR04, CARACTERÍSTICAS DE SENSOR (ELECTFREAKS)

2.8.2 Protocolo de comunicación

No se utiliza un protocolo estándar de comunicaciones, simplemente una señal de disparo de 10 us de duración y una lectura de la señal de eco recibido

2.8.3 Configuración de pines



ILUSTRACIÓN 8. HC-SR04

1. VCC, tensión de alimentación
2. Trigger, disparo
3. Echo, retorno
4. Tierra

2.8.4 Características eléctricas

Parámetro	Valor
Tensión de alimentación	5.0 Vdc
Corriente de trabajo	15 mA
Corriente de reposo	2 mA
Frecuencia de señal	40 KHz

TABLA 9. HC-SR04, CARACTERÍSTICAS ELÉCTRICAS (ELECTFREAKS)

2.8.5 Codificación

Se utiliza una biblioteca denominada RPi.GPIO de propósito general para la comunicación con el dispositivo, la biblioteca provee soporte para el manejo de pines GPIO en el entorno de python. La biblioteca RPi.GPIO que utiliza la estación de alerta temprana es la versión 0.5.10, está tiene por defecto instalada en la distribución de Linux para Raspberry Pi en una versión más baja solo necesita actualización al momento de instalar el sistema operativo.

```
import RPi.GPIO as GPIO
.
.
def Pulso10us():
    GPIO.output(TriggerPin, GPIO.HIGH)
    time.sleep(0.00001)
    GPIO.output(TriggerPin, GPIO.LOW)

def Distancia():
    Pulso10us()
    start = time.time()
    while not GPIO.input(EchoPin):
        start = time.time()
    while GPIO.input(EchoPin):
        stop = time.time()
    lapso = stop - start
.
.
    valor = (lapso*(331.5+(0.61*temperatura)))/2
    metros = '{0:0.2f}'.format(valor)
    return metros

distance = Distancia()
.
.
```

SCRIPT 5. LECTURA SENSOR DE DISTANCIA

El sensor se conecta a la Raspberry Pi mediante los pines 22 y 23 GPIO BCM. En el código se definen 2 funciones, la función Pulso10us () que genera un pulso de 10 micro segundos para generar el pulso de disparo del sensor de ultrasonido y la segunda función

Distancia () se encarga de calcular la distancia en base al tiempo de vuelo del tono generado por el sensor a una frecuencia de 40 KHz y la velocidad del sonido.

La variación de temperatura en las zonas de observación presenta cambios drásticos. Por lo que se realiza una compensación de temperatura dado que la velocidad del sonido varía en el aire con la variación de la temperatura.

La fórmula de la velocidad del sonido se expresa en función de la temperatura t del gas en grados Celsius.

$$v_s = \sqrt{\frac{\gamma RT}{M}} = \sqrt{\frac{\gamma R}{M}(T_0 + t)} \approx \sqrt{\frac{\gamma RT_0}{M}} + \frac{1}{2} \sqrt{\frac{\gamma R}{MT_0}} t$$

**ECUACIÓN 1. VELOCIDAD DEL SONIDO EN FUNCION DE LA TEMPERATURA
(VELASCO S.)**

Para obtener esta expresión aproximada, se han tomado los dos primeros términos del desarrollo de $(1+t/T_0)^{1/2}$ por el binomio de Newton

Sabiendo que:

$$T_0 = 273.15 \text{ K}, \quad \gamma = 1.4,$$

$$R = 8.314 \text{ J/(K}\cdot\text{mol)} \quad M = 28.95 \cdot 10^{-3} \text{ kg/mol},$$

Tenemos que:

$$v_s \approx 331.5 + 0.61 \cdot t$$

**ECUACIÓN 2. VELOCIDAD DEL SONIDO EN FUNCION DE LA TEMPERATURA SIMPLIFICADA
(VELASCO S.)**

Donde 331.5 m/s es la velocidad del sonido en el aire a 0°C.

Para temperaturas cercanas a la ambiente, la velocidad del sonido en el aire varía aproximadamente de forma lineal con la temperatura.

Finalmente los valores obtenidos se almacenan en un archivo de texto plano en el disco. Y se lanza el programa de alarmas para determinar en base al nivel del río si es necesario emitir alertas por llamadas telefónicas y mensajes de texto

2.9 Sensor de precipitación WS 2300

Las lluvias son un factor que influye directamente sobre la altura que un río pueda tomar y por eso determinar el nivel de precipitación por lluvia para tener un estimación de las condiciones que se puedan presentar a futuro, es decir en base a una tasa de precipitación dada en el tiempo fácilmente se puede pronosticar la cantidad de agua depositada al río o quebrada y así estimar su altura cuantitativamente si se tuviese una sección del río encausada en un canal de dimensiones conocidas.

El sensor que realiza la medición de las precipitaciones por lluvias es un pluviómetro electrónico de tipo báscula de auto vaciado que genera una señal para cada 0.2794 mm de lluvia precipitada.



ILUSTRACIÓN 9. WS2300 (SPARKFUN)

2.9.1 Características generales

- Depósito tipo báscula con auto vaciado
- Conexión tipo switch magnético
- Se puede utilizar mediante interrupciones de hardware o enclavamiento por software.

2.9.2 Protocolo de comunicación

El dispositivo que envía la señal de actividad es un switch magnético; no se utiliza ningún protocolo con formato en la transmisión de los datos. Simplemente está configurado para enviar un pulso activo bajo.

2.9.3 Codificación

Se utiliza una biblioteca denominada RPi.GPIO de propósito general utilizada con el sensor de distancia por ultrasonido para la comunicación con el dispositivo, la librería provee soporte para el manejo de pines GPIO en el entorno de python.

```
import RPi.GPIO as GPIO
.
def obtenerPrecipitacion(lluvia):
    precipitacion = lluvia*0.2794
    archivo = open('/home/pi/precipitacion', 'w')
    archivo.write(str(precipitacion))
    archivo.close()
.
GPIO.add_event_detect(27, GPIO.FALLING, callback = click, bouncetime = 2000)

while True:
    time.sleep(10)
```

SCRIPT 6. LECTURA DE SENSOR DE PRECIPITACIÓN

El pin 27 GPIO BCM es el que se utiliza para la comunicación con el pluviómetro, cada pulso que este envía genera una detección de evento por flanco descendente que llama a la función click () que realiza el conteo de los pulsos enviados; Esta misma función se encarga de llamar a la función obtenerPrecipitacion () donde se realiza el cálculo por el factor de conteo y se almacena hacia un archivo de texto plano almacenado en el disco.

2.10 Placa 3G

La estación de alerta temprana está diseñada para estar ubicada en la intemperie en donde no se cuenta con ningún tipo de conexión a Internet o toma eléctrica, esta cuenta con su propio módulo de comunicación de tercera generación 3G que le permite tener conectividad a Internet para poder realizar la actualización de los datos hacia el servidor de almacenamiento. Además este módulo permite que la estación de alerta temprana pueda emitir alertas mediante mensajes de texto y llamadas telefónicas a las personas que se encuentran en riesgo y que previamente fueron ingresadas al directorio telefónico de la placa Raspberry Pi.

2.10.1 Telefonía 3G

3G es la abreviación de tercera generación de transmisión de voz y datos a través de telefonía móvil mediante UMTS⁹. Los servicios asociados con la tercera generación proporcionan la posibilidad de transferir tanto voz como datos (una llamada telefónica o un video llamado) y datos no-voz (como la descarga de programas, intercambio de correos electrónicos, y mensajería instantánea). Las tecnologías de 3G son la respuesta a la especificación IMT-2000¹⁰ de la Unión Internacional de Telecomunicaciones.

Ventajas

- Transmisión de voz con calidad equiparable a la de las redes fijas.
- Mayor velocidad de conexión, ante caídas de señal.
- Todo esto hace que esta tecnología sea ideal para prestar diversos servicios multimedia móviles.

Desventajas

- Aparición del efecto conocido como «respiración celular», según el cual, a medida que aumenta la carga de tráfico en un sector (o celda), el sistema va disminuyendo la potencia de emisión, o lo que es lo mismo, va reduciendo el alcance de cobertura de la celda, pudiéndose llegar a generar zonas de "sombra" (sin cobertura), entre celdas adyacentes.

⁹ **UMTS**: Servicio universal de telecomunicaciones móviles.

¹⁰ **IMT-2000**: Estándar global para tercera generación de comunicaciones inalámbricas

2.10.2 Descripción

El módulo 3G es una tarjeta de expansión basado en el módulo SIM5215E y el módulo de la serie SIM5215, es una solución de tipo HSDPA / WCDMA / GSM / GPRS / EDGE¹¹ de múltiples frecuencias que admite la transmisión de datos HSDPA a una velocidad de 3,6 Mbps.

2.10.3 Características módulo SIM5215E

- Banda dual UMTS/HSDPA 900/2100MHz
- Banda triple GSM/GPRS/EDGE 850/900/1800MHz
- GPRS multi-slot class 12
- EDGE multi-slot Class 12
- WCDMA 3GPP release 99
- Potencia de Salida
- UMTS 2100/900: 0.25W (for SIM5215E)
- GSM850/GSM900: 2W
- DCS1800: 1W
- Control Vía comandos AT
- Rango de tensión: 3.3V~ 4.2V
- Temperatura de operación: -30°C to +80°C
- Especificación de transferencia de datos:

WCDMA	Max.384Kbps (DL), Max.384Kbps (UL)
EDGE	Max. 236.8Kbps (DL), Max.118Kbps (UL)
GPRS	Max. 85.6Kbps (DL), Max.42.8Kbps (UL)

¹¹ **HSDPA**: Enlace de descarga de alta velocidad por acceso de paquetes.

WCDMA: Acceso múltiple por división de código de banda ancha.

GSM: Sistema global para las comunicaciones.

GPRS: Servicio general de paquetes via radio.

EDGE: Velocidad de datos mejorada para la evolucion del GSM.



ILUSTRACIÓN 10. MÓDULO DE COMUNICACIÓN 3G

Soporta la utilización de los protocolos TCP/ UDP/ FTP/ HTTP/ HTTPS/ SMTP/ POP3/ MMS¹². El módulo 3G heredó del módulo SIM5215E todas las funciones de la aplicación, también amplió muchas interfaces comunes, incluyendo dos interfaces de cámara, un micrófono y un conector para auriculares, puerto USB, interfaz de tarjeta SIM, la interfaz de tarjeta SD.

2.10.4 Características módulo 3G

- Soporte para múltiples módulos SIM: SIM5216A, SIM 5216E, SIM5216J.
- Condensador de respaldo para RTC.
- Comunicación 3G.
- Soporte para micro SD y tarjeta SIM5215E.
- Interfaz de comunicación para cámaras: OV7670, OV7690, OV7725, OV2640, AK8856

¹² **TCP:** Protocolo de control de comunicación.

UDP: Protocolo de datagramas de usuario.

FTP: Protocolo de transferencia de archivos.

HTTP: Protocolo de transferencia de hipertexto.

POP: Protocolo de oficina postal, para transferencia de correo.

MMS: Servicio de mensajería multimedia.

SMTP: Protocolo de transferencia simple de correo.

- Control vía comandos AT
- Convertidor ADC de 12 bits
- Tensión de operación de 12 Vdc

2.10.5 Protocolo de comunicación

El módulo 3G utiliza el protocolo de comunicación serie UART a través de 2 pines de configuración o a través del puerto de comunicación USB. En este caso se utiliza el puerto de comunicación USB.

2.10.6 Configuración de Pines

Los únicos pines que son necesarios para la comunicación con la estación de alerta temprana son los pines de alimentación, los pines de conexión con la cámara OV7670 de video, los pines del convertidor analógico digital ADC y el pin de encendido /apagado D8.

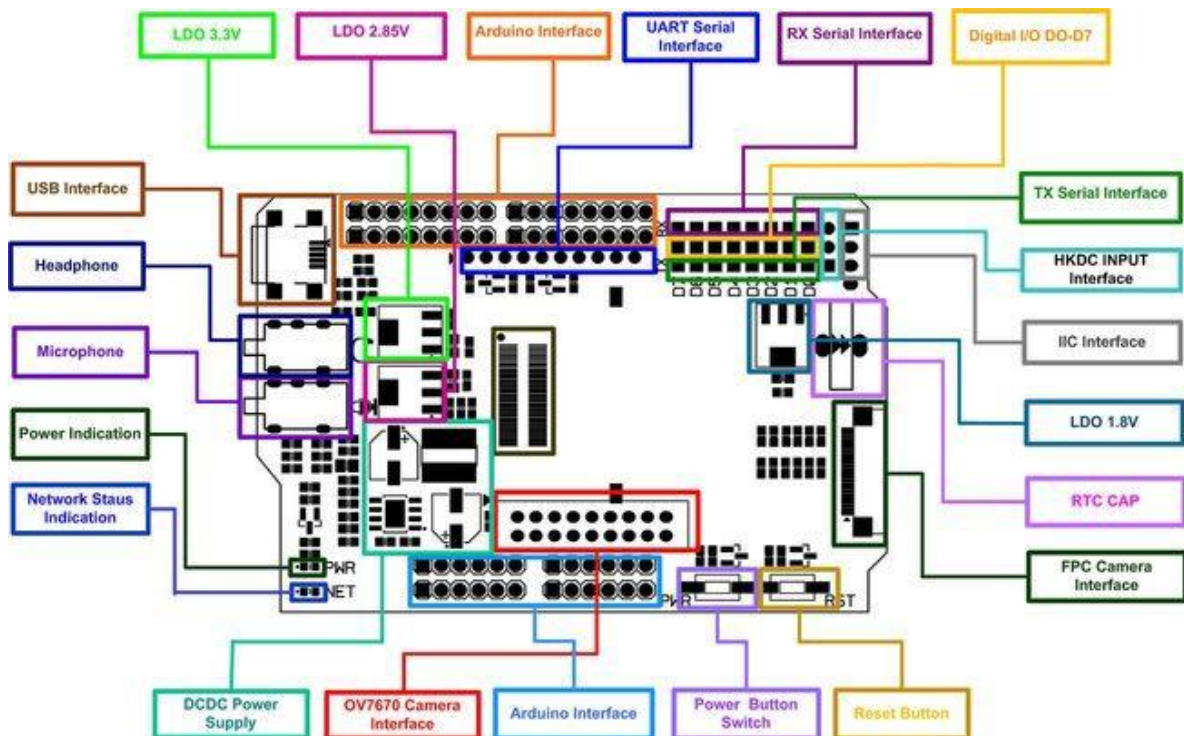


ILUSTRACIÓN 11. MÓDULO 3G, DESCRIPCIÓN DE PINES

LED de estado

LED	Estado
Siempre encendido	Buscando RED/ Conexión de llamada
200ms ON,200ms OFF	Transmisión de datos
800ms ON,800ms OFF	Registrado en la RED
Off	Apagado

TABLA 10. MÓDULO 3G, INDICADORES LED (ELECTFREAKS, S.F.)

2.10.7 Características eléctricas

Parámetro	Mínimo	Típico	Máximo	Unidad
Tensión de alimentación	7		23	Vdc
Tensión de operación	3.3	3.8	4.2	Vdc
Corriente máxima de salida			2.0	A

TABLA 11. MÓDULO 3G, CARACTERÍSTICAS ELÉCTRICAS (ELECTFREAKS, S.F.)

2.11 Cámara OV7670

La cámara de la estación de alerta temprana es un sensor de imagen OV7670 OmniVision VGA en un solo chip es decir cámara y procesador de imagen. A través del bus de control SCCB (Serial Camera Control Bus) el sensor puede enviar el frame completo de varias resoluciones de 8 bits de datos y video hasta un máximo de 30fps



ILUSTRACIÓN 12. OV7670, CÁMARA DIGITAL

La cámara está conectada mediante una interfaz de 18 pines hacia el módulo 3G, de esta manera el control de la cámara se realiza mediante comandos AT y el almacenamiento de los videos y las fotografías se ubican en el micro SD de la placa 3G.

2.11.1 Características generales

- Alta sensibilidad adecuada para aplicaciones de iluminación
- Baja tensión adecuada para aplicaciones embebidas
- SCCB interfaz estándar compatible con la interfaz I2C
- formato de salida RawRGB, RGB (GRB4: 2: 2, RGB565 / 555/444), YUV (4: 2: 2) y YCbCr (4: 2: 2)
- Soporta VGA, CIF, y de una variedad de tamaños CIF a 40x30
- Soporte de escalado de imagen
- Compensación por la pérdida de la lente óptica
- Detección automática de 50 / 60Hz
- Ajuste automático de saturación (ajuste de UV)
- Ajuste automático de realce de bordes
- Ajuste automático de reducción de ruido

2.11.2 Características eléctricas

Matriz fotosensible	640X480
Tensión IO	2,5V a 3,0V (LDO 1.8V)
Potencia	60 mW / 15 fps VGA YUV
Corriente de reposo	<20 μ A
Temperatura de funcionamiento	-30 °C a 70 °C
Estable	0 °C a 50 °C
Formatos de salida	YUV / YCbCr4: 2:2 RGB565 / 555/444 GRB4: 2:2 datos RGB sin procesar
Tamaño óptica	1/6 "
FOV	25 °
Velocidad máxima	VGA a 30 fps
Sensibilidad	1.3V / (Lux-sec)
SNR	46 dB
Rango dinámico	52 dB
Modo de visualización	Progresiva
Exposición electrónica	1 línea a línea 510
Tamaño de píxel:	3.6 μ m x 3.6 μ m
Corriente en oscuridad	12 mV / s a 60 °C

TABLA 12 OV7670, CARACTERÍSTICAS ELÉCTRICAS (OMNIVISION)

2.11.3 Descripción de pines

VDD	GND
SDIOC	SDIOD
VSYNC	HREF
PCLK	XCLK
D7	D6
D5	D4
D3	D2
D1	D0
RESET	PWDN

TABLA 13. OV7670, DESCRIPCIÓN DE PINES (OMNIVISION)

Pin	Tipo	Descripción
VDD	Fuente	Alimentación
GND	Fuente	Tierra
SDIOC	Input	SCCB reloj
SDIOD	Input/Output	SCCB datos
VSYNC	Output	Sincronización Vertical
HREF	Output	Sincronización Horizontal
PCLK	Output	Reloj pixel
XCLK	Input	Reloj del sistema
D0-D7	Output	Salida paralela de video
RESET	Input	Reset (Activo bajo)
PWDN	Input	Apagado (Active alto)

TABLA 14. OV7670, FUNCION DE PINES (OMNIVISION)

2.11.4 Codificación

Los script de python para la captura de video y toma de fotografías se muestran a continuación, Se utilizan comando AT para la configuración de la cámara y la captura de imagen desde el módulo 3G

```
import serial
import time
.
.
class Camara:
    def foto(self, resolucion, balance, brillo, rotacion, zoom, noche):
        movil = wcdma()
        movil.enviarAT2("AT+CCAME", "OK", "ERROR", 5) # Finalizar cámara
        movil.enviarAT2("AT+FSLOCA=1", "OK", "ERROR", 5) # Seleccionar Memoria
        SD para almacenaje
        estado = movil.enviarAT2("AT+CCAMS", "OK", "CAMERA NO SENSOR0", 5) #
        Inicializar cámara
        if estado[1] == 1:
```

```

        movil.enviarAT2("AT+CCAMSETD="+resolucion, "OK", "ERROR", 5) # Re-
solución 640x480
        movil.enviarAT2("AT+CCAMSETWB="+str(balance), "OK", "ERROR", 5) #
Ajustar balance de blancos
        movil.enviarAT2("AT+CCAMSETB="+str(brillo), "OK", "ERROR", 5) #
Ajustar brillo
        movil.enviarAT2("AT+CCAMSETZ="+str(zoom), "OK", "ERROR", 5) # Ajustar zoom
        movil.enviarAT2("AT+CCAMSETR="+str(rotación), "OK", "ERROR", 5) #
Definir la rotación de la cámara
        movil.enviarAT2("AT+CCAMSETN="+str(noche), "OK", "ERROR", 5) # Definir la rotación de la cámara
        time.sleep(5)
        estado = movil.enviarAT2("AT+CCAMTP", "OK", "ERROR", 5) # Tomar fotografía
    .
    .
if __name__=="__main__": # programa principal
    .
    .
    fotografia = Camara()

```

SCRIPT 7. CAPTURA FOTOGRÁFICA

Se implementa un comando que toma los valores de configuración de resolución, balance, fluorescencia, rotación el nivel de zoom o la condición nocturna de la cámara desde la terminal; luego se establece mediante comandos AT el almacenamiento al micro SD para tomar la fotografía. Por defecto la cámara toma fotografías a 640,480 según codificación implementada si no se especifica lo contrario.

```

import serial
import time
.
.
class Camara:
    def video(self, resolucion, balance, brillo, fps, duracion):
        movil = wcdma()
        movil.enviarAT2("AT+CCAME", "OK", "", 5) # Finalizar cámara
        movil.enviarAT2("AT+FSLOCA=1", "OK", "ERROR", 5) # Seleccionar Memoria SD para almacenaje
        estado = movil.enviarAT2("AT+CCAMS", "OK", "CAMERA NO SENSOR0", 5) # Inicializar cámara
        if estado[1] == 1:

```

```

        movil.enviarAT2("AT+CCAMSETD="+resolucion, "OK", "ERROR", 5) # Re-
solución
        movil.enviarAT2("AT+CCAMSETWB="+str(balance), "OK", "ERROR", 5) #
Ajustar balance de blancos
        movil.enviarAT2("AT+CCAMSETB="+str(brillo), "OK", "ERROR", 5) #
Ajustar brillo
        movil.enviarAT2("AT+CCAMSETF="+str(fps), "OK", "ERROR", 5) # Defi-
nir la fps de la cámara
        time.sleep(5)
        estado = movil.enviarAT2("AT+CCAMRS", "D:/Video/", "ERROR", 5) #
Guardar video en SD
    .
    .
if __name__=="__main__": # programa principal
    .
    .
    video = Camara()

```

SCRIPT 8. CAPTURA DE VIDEO

2.12 Envió de alertas

Para el envío de datos hacia el servidor que almacena la base de datos se utiliza la conexión 3G que proporciona el módulo SIM5215E a través del puerto de comunicación mini USB hacia la placa Raspberry Pi. Esta conexión se realiza cada vez que la estación de alerta temprana realiza una actualización de los sensores, alarmas, la ubicación geográfica proporcionada por el GPS y las condiciones de la estación de alerta temprana.

La configuración de la comunicación se realiza desde la propia Raspberry Pi dado que la placa 3G se presenta como un módem de conexión inalámbrica USB. La Raspberry Pi a través del comando wvdial y el archivo de configuración wvdial.conf configura la conexión para el envío de datos utilizando programación python, sin tener que configurar directamente los comandos AT necesarios para la conexión.

2.12.1 Actualización de Sensores, GPS y estación SAT

El envío de los datos hacia el servidor de base de datos la realiza un script de python que se conecta a la página web y es esta la que finalmente los almacena en la base de datos utilizando Django.

```

.
.
import requests # Manejo de POST y GET para la base de datos
.
.
if __name__=="__main__": # programa principal

    gsm = modem()
    if gsm.start():
        time.sleep(5)
        try:
            with open("/home/pi/credenciales") as credenciales:
                data = credenciales.readline().split()
                sat = estacion(data[0],data[1],data[2])

            with open("/home/pi/hostname") as hostname:
                host = hostname.read().split()[0]
                sat.login(host) # inicio de session en sat

            with open("/home/pi/temperatura") as archivo:
                temperatura = archivo.read()
                while temperatura == '':
                    temperatura = archivo.read()
                sat.actualizar(host,"/update/"+data[0]+'/temperatura/'+str(int(time.time()))+'/' +temperatura.split()[0]+'/')

            with open("/home/pi/presion") as archivo:
                presion = archivo.read()
                while presion == '':
                    presion = archivo.read()
                sat.actualizar(host,"/update/"+data[0]+'/presion/'+str(int(time.time()))+'/' +presion.split()[0]+'/')

            with open("/home/pi/humedad") as archivo:
                humedad = archivo.read()
                while humedad == '':
                    humedad = archivo.read()
                sat.actualizar(host,"/update/"+data[0]+'/humedad/'+str(int(time.time()))+'/' +humedad.split()[0]+'/')

            with open("/home/pi/precipitacion") as archivo:
                precipitacion = archivo.read()
                while precipitacion == '':
                    precipitacion = archivo.read()

```

```

        sat.actualizar(host, "/update/"+data[0]+'/precipitacion/' + str(int(time.time())) + '/' + precipitacion.split()[0] + '/')

        with open("/home/pi/nivel") as archivo:
            nivel = archivo.read()
            while nivel == '':
                nivel = archivo.read()
            sat.actualizar(host, "/update/"+data[0]+'/nivel/' + str(int(time.time())) + '/' + nivel.split()[0] + '/')

        with open("/home/pi/latitud", 'r') as archivo:
            latitud = archivo.read()
            while latitud == '':
                latitud = archivo.read()

        with open("/home/pi/longitud", 'r') as archivo:
            longitud = archivo.read()
            while longitud == '':
                longitud = archivo.read()

        sat.actualizar(host, "/gps_update/"+data[0]+'/' + latitud.split()[0] + '/' + longitud.split()[0] + '/')

        with open("/home/pi/alerta") as archivo:
            alert = archivo.read()
            while alert == '':
                alert = archivo.read()
            sat.actualizar2(host, "/alerta_update/"+data[0]+'/', {alerta:alert.split()[0]})

        with open('/home/pi/internal_temp', 'r') as archivo:
            temp_interna = archivo.read()
            while temp_interna == '':
                temp_interna = archivo.read()

        temp_interna = int(temp_interna)/1000.00

        with open('/home/pi/bateria', 'r') as archivo:
            bateria = archivo.read()
            while bateria == '':
                bateria = archivo.read()
            bateria = float(bateria)

```

```

        sat.actualizar(host, "/estacion_up-
date/"+data[0]+"/"+str(psutil.cpu_percent())+"/"+str(psutil.vir-
tual_memory().percent)+"/"+'{0:0.2f}'.format(temp_in-
ternã)+"/"+'{0:0.2f}'.format(bateria))

        gsm.stop()
    except:
        gsm.stop()

```

SCRIPT 9. ACTUALIZACIÓN AL SERVIDOR

La actualización de los datos comienza por realizar un autenticación con las credenciales propias de la estación de alerta temprana, sin este registro el envío de datos sería imposible ya que por seguridad el sistema solo permite actualización de datos a estaciones SAT registradas.

El script de actualización lee los archivos de texto plano que contienen los valores de las últimas mediciones realizadas por los sensores, el GPS y la estación y los envía a las url de obtención de datos mediante una petición GET o POST.

Para realizar las peticiones a la página web y las urls de actualización de datos se utiliza la biblioteca de python request, esta biblioteca maneja a alto nivel las peticiones GET y POST para el envío de los datos sin tener que pasar por codificar directamente la petición HTTP POST y HTTP GET.

2.12.2 Codificación ADC- modo medición de tensión.

Para realizar las mediciones de la tensión de la batería de la estación de alerta temprana se utiliza un script que mediante comandos AT realiza la configuración del ADC integrado en la placa 3G.

```

.
.
import serial
import argparse
import time
.
class Analogico:
    def leer(self, modo):

```



```

movil = wcdma()
valor = 0
estado = movil.enviarAT("AT+CADC="+str(modo), "OK", 10)
if estado[0]:
    valor = (estado[1].split()[2])
else:
    print "Error: lectura analógica"
movil.cerrar()
return valor

if __name__=="__main__": # programa principal
.
.
.
    adc = Analogico()
    valor_leido = adc.leer(datos.modo) # lectura del convertidor en modo ten-
sión
.
.
    tension = float(valor_leido)/1000.0 # conversión a voltios
    bateria = (tension*7600.0)/1000.0 # ajuste del circuito de acople
.

```

SCRIPT 10. LECTURA DE TENSIÓN EN BATERIA

El script implementa una clase Analógico, que configura el ADC en el modo que se define por la línea de comandos. El sistema ejecuta el script como un comando del sistema en modo de medición de tensión y el script se encarga de guardar el valor obtenido de la conversión hacia un archivo de texto plano.

2.13 Sistema de posicionamiento global

La ubicación de la estación de alerta temprana se obtiene del módulo GPS que proporciona la ubicación geográfica brindando latitud y longitud, además dado que la placa Raspberry Pi no posee un reloj interno de tiempo real RTC, la estación obtiene la fecha y hora del módulo GPS. Esto es importante ya que cuando la placa Raspberry Pi realiza la actualización de datos, envía la fecha y hora en que ese dato se obtuvo.

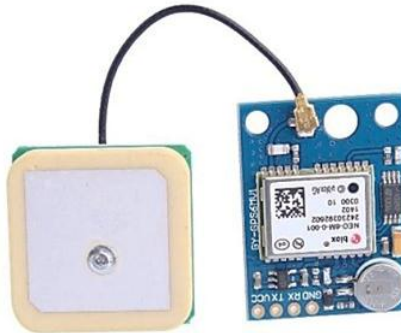


ILUSTRACIÓN 13. GPS- NEO 6M

La estación de alerta temprana utiliza el módulo GPS GY-GPS6MV1 que cuenta con un núcleo ublox NEO -6M. La serie NEO-6 es una familia de receptores GPS autónomos que ofrecen alto rendimiento

2.13.1 Características

- Ultra sensibilidad: -165dBm.
- Receptor de 22 / 66 canales de rastreo y adquisición.
- Soporta estándares WAAS/EGNOS/MSAS/GAGAN.
- Frecuencia de actualización de 1Hz.
- Velocidad de desplazamiento máxima: 500m/seg.
- Protocolo NMEA.
- Puerto serial (9600bps).
- Antena incorporada de 18.2 x 18.2 x 4.0 mm.
- Rango de temperatura: -40 to 85 C.
- Tamaño reducido 30mm x20mm x 11.4mm.

2.13.2 Especificaciones

Tipo de receptor	L1 frequency band, C/A code 22 Tracking / 66 Canales de lectura	
Sensibilidad	Rastreo	-165dBm
	Adquisición	-148dBm

Precisión	Posición Velocidad Tiempo (PPS)	3mts. 3D RMS sin SA 0.1m/s sin SA 60ns RMS
Tiempo de lectura	Cold Start Warm Start Hot Start Re-Acquisition	36s 33s 1s <1s
Consumo de energía	Rastreo Adquisición Sleep/Standby	<30mA @ 3V Vcc 40mA TBD
Frecuencia de actualización de datos de navegación	1Hz	
Límites de operación	Altitud Velocidad Aceleración	Max 18,000m Max 515m/s Menor a 4g
Especificación de la antena	Dimensiones externas Frecuencia Central Ancho de banda Impedancia Rango axial Polarización	18.2 x 18.2 x 4.0 mm 1575 ± 3 MHz 10 MHz min 50 Ω 3 dB máx. RHCP
Dimensiones y peso	Dimensiones Peso	30mm x20mm x 11.4mm 9g
Tensión de alimentación	VCC Corriente	5V ±5% 55mA(típico)
Entorno	Temperatura de operación Temperatura de almacenamiento	40 ~ +85 0 ~ +125

TABLA 15. GPS, ESPECIFICACIONES TÉCNICAS (UBLOX)

2.13.3 Protocolo de comunicación

Se comunica a través de puerto serial UART. Mediante sentencias NMEA. Es una especificación combinada eléctrica y datos entre aparatos electrónicos marinos.

El protocolo NMEA es un medio a través del cual los instrumentos marítimos y la mayoría de los receptores GPS pueden comunicarse los unos con los otros. Estas sentencias NMEA ha sido definidas y están controladas por la organización estadounidense National Marine Electronics Association.

2.13.4 Codificación

Los datos de latitud y longitud geográfica se obtienen utilizando el servicio (daemon) de Linux gpsd. Creando un socket entre el puerto serial ttyAMA0 de la Raspberry y el daemon gpsd. Los datos se capturan en el entorno de python mediante la biblioteca python-gps que se obtiene de repositorios.

```
.
.
.
from gps import * # manejo del GPS

# Esta clase maneja los datos suministrados por el gps
class geo:
    def __init__(self):
        self.gpsd = gps(mode=WATCH_ENABLE) # inicializacion del gps
        for i in range (0,10):
            self.gpsd.next()
.
.
.
    def obtenerLatitud(self): # obtener la latitud geodésica
        self.gpsd.next()
        return str(self.gpsd.fix.latitude)

    def obtenerLongitud(self): # obtener la logitud geodésica
        self.gpsd.next()
        return str(self.gpsd.fix.longitude)
.
.
.
if __name__=="__main__": # programa principal
    neo6M = geo()
    try:
        archivo = open('/home/pi/latitud', 'w')
        archivo.write(neo6M.obtenerLatitud())
        archivo.close()
```

```
    archivo = open('/home/pi/longitud', 'w')
    archivo.write(neo6M.obtenerLongitud())
    archivo.close()
.
.
.
except (KeyboardInterrupt, SystemExit):
    pass
```

SCRIPT 11. LECTURA DEL GPS

La clase geo, se encarga de la iniciación del GPS en modo WATCH_ENABLE para que estos puedan ser capturados, luego de esto se realiza un lazo de 10 iteraciones para obtener datos provenientes del GPS. Las cuatro funciones reúnen los valores de latitud, longitud, altitud y fecha para ser guardadas hacia diferentes archivos de texto plano para posterior análisis y envié hacia el servidor.

2.13.5 NTPd

El demonio Network Time Protocol (ntpd) es un programa de sistema operativo que mantiene la hora del sistema en sincronización con servidores de tiempo utilizando el protocolo de tiempo de red (NTP).

La estación de alerta temprana utiliza el GPS para establecer la hora y fecha del sistema mediante las sentencias NMEA. El demonio ntpd extrae la información necesaria de las sentencias NMEA 0183 y ajusta la hora del sistema en base a la fecha y hora reportada por el GPS. La instalación del servicio ntp se realiza mediante los repositorios disponibles para la placa Raspberry Pi.

2.13.6 Configuración

La configuración del servicio ntp encuentra declarada en el archivo ntp.conf para la escucha de sentencias NMEA 0183.

```
.  
.br/>#GPS  
server 127.127.28.0 minpoll 4  
fudge 127.127.28.0 time1 0.183 refid NMEA  
server 127.127.28.1 minpoll 4 prefer  
fudge 127.127.28.1 refid PPS  
.br/>.
```

SCRIPT 12. NTP.CONF

En la sección GPS las 4 líneas establecen la configuración necesaria para la escucha de las sentencias enviadas por ntpd. El GPS utiliza un protocolo de comunicación UART y el servicio de ntp utiliza protocolos de Internet por lo que es necesario establecer un socket de conexión entre el puerto serial de la placa Raspberry Pi y el demonio gpsd para que el sistema arranque al inicio con la hora.

En el archivo rc.local se encuentra establecido el socket de conexión para que arranque con el sistema.

```
#!/bin/sh -e  
.br/>.br/>.br/># socket de conexión para el puerto serie ttyAMA0 y el demonio gpsd  
sudo gpsd /dev/ttyAMA0 -F /var/run/gpsd.sock  
exit 0
```

SCRIPT 13. RC.LOCAL

2.14 Diseño PCB

El PCB de la estación de alerta temprana es una interfaz de comunicación con todos los elementos de la estación. Los elementos se montan en un header específico para cada dispositivo.

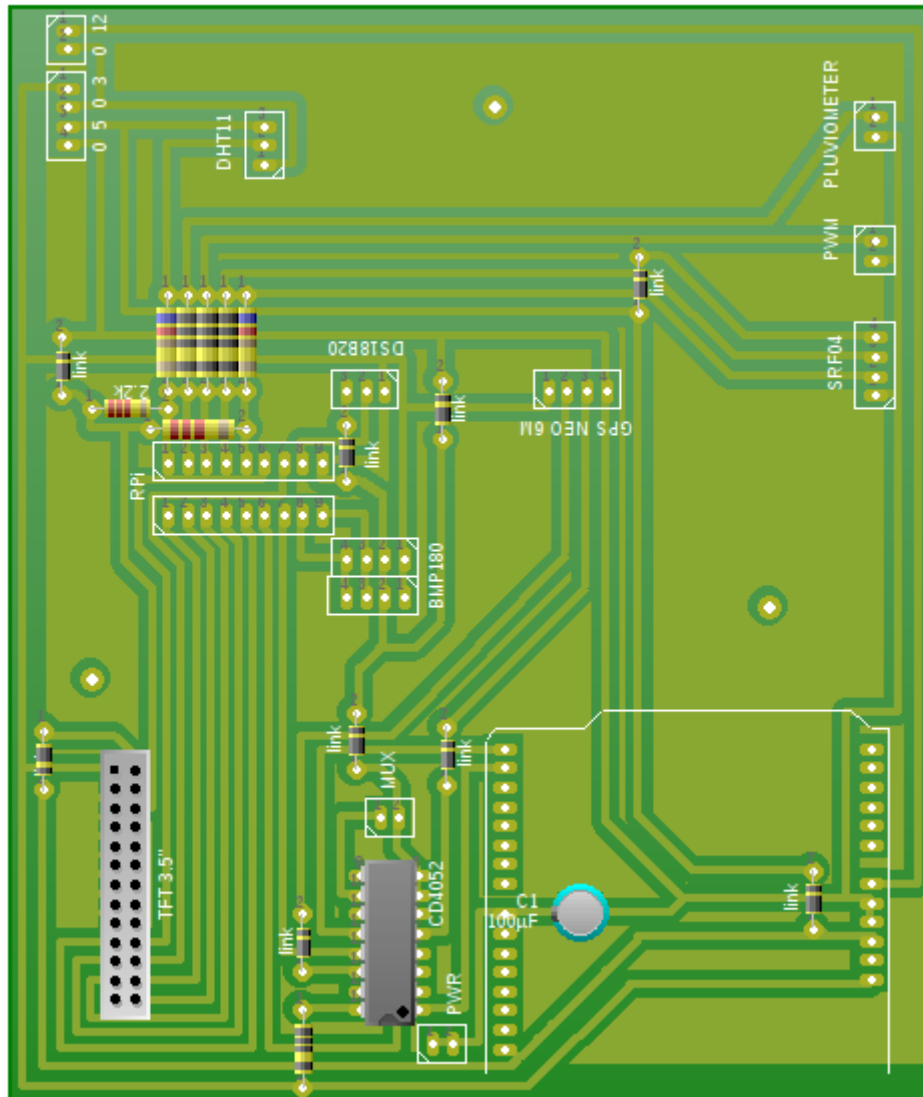


ILUSTRACIÓN 14. PCB BASE ESTACIÓN SAT - LADO DE COMPONENTES

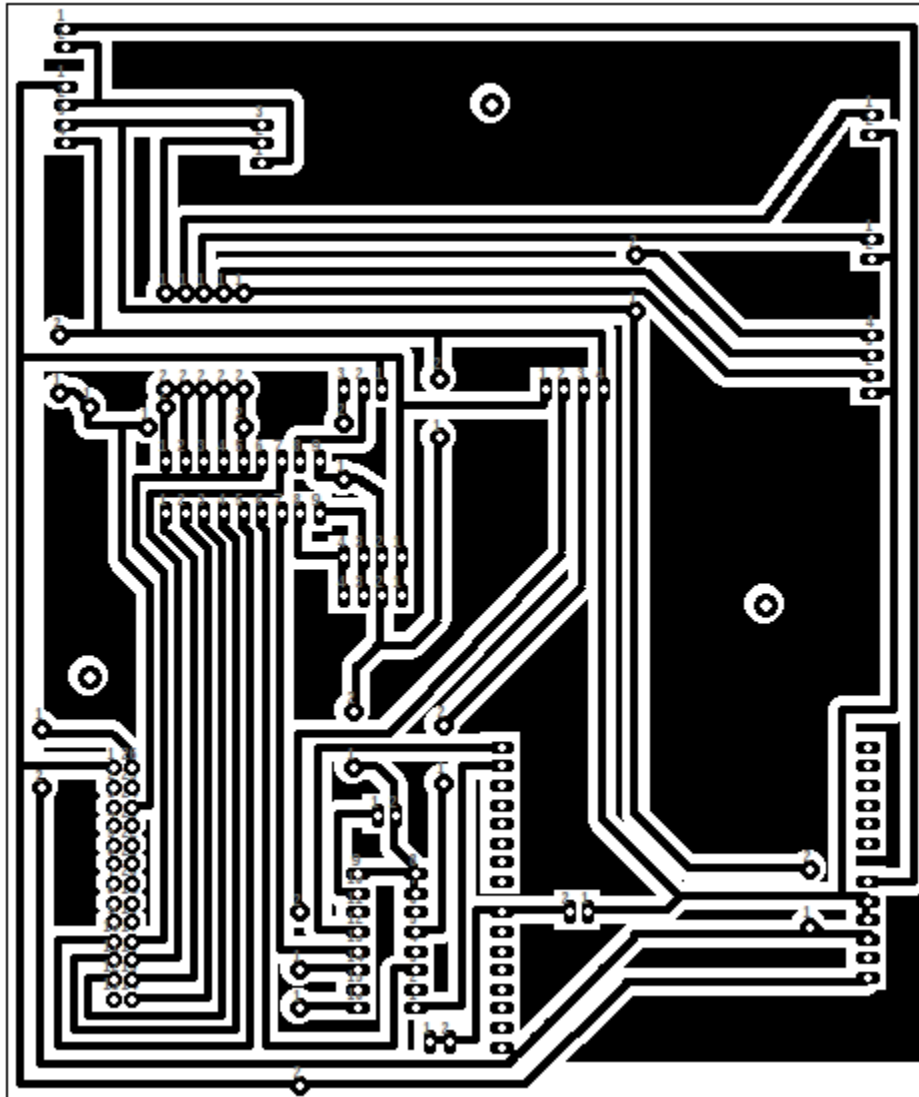


ILUSTRACIÓN 15. PCB BASE ESTACIÓN SAT - LADO DE PISTAS

Adicionalmente dos PCBs mas realizan la función de fuente de alimentación y driver de motor – Raspberry Pi.

La fuente de alimentación reduce la tensión de la batería de alimentación de 12 voltios a 5.0 voltios y 3.0 voltios para la alimentación de los dispositivos y la Raspberry Pi.

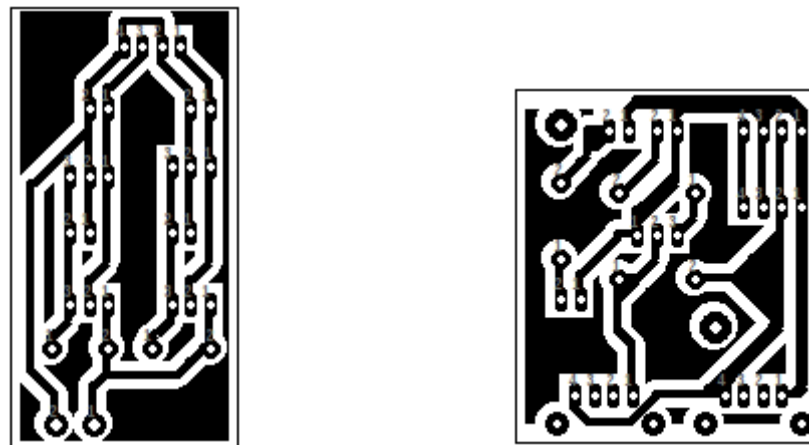
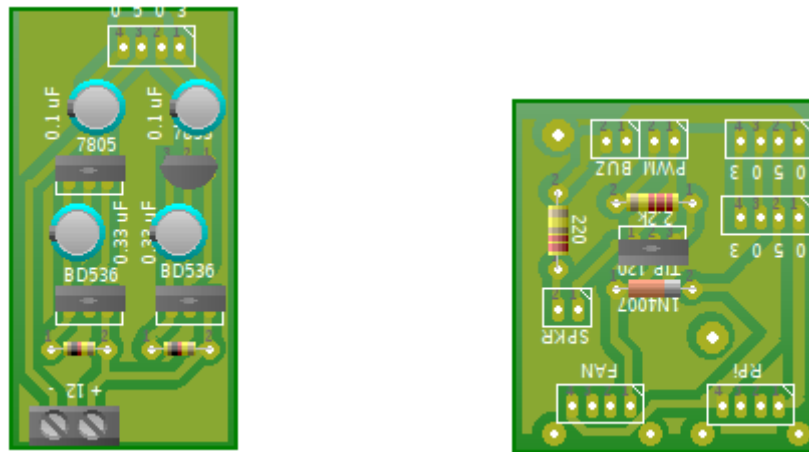


ILUSTRACIÓN 16. DERECHA DRIVER DE MOTOR, IZQUIERDA FUENTE DE ALIMENTACIÓN

La cuarta PCB es un circuito de acople para el convertidor ADC de 12 bits de la placa 3G. Sirve para crear una ampliación de escala ya que el convertidor ADC está programado en modo de medición de tensión.



ILUSTRACIÓN 17. CIRCUITO DE ACOPLE CONVERTIDOR ADC

2.15 Integración del sistema.

2.15.1 CRON

Cada script presentado hasta acá por sí mismo no se ejecuta en la estación de alerta temprana. En Linux el controlador que realiza la ejecución de tareas automatizadas es un servicio denominado cron. Es un administrador regular de procesos en segundo plano que ejecuta procesos a intervalos regulares (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero crontab

Cron es impulsado por un crond, un archivo de configuración que especifica comandos shell para ejecutarse periódicamente a una hora específica. Los archivos crontab son almacenados en donde permanecen las listas de trabajos y otras instrucciones para el demonio cron

Cada línea de un archivo crontab representa un trabajo y es compuesto por una expresión CRON, seguida por un comando shell para ejecutarse.

2.15.2 Formato del fichero crontab

```

.----- minuto (0 - 59)
| .----- hora (0 - 23)
| | .----- día del mes (1 - 31)
| | | .----- mes (1 - 12)
| | | | .---- día de la semana (0 - 6) (Domingo = 0 ó 7)
| | | | |
* * * * * comando a ejecutar

```

SCRIPT 14. FORMATO CRONTAB

2.15.3 Definición de horarios predefinidos

Hay varios valores predefinidos que se pueden utilizar para sustituir la expresión CRON.

Entrada	Descripción	Equivalente
@yearly	Se ejecuta una vez al año	0 0 1 1 *
@annually	(igual que @yearly)	0 0 1 1 *
@monthly	Se ejecuta una vez al mes	0 0 1 * *
@weekly	Se ejecuta una vez a la semana	0 0 * * 0
@daily	Se ejecuta una vez al día	0 0 * * *
@midnight	(igual que @daily)	0 0 * * *
@hourly	Se ejecuta una vez cada hora	0 * * * *

TABLA 16. CRONTAB, HORARIOS PREDEFINIDOS (UBUNTU DOCUMENTATION, S.F.)

También está disponible @reboot, que permite a un trabajo ejecutarse una vez cada vez que el demonio cron se inicie, en el arranque de la estación de alerta temprana. Por lo tanto el sistema de alerta temprana carga todos sus script al momento de iniciar el sistema operativo.

El archivo crontab de la estación de alerta temprana tiene definidos todos los script del sistema de monitoreo de sensores, actualización al servidor, localización GPS, entre otros script de soporte para la estación de alerta temprana.

Todos definidos para intervalos regulares de tiempo, esta es una de las ventajas de utilizar sistemas a microprocesador que permiten la gestión de las aplicaciones de forma nativa.

2.15.4 Configuración crontab

A continuación se muestra el fichero crontab de la estación de alerta temprana.

```
# BEEP
@reboot /home/pi/SAT/buzzer.py -v 1 -t 500
@reboot /home/pi/SAT/apagar-reiniciar.py
#
# SENSORES
# Actualización de la temperatura
* * * * * /home/pi/SAT/temperatura.py
#
# Actualización de la presión barométrica
*/2 * * * * /home/pi/SAT/presion.py
#
# Actualización de la humedad
* * * * * /home/pi/SAT/humedad.py
#
# Actualización del nivel del rio
* * * * * /home/pi/SAT/sonar.py
#
# Actualización de la precipitacion
@reboot /home/pi/SAT/pluviometro.py
#
# Actualización de temperatura interna y frecuencia de la CPU
* * * * * /home/pi/SAT/interna.py
#
# Actualización GPS latitud, longitud, altitud
*/10 * * * * /home/pi/SAT/geolocalizacion.py
#
# Sistema de enfriamiento
*/5 * * * * /home/pi/SAT/cooler.py
#
# Inicialización GSM 3G
@reboot /home/pi/SAT/iniciarGSM.py
```

SCRIPT 15. ARCHIVO CRONTAB

2.15.5 Interrelación de scripts

El diagrama de interrelación de los scripts del sistema se muestra en el siguiente ilustración

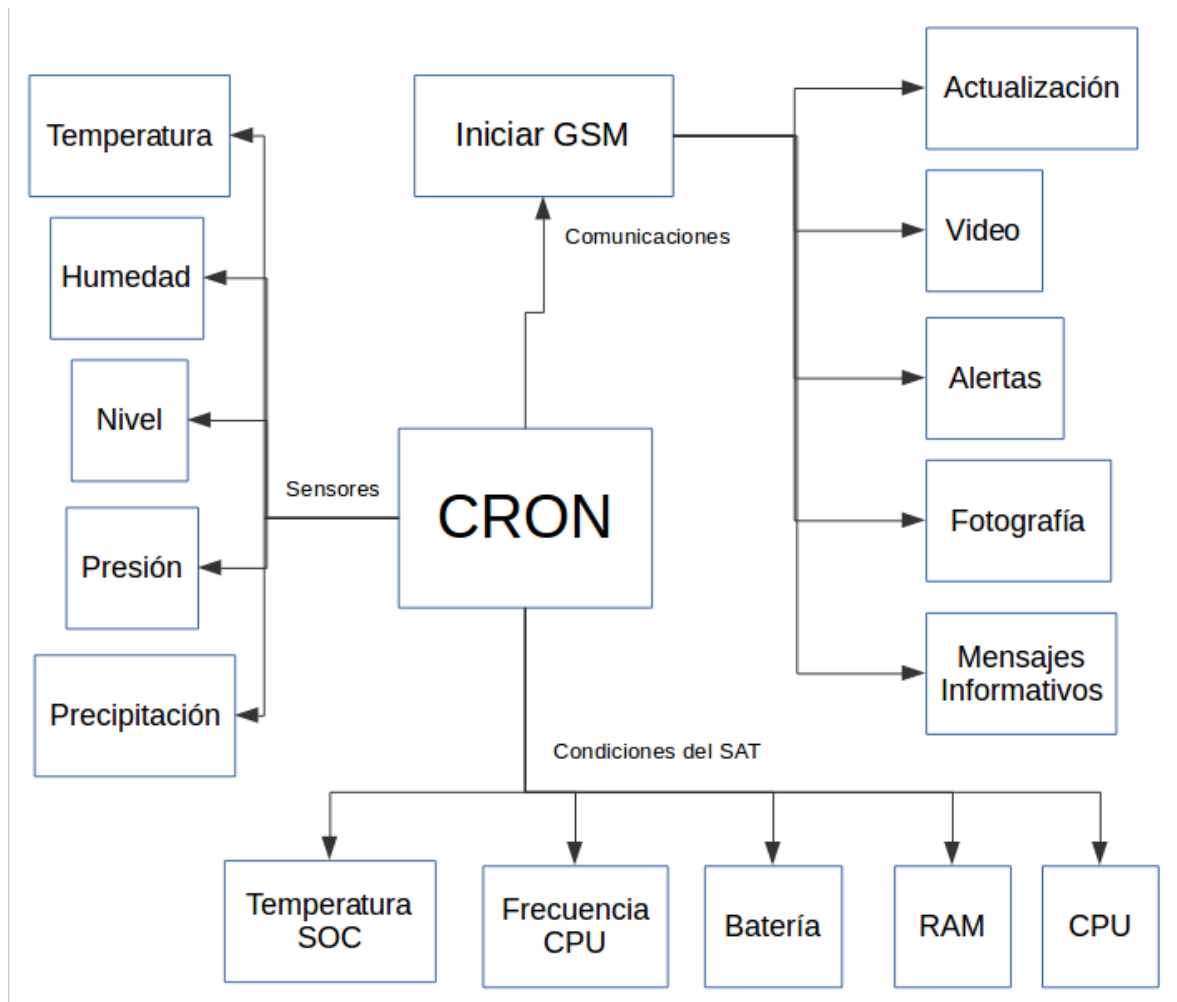


ILUSTRACIÓN 18. INTERRELACIÓN DE SCRIPTS DEL SISTEMA

Los procesos de lectura de sensores y condiciones del SAT son directamente controlados por CRON mediante el archivo crontab. El módulo 3G es controlado por CRON, el envío de alertas, mensajes de consulta, actualización, toma de fotos y video son controlados por el script de inicialización del módulo 3G puesto que este necesita establecer una secuencia de utilización de sus recursos.

2.15.6 Diagrama de bloques del SAT

El diagrama de bloques de la estación de alerta temprana, muestra de forma simplificada. Cada bloque representa un módulo o conjunto de dispositivos que efectúan tareas similares.

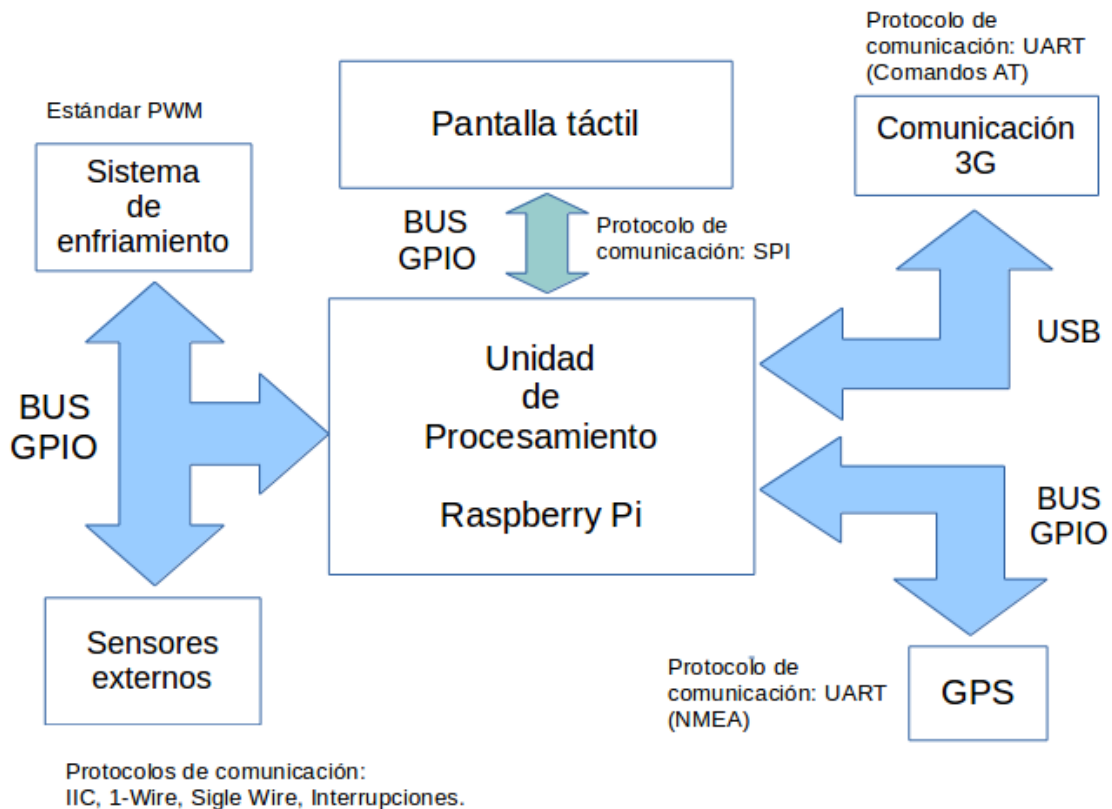


ILUSTRACIÓN 19. DIAGRAMA DE BLOQUES ESTACION SAT

El objetivo de este diagrama es mostrar de manera clara la interconexión de los diferentes dispositivos con los que la estación de alerta temprana cuenta.

Capítulo 3

Servidor WEB y Base de Datos

3.1 Ubuntu server

3.1.1 Introducción

Ubuntu server es un sistema operativo basado en Linux que cuenta con la filosofía de software libre, es un sistema multiusuario sin interfaz gráfica destinado al manejo de aplicaciones cliente servidor.

Cada seis meses se publica una nueva versión de Ubuntu. Esta recibe soporte por parte de Canonical durante nueve meses por medio de actualizaciones de seguridad, parches para bugs críticos y actualizaciones menores de programas. Las versiones LTS (Long Term Support), que se liberan cada dos años, reciben soporte durante cinco años.

La base de datos de las condiciones registradas por la estación de alerta temprana y la página web de motorización de la estación; están hospedadas en una versión de sistema operativo Linux distribución Ubuntu server 12.04 LTS.

3.1.2 Características Ubuntu Server

- Soporte de larga duración (LTS o Long Term Support) durante 5 años
- Incluye a Essex, la última versión de OpenStack, una plataforma para crear servicios en la nube.
- MAAS (Metal As Service) una utilidad para configurar el hardware donde va a desplegarse cualquier servicio que necesite configuración y escalabilidad de forma dinámica
- AWESOME (Any Web Service Over Me) un API o interfaz de programación que facilita el despliegue y administración de servicios tanto en nubes de Amazon como de OpenStack
- Soporte de Java con OpenJDK 7 (muy cercano al JDK 7 oficial de Oracle)
- Apache ActiveMQ 5.5, el servidor más popular de código abierto de mensajería y patrones de integración
- Juju, una herramienta para facilitar los servicios en la nube
- Servidor de integración continua Jenkins 1.424.6

- Servidor de aplicaciones Apache Tomcat 7.0.26
- Los lenguajes de programación Groovy, Clojure y Scala soportados sobre el OpenJDK
- La virtualización incluye Xen, KVM y LXC
- El mínimo requerimiento de memoria es de 128 MB
- La imagen de 32 bits puede llegar a un máximo de 16 GB de memoria y utilizar un procesador de hasta 8 núcleos
- Está disponible sobre arquitectura Intel x86 (32 bits), Intel AMD64 (64 bits) y también para procesadores ARM (existen 4 imágenes para ARM)

3.1.3 Requisitos Ubuntu Server

Los requisitos para una versión server Linux son mínimos debido a que no utiliza el entorno gráfico, pero para que actúe como servidor dependiendo del tráfico a la página web y datos en la base de datos que se tengan puede requerir más, dado que se monitoriza datos en tiempo real.

El mínimo de memoria requerido para Ubuntu Server 12.04 es 128 MB. La imagen i386 para 32 bits tiene un máximo recomendado de 16 GB en RAM, y un máximo de 8 núcleos de CPU.

3.2 Django

3.2.1 Introducción

Django es un framework de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como Modelo–Vista–Controlador actualmente se encuentra disponible la versión 1.8.

El diseño modelo-vista-controlador (MVC) se distingue por separar las partes involucradas en la creación de código y lograr así reutilizar las partes de código genéricas para reducir la codificación necesaria en la creación de un sitio web.

3.2.2 Descripción

Django divide la creación de sitios web básicamente en 4 partes fundamentales para cumplir el modelo MVC, estas partes son, los modelos, vistas, urls y las plantillas.

Los modelos son las líneas de código python necesarias para definir un objeto por ejemplo para la estación de alerta temprana, solo se necesita delimitar que es una estación de alerta temprana y con que dispositivos cuenta y así se definirá su modelo. En tal caso se puede definir la estación SAT como poseedora de un nombre, una ubicación geográfica, un número determinado de sensores y actuadores, etc. y utilizar el mismo modelo para definir otras estaciones de alerta temprana, y de la misma forma definir un modelo para los sensores y así utilizar este para todos los sensores con que cuenta cada estación de alerta temprana y relacionarlos entre sí.

Las vistas es el código puro que maneja tras de cámaras lo que se le presenta al usuario al solicitar una página web, es decir en las vistas se define lo que ve el usuario o lo que se le ha permitido ver según criterios de selección y permisos asignados. Es también código python y cada vista define por sí misma una página web de un sitio o una petición.

Las urls son la definición de la ruta por la que alguna vista será llamada a generar una página. Cada url está asignada a una vista y son estas las que se muestran en el navegador o son por ejemplo urls de actualización de datos que no tienen por objetivo definir una página web sino coleccionar datos o también proporcionar datos al usuario en otro formato que no es el HTML sino por ejemplo descargar la base de datos de la temperatura en un periodo de tiempo dado. Las urls también son código python

Las plantillas son el lienzo por el cual las vistas van a generar una página web, estas son el código HTML base para generar la página web total y por lo tanto pueden ser utilizadas por muchas vistas para generar distintas páginas web en base a un mismo HTML.

3.2.3 Instalación

Django está instalado globalmente en el sistema operativo, no se ha hecho uso de entornos virtuales o versiones de desarrollo, es versión de producción que abarca todo el sistema operativo y utiliza Apache 2 como servidor HTTP. La instalación se realiza desde los repositorios de Ubuntu, los paquetes instalados son python-django, python-pip apache2 y libapache2-mod-wsgi

Libapache2-mod-wsgi sirve de puente entre la aplicación desarrollada con Django y el servidor HTTP Apache. El paquete `mod_wsgi` implementa un módulo de Apache que puede albergar cualquier aplicación web de Python que soporta la especificación de Python WSG.

3.2.4 Aplicación en Django

La aplicación de Django esta creada dentro de un proyecto de Django que se creó en el directorio donde Apache almacena sus archivos.

```
alerta/  
  __init__.py  
  manage.py  
  settings.py  
  urls.py
```

Estos archivos son:

- **__init__.py**: Un archivo vacío que le dice a Python que este directorio debería ser considerado un paquete Python.
- **manage.py**: Una utilidad de línea de comandos que permite interaccionar de distintas formas con este proyecto Django.
- **settings.py**: Configuraciones del proyecto.
- **urls.py**: Las URLs para este proyecto Django; una "tabla de contenidos" del sitio web

Dentro del proyecto está creada la aplicación `alertaApp`

```
alertaApp/  
  __init__.py  
  models.py  
  test.py  
  views.py
```

La aplicación creada se instala en el archivo de configuración del proyecto `alerta`.

```

INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'alertaApp',
)

```

3.2.5 Archivos de aplicación y configuración.

Settings.py

El archivo de configuración del proyecto contiene toda la información de la configuración del sitio.

```

"""
Django settings for alerta project.

Generated by 'django-admin startproject' using Django 1.8.

For more information on this file, see
https://docs.djangoproject.com/en/1.8/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/1.8/ref/settings/
"""

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
import os

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/1.8/howto/deployment/checklist/

```

```

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'c&*s+pmc#-^r%td(%&ve$0j^s3tiw&k((w3vov!lw^#x+=+iiqt'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = False

ALLOWED_HOSTS = ['localhost', 'alertatem-
prana.ddns.net', '127.0.0.1', '10.1.1.101', '190.5.157.129']

# Application definition

INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'alertaApp',
)

MIDDLEWARE_CLASSES = (
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'django.middleware.security.SecurityMiddleware',
)

ROOT_URLCONF = 'alerta.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(os.path.dirname(__file__), 'plantillas')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
            ]
        }
    }
]

```

```

        'django.contrib.auth.context_processors.auth',
        'django.contrib.messages.context_processors.messages',
    ],
},
},
]

WSGI_APPLICATION = 'alerta.wsgi.application'

# Database
# https://docs.djangoproject.com/en/1.8/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'alerta',
        'USER': 'root',
        'PASSWORD': 'osegueda',
        'HOST': '127.0.0.1',
        'PORT': '3306',
    }
}

# Internationalization
# https://docs.djangoproject.com/en/1.8/topics/i18n/

LANGUAGE_CODE = 'es'

TIME_ZONE = 'America/El_Salvador'

USE_I18N = True

USE_L10N = False

USE_TZ = True

# ADMINISTRADORES DEL SITIO

ADMINS = (('Rigoberto Osegueda', 'rigoberto.ues@gmail.com'),)

# CONFIGURACION DE EMAIL

EMAIL_BACKEND='django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST='smtp.gmail.com'

```

```

# encriptamiento
#EMAIL_PORT=587 # puerto para tls
#EMAIL_USE_TLS=True

EMAIL_PORT=465 # puerto para ssl
EMAIL_USE_SSL=True

# credenciales
EMAIL_HOST_USER='sistema.alerta.temprana.sat@gmail.com'
EMAIL_HOST_PASSWORD='alertatemprana'

DEFAULT_FROM_EMAIL='webmaster@alertatemprana'
SERVER_EMAIL='root@alertatemprana'

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.8/howto/static-files/

STATIC_URL = '/static/'
STATICFILES_DIRS = STATIC_URL = '/static/'
STATICFILES_DIRS = os.path.join(os.path.dirname(__file__), 'static'),

LOGIN_REDIRECT_URL = '/home/lempa/'
LOGIN_URL = '/login/'
LOGOUT_REDIRECT_URL = '/bloquear/'

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(os.path.dirname(__file__), 'media')

STATIC_ROOT = '/static/'

```

SCRIPT 16. ARCHIVO DE CONFIGURACION DE DJANGO

Lo más importante de este archivo de configuración es la definición de la base de datos, la definición de las rutas de la jerarquía creada en el proyecto.

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'alerta',

```

```

        'USER': 'root',
        'PASSWORD': 'osegueda',
        'HOST': '127.0.0.1',
        'PORT': '3306',
    }
}

```

Utiliza el propio motor de Django para el manejo de las conexiones hacia la base de datos en MySQL. El host indica que la base de datos se ubica en el mismo servidor de la pagina web 127.0.0.1 y el puerto de comunicación con la base de datos es el 3306.

La definición del directorio padre que define las rutas relativas hacia los demás archivos del sitio.

```

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

```

La ruta de archivos estáticos y archivos media:

```

STATICFILES_DIRS = os.path.join(os.path.dirname(__file__), 'static'),
MEDIA_ROOT = os.path.join(os.path.dirname(__file__), 'media')

```

Models.py

Este archivo define la estación de alerta temprana con sus dispositivos y capacidades. Solo se muestra la definición de la clase para la Estación.

```

from django.db import models
from django.contrib.auth.models import User
.
.
.
# Modelo que describe las estaciones de alerta temprana
class Estacion(models.Model): # clase para el manejo de estaciones (múltiples)
    nombre = models.CharField(max_length = 30) # nombre de la estación
    coordenadas = models.CharField(max_length = 30) # coordenadas del GPS
    estado = models.BooleanField(default = True) # activo o desactivado

```

```

    correo = models.EmailField() # email de la estación.
    descripcion = models.TextField() # breve descripción de las condiciones de
    la estación
    alerta = models.TextField(blank = True) # condición de la alerta: verde,
    amarilla, roja, naranja
    imagen = models.ImageField(upload_to = 'Imagen_Dispositivos', blank =
    True) # fotografía de la estación
    ip = models.GenericIPAddressField() # ip address
    telefono = models.CharField(max_length = 8) # telefono del chip
    bateria = models.DecimalField(max_digits = 4, decimal_places = 2) # ten-
    sión de operación
    temperatura = models.DecimalField(max_digits = 5, decimal_places = 2) #
    tensión de operación
    ram = models.DecimalField(max_digits = 5, decimal_places = 2) # tensión de
    operación
    cpu = models.DecimalField(max_digits = 5, decimal_places = 2) # tensión de
    operación
    url = models.URLField(blank = True) # url de conexión

    class Meta:
        ordering = ['nombre'] # ordenamiento requerido a la hora de obtener
        desde la base de datos

    def __str__(self): # referencia de modelo
        return "Estacion: %s, Ubicacion: %s, Estado: %s" %(self.nombre,
        self.coordenadas, self.estado)

```

SCRIPT 17. MODELOS ESTACION SAT

La definición de la alerta temprana se hace mediante la creación de una clase denominada Estación y recibe como argumento la clase models que define el tipo de datos que soporta Django para la creación de modelos.

Las variables utilizadas para la definición del modelo son:

- **nombre:** define el nombre de la estación de alerta temprana
- **coordenadas:** la ubicación geográfica de la estación de alerta temprana
- **estado:** si esta activa o fuera de línea
- **correo:** dirección de correo electrónico de la estación de alerta temprana

- descripción: cadena de texto que muestra una referencia de la estación de alerta temprana.
- **alerta:** la condición de alerta ultima registrada
- **imagen:** fotografía de la estación de alerta temprana
- **IP: dirección** ip asignada en el momento de la conexión
- **teléfono:** número telefónico de la estación determinado por el chip de la placa 3G
- **batería:** tensión de la batería de alimentación de la estación
- **temperatura:** temperatura del microprocesador de la estación de alerta temprana
- **RAM:** memoria utilizada por los procesos activos de la estación de alerta temprana
- **CPU:** porcentaje de utilización de la CPU del microprocesador
- **url:** url de referencia con la página web.

Cada variable se define como un tipo de datos que pueden ser tipo cadenas de texto, números enteros, direcciones de correo, direcciones IP, archivos de tipo imagen, números de coma flotante. Cada vez que se requiera de crear una nueva estación de alerta temprana se utilizará este modelo.

La clase Meta es parte de la definición de la estación y se utiliza para ordenar los resultados de búsqueda en el caso que haya más estaciones de alerta temprana, estas se ordenarán alfabéticamente según el nombre de cada estación.

View.py

El archivo de vistas es el archivo base de la creación de las páginas web del sitio en conjunto con las plantillas. Solo se muestra la función que maneja la creación de la página de inicio home.

```
#_*_ coding: utf-8 *_*  
  
from django.contrib.auth.models import User # Manejo de usuarios  
from django.core import mail # Envío de mensajes de correo desde los usuarios staff  
from django.core.urlresolvers import reverse # dirección anterior  
from django.http.response import HttpResponseRedirect, HttpResponse # valor devuelto a la petición  
from django.shortcuts import render # Contextualización de plantillas  
from django.template.context import RequestContext # solicitar contexto tal como la sesión
```

```

from django.contrib.auth.decorators import login_required # decorador para
proteger paginas
from forms import UserForm, PerfilForm, EventoForm # formularios que utilizo
para enlazar con modelos
from models import Dispositivo, Estacion, Condicion, Perfil, Evento, Video,
Fotografia # modelos de la estación de alerta temprana
from django.db.models import Max, Min, Avg # máximo, mínimo y promedio de la
serie de valores de los sensores en base de datos
from django.http import JsonResponse, StreamingHttpResponse # JavaScript ob-
ject notation y Streaming de files
import time, psutil, pytz, datetime, csv, os # tiempo, sistema, timezone, fe-
chas y csv
import gspread # Manejo de datos en googleDocs spreadsheet
from django.core.files.storage import default_storage
from django.core.files.base import ContentFile
from django.conf import settings
from random import randrange
from tweets import get_tweets, post_tweet
#####
##### --
def home(request, contexto): # página de inicio por ejemplo http://www.servi-
dor.com/home/lempa/

    estaciones = Estacion.objects.all() # todas las estaciones
    try:
        estación = estaciones.get(nombre=contexto) # la estación seleccionada
    except estaciones.model.DoesNotExist:
        return HttpResponseRedirect('/admin/')

    fotografias = estacion.fotografia_set.all() # todas las fotografías
    ultima_fotografia = fotografias.latest() # ultima fotografía capturada
    demas_fotografias = fotografias.exclude(fotografia=ultima_fotografia.foto-
grafia) # todas las fotografías menos la ultima

    videos = estacion.video_set.all() # todas las fotografías
    ultimo_video = videos.latest() # ultima fotografía capturada
    demas_videos = videos.exclude(video=ultimo_video.video) # todos los videos
menos el ultimo

    dispositivos = estacion.dispositivo_set.all() # todos los dispositivos re-
gistrados de la estación seleccionada

    try:
        precipitacion = dispositivos.get(nombre='precipitacion').condi-
cion_set.latest().valor # ultimo valor registrado en la base de datos
    except:

```

```

    precipitacion = 0.0
    try:
        presión = dispositivos.get(nombre='presión').condicion_set.latest().valor # ultimo valor registrado en la base de datos
    except:
        presión = 0.0
    try:
        humedad = dispositivos.get(nombre='humedad').condicion_set.latest().valor # ultimo valor registrado en la base de datos
    except:
        humedad = 0.0
    try:
        temperatura = dispositivos.get(nombre='temperatura').condicion_set.latest().valor # ultimo valor registrado en la base de datos
    except:
        temperatura = 0.0
    try:
        tweets = get_tweets()
    except:
        tweets = []

    return render(request, 'home.html', {
        'user': request.user,
        'estacion': estación,
        'estaciones': estaciones,
        'precipitacion': precipitacion,
        'presion': presión,
        'humedad': humedad,
        'temperatura': temperatura,
        'dispositivos': dispositivos,
        'tweets': tweets,
        'fotos': demas_fotografias,
        'ultima_fotografia': ultima_fotografia,
        'ultimo_video': ultimo_video
    })

# --
#####
##### --

```

SCRIPT 18. DEFINICION DE VISTAS

Dada la complejidad del archivo solo se muestra la definición de la página web de inicio. Cada página web se define como una función en python que recibe un contexto y un request, el contexto tiene como propósito capturar la ruta especificada en la url y el request contiene información del tipo de petición que se realiza a la página web. Esto se utiliza ya que esta misma función define la página web home de cualquiera otra estación de alerta temprana que se crea, por lo tanto en el contexto se determina a que estación se le está solicitando la página de home para que esta función puede pedir a la base de datos los valores correspondientes a la estación solicitada. Luego de haber extraído toda la información de la base de datos se procede a generar la respuesta hacia el usuario que la haya solicitado con la función render. Esta función toma el archivo base.html y lo retorna con una serie de valores tomados de la base de datos para que estos sean presentados al navegador.

Home.html

```
{% extends "base.html" %}

{% load static %}

{%block title%} | Home{%endblock %}

{% block titulo_contenido %}
    <h1>Estación {{ estacion.nombre }}<small>Sistema de alerta temprana</small></h1>
{% endblock %}

{% block ruta %}
    <li><a href="/home/{{ estacion.nombre | lower }}"><i class="fa fa-home"></i> Home</a></li>
{% endblock %}

{% block contenido %}
    <section class="content">
        <div class="row">
<!--
#####
#####-->
            <div class="col-lg-3 col-xs-6">
                <div class="small-box bg-aqua">
                    <div class="inner">
```

```

        <h3>{{ precipitación | escape }}<sup
style="font-size: 20px"> mm</sup></h3>
        <p>Lluvia</p>
    </div>
    <div class="icon">
        <i class="fa fa-umbrella"></i>
    </div>
    <a href="/graficas/{{ estacion.nombre | lower }}/pre-
cipitacion/" class="small-box-footer">
        Más info <i class="fa fa-arrow-circle-
right"></i>
    </a>
</div>
</div>
<div class="col-lg-3 col-xs-6">
    <div class="small-box bg-green">
        <div class="inner">
            <h3>{{ presión | escape }}<sup style="font-
size: 20px"> Pa</sup></h3>
            <p>Presión</p>
        </div>
        <div class="icon">
            <i class="fa fa-cloud"></i>
        </div>
        <a href="/graficas/{{ estacion.nombre | lower }}/pre-
sion/" class="small-box-footer">
            Más info <i class="fa fa-arrow-circle-
right"></i>
        </a>
    </div>
</div>
<div class="col-lg-3 col-xs-6">
    <div class="small-box bg-yellow">
        <div class="inner">
            <h3>{{ humedad | escape }}<sup style="font-
size: 20px"> %</sup></h3>
            <p>Humedad relativa</p>
        </div>
        <div class="icon">
            <i class="fa fa-refresh"></i>
        </div>
        <a href="/graficas/{{ estacion.nombre | lower
}}/humedad/" class="small-box-footer">
            Más info <i class="fa fa-arrow-circle-
right"></i>
        </a>
    </div>
</div>

```

```

        </a>
        </div>
    </div>
    <div class="col-lg-3 col-xs-6">
        <div class="small-box bg-red">
            <div class="inner">
                <h3>{{ temperatura | escape }}<sup style="font-size: 20px"> C</sup></h3>
                <p>Temperatura</p>
            </div>
            <div class="icon">
                <i class="fa fa-fire"></i>
            </div>
            <a href="/graficas/{{ estacion.nombre | lower }}/temperatura/" class="small-box-footer">
                Más info <i class="fa fa-arrow-circle-right"></i>
            </a>
        </div>
    </div>
</div>
<!--
#####
#####-->
<!-- Carousel -->
    <div class="row">
        <div class="col-xs-8">
            <div class="box box-solid box-danger">
                <div class="box-header with-border">
                    <h3 class="box-title">Fotogaleria</h3>
                </div>
                <div class="box-body">
                    <div id="carousel-example-generic" class="carousel slide" data-ride="carousel">
                        <div class="carousel-inner">
                            <div class="item active">
                                
                                <div class="carousel-caption">
                                    {{ ultima_fotografia.fecha_hora
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
    {% for foto in fotos %}

```

```

        <div class="item" >
            
            <div class="carousel-caption">
                {{ foto.fecha_hora }}
            </div>
        </div>
    {% endfor %}
</div>
<a class="left carousel-control"
href="#carousel-example-generic" data-slide="prev">
    <span class="fa fa-angle-
left"></span>
</a>
<a class="right carousel-control"
href="#carousel-example-generic" data-slide="next">
    <span class="fa fa-angle-
right"></span>
</a>
</div>
</div>
</div>
</div>
<!--
#####
#####-->
    <div class="col-md-4">
        <div class="box box-info box-solid">
            <div class="box-header with-border">
                <h3 class="box-title">Twitter</h3>
                <div class="box-tools pull-right">
                    <button class="btn btn-box-tool" data-
widget="collapse" data-toggle="tooltip" title="Ocultar"><i class="fa fa-mi-
nus"></i></button>
                </div>
            </div>
            <div class="box-body">
                <div id="scroll" class="block social-block so-
cial-twitter">
                    {% load twitter_tags %}
                    <ul>
                        {% for tweet in tweets %}
                            <li>
                                <div class="tweet-meta">
                                    <a href="https://www.twitter.com/{{
tweet.user.screen_name }}/status/{{ tweet.id }}" target="_blank">

```

```

                                
                                <h4>{{ tweet.user.name
}}<br /><span>@{{ tweet.user.screen_name }} &middot; {{ tweet.cre-
ated_at|twitter_date }}</span></h2>
                                </a>
                                </div>
                                <p>{{ tweet|expand_tweet_urls|url-
ize_tweet_text|safe }}</p>
                                </li>
                                {% endfor %}
                                </ul>
                                </div>
                                </div>
                                <div class="box-footer">
                                <a class="btn btn-block btn-social btn-twitter"
href="https://www.twitter.com/{{ tweet.user.screen_name }}" target="_blank">
                                <i class="fa fa-twitter"></i> Siguenos en Twitter
                                </a>
                                </div>
                                </div>
                                </div>
                                </div>
                                </div>
                                <!--
#####
#####-->
                                <!-- Video -->
                                <div class="row">
                                <div class="col-md-8">
                                <div class="box box-primary box-solid">
                                <div class="box-header with-border">
                                <h3 class="box-title">Ultimo video</h3>
                                <div class="box-tools pull-right">
                                <button class="btn btn-box-tool" data-
widget="collapse" data-toggle="tooltip" title="Ocultar"><i class="fa fa-mi-
nus"></i></button>
                                </div>
                                </div>
                                </div>
                                <div class="box-body">
                                <div class="embed-responsive embed-responsive-16by9">
                                <video src="{{ ultimo_video.video.url }}" con-
trols></video>
                                </div>
                                </div>
                                <div class="box-footer">

```



```

                Fecha :
                <span class="label label-default">{{ ul-
timo_video.fecha_hora }}</span>
                </div>
            </div>
        </div>
    </div>
<!--
#####
#####-->
<!-- Mini mapa -->

        <div class="col-md-4">
            <div class="box box-success box-solid">
                <div class="box-header with-border">
                    <h3 class="box-title">{{ estacion.descripcion
}}</h3>

                    <div class="box-tools pull-right">
                        <button class="btn btn-box-tool" data-
widget="collapse" data-toggle="tooltip" title="Ocultar"><i class="fa fa-mi-
nus"></i></button>

                    </div>
                </div>
                <div class="box-body">
                    <div style="height: 360px; background-color:
rgb(229, 227, 223);" id="map"></div>
                </div>
                <div class="box-footer">
                    Coordinadas :
                    <span class="label label-default">{{ esta-
cion.coordenadas }}</span>
                </div>
            </div>
        </div>
    </div>
</section>
{% endblock %}
<!--
#####
#####-->

```

SCRIPT 19. PLANTILLA DE PÁGINA HOME

El archivo `home.html` define el esqueleto poblado por la vista `home` cada campo que se ha definido en el archivo HTML como una variable de Django mediante las llaves dobles es alterada en cada presentación de la página y para cada estación. El archivo `home.html` también procede de un archivo `base.html` que tiene las decoraciones y funciones JavaScript básicas de todas las páginas del sitio por lo que cada vista solo define el contenido del cuerpo de la página web.

Urls.py

Este archivo se encarga de enlazar cada url del sitio con cada vista y por consecuencia con cada plantilla correspondiente. Su función es definir la url como una ruta donde cada parte de ella es tomada como un contexto para cada vista. Este archivo contiene únicamente las definiciones de las url propias del sitio y las url de administración proporcionadas por el framework de Django.

```

from django.conf.urls import include, url
from django.contrib import admin
from django.contrib.auth.views import login, logout

urlpatterns = [
    # Examples:
    # url(r'^$', 'alerta.views.home', name='home'),
    # url(r'^blog/', include('blog.urls')),

    url(r'^admin/', include(admin.site.urls)),
    url(r'^$', 'alertaApp.views.index', name='index'),
    url(r'^home/([a-z]+)/$', 'alertaApp.views.home', name='home'),
    url(r'^login/$', login, {'template_name': 'login.html'}, name='login'),
    url(r'^logout/$', logout, {'template_name': 'bloquear.html'}, name='logout'),
    url(r'^alerta_update/([a-z]+)/$', 'alertaApp.views.alerta_update',
name='alerta_update'),
    url(r'^registrar/$', 'alertaApp.views.registrar', name='registrar'),
    url(r'^recuperar/$', 'alertaApp.views.recuperar', name='recuperar'),
    url(r'^perfil/$', 'alertaApp.views.perfil', name='perfil'),
    url(r'^dispositivo/([a-z]+)/([a-z]+)/$', 'alertaApp.views.dispositivo',
name='dispositivo'),
    url(r'^graficas/([a-z]+)/([a-z]+)/$', 'alertaApp.views.graficas',
name='graficas'),
    url(r'^informacion/([a-z]+)/$', 'alertaApp.views.informacion', name='informacion'),
    url(r'^imagenes/([a-z]+)/$', 'alertaApp.views.imagenes', name='imagenes'),
    url(r'^videos/([a-z]+)/$', 'alertaApp.views.videos', name='videos'),

```

```

    url(r'^tabla/([a-z]+)/([a-z]+)/$', 'alertaApp.views.tabla', name='ta-
    bla'),
    url(r'^calendario/([a-z]+)/$', 'alertaApp.views.calendario', name='calen-
    dario'),
    url(r'^mapa/([a-z]+)/$', 'alertaApp.views.mapa', name='mapa'),
    url(r'^servidor/([a-z]+)/$', 'alertaApp.views.servidor', name='servi-
    dor'),
    url(r'^servidor_data/$', 'alertaApp.views.servidor_data', name='servi-
    dor_data'),
    url(r'^estacion_data/([a-z]+)/$', 'alertaApp.views.estacion_data',
    name='estacion_data'),
    url(r'^estacion/([a-z]+)/$', 'alertaApp.views.estacion', name='esta-
    ción'),
    url(r'^data/([a-z]+)/([a-z]+)/$', 'alertaApp.views.data', name='data'),
    url(r'^last_data_tacometro/([a-z]+)/([a-z]+)/$',
    'alertaApp.views.last_data_tacometro', name='last_data_tacometro'),
    url(r'^last_data/([a-z]+)/([a-z]+)/$', 'alertaApp.views.last_data',
    name='last_data'),
    url(r'^update/([a-z]+)/([a-z]+)/(\d+)/(\d{1,5}.\d{1,5})/$',
    'alertaApp.views.update', name='update'),
    url(r'^estacion_update/([a-
    z]+)/(\d{1,3}.\d{1,2})/(\d{1,3}.\d{1,2})/(\d{1,3}.\d{1,2})/(\d{1,2}.\d{1,2})/
    $', 'alertaApp.views.estacion_update', name='estacion_update'),
    url(r'^gps_update/([a-z]+)/(\d{1,3}.\d{1,9})/(\d{1,3}.\d{1,9})/$',
    'alertaApp.views.gps_update', name='gps_update'),
    url(r'^csv_file/([a-z]+)/([a-z]+)/$', 'alertaApp.views.csv_file',
    name='csv_file'),
    url(r'^enviar_mail/$', 'alertaApp.views.enviar_mail', name='en-
    viar_mail'),
    url(r'^tweet/$', 'alertaApp.views.tweet', name='tweet'),
    url(r'^upload_foto/([a-z]+)/$', 'alertaApp.views.upload_foto', name='up-
    load_foto'),
    url(r'^upload_video/([a-z]+)/$', 'alertaApp.views.upload_video',
    name='upload_video'),
]

handler404 = 'alertaApp.views.error404'
handler500 = 'alertaApp.views.error500'

```

SCRIPT 20. DEFINICION DE URLS DEL SITIO

Otros archivos

Existen también otros archivos que se utilizan en la creación y control del sitio. Algunos de ellos han sido creados para la administración del sitio de Django, por ejemplo el archivo admin sirve de enlace entre la aplicación admin de Django y la aplicación alertaApp

con el propósito de brindar una interfaz web para la administración del sitio de alerta temprana.

3.2.6 Configuración de Apache para uso con Django

La configuración necesaria para que Apache sirva la aplicación de Django se encuentra al final en el archivo de configuración “apache2.conf”

```
# CONFIGURACION DE LA APLICACION DE DJANGO

ServerName alertatemprana.com
WSGIDaemonProcess alertatemprana.com python-path=/var/www/alerta:/usr/local/lib/python2.7/site-packages
WSGIProcessGroup alertatemprana.com

Alias /favicon.ico /var/www/alerta/alerta/static/Imagenes/favicon.ico
Alias /media/ /var/www/alerta/alerta/media/
Alias /static/admin /usr/local/lib/python2.7/dist-packages/django/contrib/admin/static/admin/
Alias /static/ /var/www/alerta/alerta/static/

<Directory /usr/local/lib/python2.7/dist-packages/django/contrib/admin/static/admin/>
    Require all granted
</Directory>

<Directory /var/www/alerta/alerta/static>
    Require all granted
</Directory>

<Directory /var/www/alerta/alerta/media>
    Require all granted
</Directory>

WSGIScriptAlias / /var/www/alerta/alerta/wsgi.py

<Directory /var/www/alerta/alerta>
    <Files wsgi.py>
        Require all granted
    </Files>
</Directory>
```

SCRIPT 21. APACHE2.CONF

La configuración mostrada se utiliza para habilitar mediante el servidor de Apache los directorios y archivos en donde se encuentra la aplicación web creada con Django.

```
WSGIScriptAlias / /var/www/alerta/alerta/wsgi.py
```

El primer "/" en la línea WSGIScriptAlias es la ruta URL base que se sirve (/ url indica la raíz), y el segundo es la localización de un "archivo WSGI"

```
<Directory /var/www/alerta/alerta>  
  <Files wsgi.py>  
    Require all granted  
  </Files>  
</Directory>
```

Ruta hacia el directorio en donde se encuentra el archivo WSGI

```
WSGIDaemonProcess alertatemprana.com python  
path=/var/www/alerta:/usr/local/lib/python2.7/site-packages
```

Se utiliza el modo daemon que es el recomendado para sistemas no Windows en donde se especifica la ruta de los paquetes de python y la ruta de los paquetes de la aplicación web.

```
<Directory /var/www/alerta/alerta/static>  
  Require all granted  
</Directory>
```

Indica la ruta en donde se encuentran los archivos estáticos, es decir los plugins y software necesario para la creación de las plantillas y de igual manera se tiene para los archivos media que son los que los usuarios de la aplicación crearán.

3.3 MySQL server

3.3.1 Introducción

MySQL es un sistema de administración de bases de datos. Una base de datos es una colección estructurada de tablas que contienen datos. Esta puede ser desde una simple lista de compras a una galería de pinturas o el vasto volumen de información en una red corporativa. Para agregar, acceder y procesar datos guardados en un computador, necesita un administrador como MySQL Server. Dado que los computadores son muy buenos manejando grandes cantidades de información, los administradores de bases de datos juegan un papel central en computación, como aplicaciones independientes o como parte de otras aplicaciones.

MySQL es un sistema de administración relacional de bases de datos. Una base de datos relacional archiva datos en tablas separadas en vez de colocar todos los datos en un gran archivo. Esto permite velocidad y flexibilidad. Las tablas están conectadas por relaciones definidas que hacen posible combinar datos de diferentes tablas sobre pedido.

MySQL es software de fuente abierta. Fuente abierta significa que es posible para cualquier persona usarlo y modificarlo. Cualquier persona puede bajar el código fuente de MySQL y usarlo sin pagar. Cualquier interesado puede estudiar el código fuente y ajustarlo a sus necesidades. MySQL usa el GPL (GNU General Public License) para definir qué puede hacer y qué no puede hacer con el software en diferentes situaciones. Existe también una versión comercial licenciada.

3.3.2 Características

- Usa GNU Automake, Autoconf, y Libtool para portabilidad
- Uso de multihilos mediante hilos del kernel.
- Usa tablas en disco b-tree para búsquedas rápidas con compresión de índice
- Tablas hash en memoria temporales
- El código MySQL se prueba con Purify (un detector de memoria perdida comercial) así como con Valgrind, una herramienta GPL.
- Completo soporte para operadores y funciones en cláusulas select y where.
- Completo soporte para cláusulas group by y order by, soporte de funciones de agrupación

- Seguridad: ofrece un sistema de contraseñas y privilegios seguro mediante verificación basada en el host y el tráfico de contraseñas está cifrado al conectarse a un servidor.
- Soporta gran cantidad de datos. MySQL Server tiene bases de datos de hasta 50 millones de registros.
- Se permiten hasta 64 índices por tabla (32 antes de MySQL 4.1.2). Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes (500 antes de MySQL 4.1.2).
- Los clientes se conectan al servidor MySQL usando sockets TCP/IP en cualquier plataforma. En sistemas Windows se pueden conectar usando named pipes y en sistemas Unix usando ficheros socket Unix.
- En MySQL 5.0, los clientes y servidores Windows se pueden conectar usando memoria compartida.
- MySQL contiene su propio paquete de pruebas de rendimiento proporcionado con el código fuente de la distribución de MySQL.
- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- A bueno
- Posibilidad de selección de mecanismos de almacenamiento que ofrecen diferentes velocidades de operación, soporte físico, capacidad, distribución geográfica, transacciones...
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.

3.3.3 Instalación

MySQL se encuentra disponible en los repositorio de Ubuntu por lo tanto se instala el servidor de MySQL directamente así como también el cliente. *mysql-server*, *mysql-client* respectivamente.

3.3.4 Creación de base de datos

Para la creación de la base de datos que alberga los datos de los sensores y las condiciones de la estación de alerta temprana, así como también de los usuarios registrados en el sistema, se utiliza la sentencia de mysql para la línea de comandos

```
#mysqladmin -u root -p create alerta  
Enter password:*****
```

Dado que se utiliza el framework Django no es necesario realizar ninguna configuración más, puesto que la configuración se hace en el archivo settings.py de Django.

3.4 HTML5

Este es el lenguaje que se utiliza en las plantillas de la estación de alerta temprana. Es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. HTML5 especifica dos variantes de sintaxis para HTML: una «clásica», HTML (text/HTML), conocida como HTML5, y una variante XHTML conocida como sintaxis XHTML5, se utiliza html5 para las plantillas.

HTML5 establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Algunos de ellos son técnicamente similares a las etiquetas <div> y , pero tienen un significado semántico, como por ejemplo <nav> (bloque de navegación del sitio web) y <footer>.

3.4.1 Características

- Incorpora etiquetas (canvas 2D y 3D, audio, vídeo) con codecs para mostrar los contenidos multimedia. Actualmente hay una lucha entre imponer codecs libres (WebM + VP8) o privados (H.264/MPEG-4 AVC).
- Etiquetas para manejar grandes conjuntos de datos: Datagrid, Details, Menu y Command. Permiten generar tablas dinámicas que pueden filtrar, ordenar y ocultar contenido en cliente.
- Mejoras en los formularios. Nuevos tipos de datos (eMail, number, url, datetime) y facilidades para validar el contenido sin Javascript.

- Visores: MathML (fórmulas matemáticas) y SVG (gráficos vectoriales). En general se deja abierto a poder interpretar otros lenguajes XML.
- Drag & Drop. Nueva funcionalidad para arrastrar objetos como imágenes.

3.5 Bootstrap

Twitter Bootstrap es un framework o conjunto de herramientas de software libre para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales. Este se utiliza en la creación de las plantillas del sitio web para darle una mayor vistosidad a la página web. No se utiliza el framework como tal sino solo los paquetes de JavaScript que permiten ciertas funcionalidades en el sitio de la estación de alerta temprana como el manejo de tablas mostradas en el navegador y el calendario.

3.5.1 Características

Bootstrap tiene un soporte relativamente incompleto para HTML5 y CSS 3, pero es compatible con la mayoría de los navegadores web. La información básica de compatibilidad de sitios web o aplicaciones está disponible para todos los dispositivos y navegadores. Existe un concepto de compatibilidad parcial que hace disponible la información básica de un sitio web para todos los dispositivos y navegadores. Por ejemplo, las propiedades introducidas en CSS3 para las esquinas redondeadas, gradientes y sombras son usadas por Bootstrap a pesar de la falta de soporte de navegadores antiguos.

3.6 CSS3

Hoja de estilo en cascada o CSS (siglas en inglés de cascading style sheets) es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML se utiliza en las plantillas de la página web para los estilos dados a los elementos de los bloques y para hacer que las paginas luzcan más amigables.

3.7 Google Maps

La estación de alerta temprana tiene como una de sus características un GPS que reporta las coordenadas en un formato de latitud y longitud. Se ha definido para la

determinación de riesgo que cada estación de alerta temprana cubra un radio de un kilómetro, por lo tanto cada usuario registrado en el sitio de la estación de alerta temprana tiene la capacidad de observar si en la ubicación que se encuentra al momento de visitar el sitio, se pueda encontrar en peligro.

Esto se debe a que la Api de google maps en su tercera versión permite el rastreo mediante navegador web. Esto es, cada vez que un usuario acceda al sitio de la estación de alerta temprana la página solicitará al usuario el permiso para realizar la localización del computador o de la terminal móvil. Y lo ubicará en el mapa de El Salvador y mostrará mediante un círculo de color azul la zona que se definió como de riesgo.

3.7.1 Codificación

La codificación se ha hecho en cada plantilla en donde se muestra el mapa de la ubicación de la estación de alerta temprana. El lenguaje de codificación es JavaScript que corre o se ejecuta en el navegador en el momento de cargar la pagina por lo tanto siempre muestra la posición real del usuario de la estación de alerta temprana.

```
<script type="text/javascript"
src="http://maps.google.com/maps/api/js"></script>

<script>
function iniciar() {
    var mapOptions = {
        center: new google.maps.LatLng({{estacion.coordenadas}}),
        zoom: 9,
        mapTypeId: google.maps.MapTypeId.HYBRID
    };
    var map = new google.maps.Map(document.getElementById("map"),mapOptions);

    var control = new google.maps.LatLng(13.706296, -89.1351465);

    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(mostrarUbicacion);
    }

    var marker = new google.maps.Marker({
        position: control,
```

```

        map: map,
        title: 'Estación de Control',
        animation: google.maps.Animation.BOUNCE,});

var marker2 = new google.maps.Marker({
    position: map.getCenter(),
    map: map,
    title: 'Estación de alerta temprana {{ estacion.nombre }}',
    animation: google.maps.Animation.BOUNCE,});

var popup = new google.maps.InfoWindow({content: 'Centro de control'});
popup.open(map, marker);

var popup = new google.maps.InfoWindow({content: 'Estación {{ es-
tacion.nombre }}'});
popup.open(map, marker2);

function showInfo() {
    map.setZoom(16);
    map.setCenter(marker.getPosition());
    var contentString = 'Ubicación Centro de Control';
    var infowindow = new google.maps.InfoWindow({content: 'Centro de
control ubicado en ya saben donde'});
    infowindow.open(map,marker);}

var cobertura = new google.maps.Circle({
    center:map.getCenter(),
    radius:1000,
    strokeColor:"#0000FF",
    strokeOpacity:1.0,
    strokeWeight:2,
    fillColor:"#0000FF",
    fillOpacity:0.2,
    editable:false});

cobertura.setMap(map);

function showInfo2() {
    map.setZoom(16);
    map.setCenter(marker2.getPosition());
    var contentString = 'Ubicación Estación {{ estacion.nombre }}';
    var infowindow = new google.maps.InfoWindow({content: 'Rio {{ esta-
cion.nombre }}, más información en: <a href="/home/{{ estacion.nombre | lower
}}/">AdminSAT</a>'});

```

```

        infowindow.open(map,marker2);}

function showInfo3() {
    map.setZoom(16);
    map.setCenter(marker3.getPosition());
    var contentString = 'Mi ubicación';
    var infowindow = new google.maps.InfoWindow({content: 'Aquí esoty
yo!'});
    infowindow.open(map,marker3);}

    google.maps.event.addListener(marker, 'click', showInfo);
    google.maps.event.addListener(marker2, 'click', showInfo2);

function mostrarUbicacion(position) {
    var latitud = position.coords.latitude;
    var longitud = position.coords.longitude;

    var yo = new google.maps.LatLng(latitud, longitud);

    {% if user.is_authenticated %}
    var fotografia = {
        url: '{{ user.perfil.fotografia.url }}',
        scaledSize: new google.maps.Size(70, 75),
        origin: new google.maps.Point(0, 0),
        anchor: new google.maps.Point(0, 32)};
    {% else %}
    var fotografia = {
        url: '/static/Imagenes/guest.jpg',
        scaledSize: new google.maps.Size(70, 75),
        origin: new google.maps.Point(0, 0),
        anchor: new google.maps.Point(0, 32)};
    {% endif %}

    var marker3 = new google.maps.Marker({
        position: yo,
        icon: fotografia,
        map: map,
        title: 'Esta es mi ubicación geográfica',
        animation: google.maps.Animation.BOUNCE,});

    var popup = new google.maps.InfoWindow({content: 'Aquí estoy!'});
    popup.open(map, marker3);

    google.maps.event.addListener(marker3, 'click', showInfo3);}

```

```

    }
    google.maps.event.addDomListener(window, 'load', iniciar);
</script>

```

SCRIPT 22. CONFIGURACIÓN MAPA DE RIESGO

En el código se definen 3 mapas o ubicaciones geográficas, la primera es la ubicación geográfica de la estación de alerta temprana

```

enter: new google.maps.LatLng({{estacion.coordenadas}})

```

La segunda es la ubicación del centro de control.

```

var control = new google.maps.LatLng(13.706296, -89.1351465)

```

Una tercera ubicación se establece si el usuario acepta el servicio de localización web.

```

var yo = new google.maps.LatLng(latitud, longitud)

```

Luego de esto se establece el marcador en el mapa para cada ubicación geográfica registrada.

```

var marker = new google.maps.Marker({
  position: control,
  map: map,
  title: 'Estación de Control',
  animation: google.maps.Animation.BOUNCE,});

var marker2 = new google.maps.Marker({
  position: map.getCenter(),
  map: map,
  title: 'Estación de alerta temprana {{ estacion.nombre }}',

```

```
animation: google.maps.Animation.BOUNCE, });  
  
var marker3 = new google.maps.Marker({  
    position: yo,  
    icon: fotografia,  
    map: map,  
    title: 'Esta es mi ubicación geográfica',  
    animation: google.maps.Animation.BOUNCE, });
```

Si el usuario ha proporcionado una fotografía al momento de registrarse esta se mostrara en la ubicaron en donde se encuentra en el mapa.

Capítulo 4

Resultados

4.1 Sitio WEB

Mediante capturas de pantalla se muestra la página web en funcionamiento de la estación de alerta temprana.

Esta es la página de inicio del sistema de alerta temprana. Se muestran las imágenes capturadas por la cámara, las alertas emitidas por Twitter y las últimas condiciones registradas por los sensores. A la izquierda está el panel de navegación.

The screenshot displays the 'adminSAT' web application interface for the 'Estación Lempa'. The main content area shows four real-time sensor data cards: Rain (1.68 mm), Pressure (25.15 Pa), Relative Humidity (37.00%), and Temperature (31.44°C). Below these is a 'Fotogaleria' section with a photo of a river. To the right is a 'Twitter' section showing three tweets from '@alert_sat' with the status 'alerta verde'. The left sidebar contains navigation options like 'Principal', 'Dispositivos', 'Gráficas', 'Tablas', 'Mapa', 'Calendario', 'Imágenes', 'Videos', and 'Info+'. The top navigation bar includes 'Estaciones' and a user profile for 'Jose'.

ILUSTRACIÓN 20. PÁGINA DE HOME (INICIO)

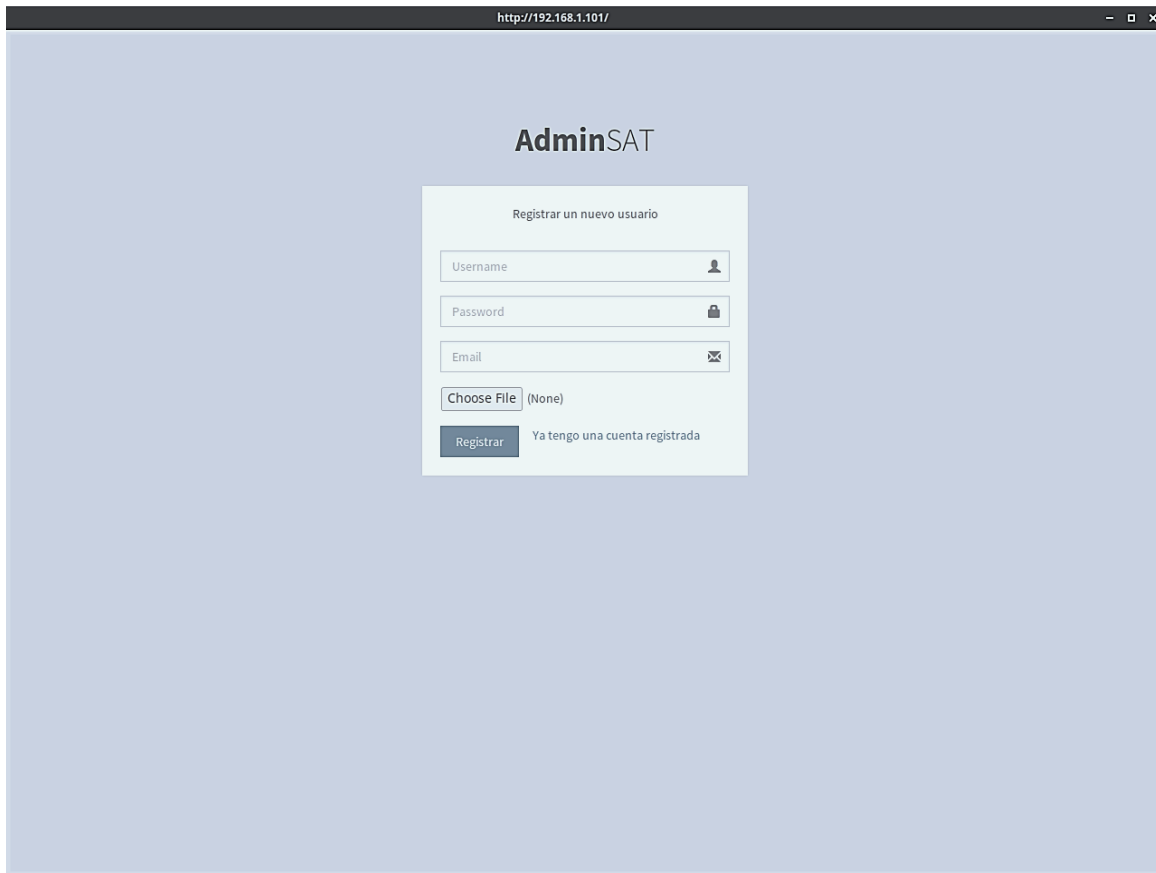


ILUSTRACIÓN 21. PÁGINA DE REGISTRO DE USUARIOS

Página de registro de usuarios, en donde los habitantes de las comunidades pueden registrarse y observar las condiciones de la estación de alerta temprana. Al estar registrados se les permite un mayor acceso a las características del sitio.

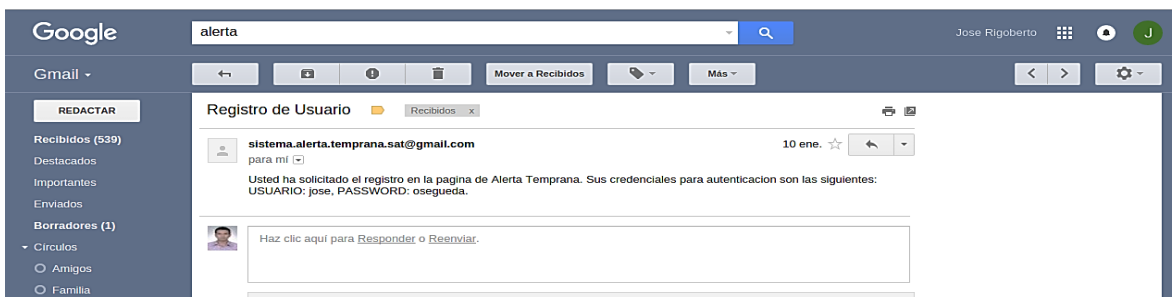
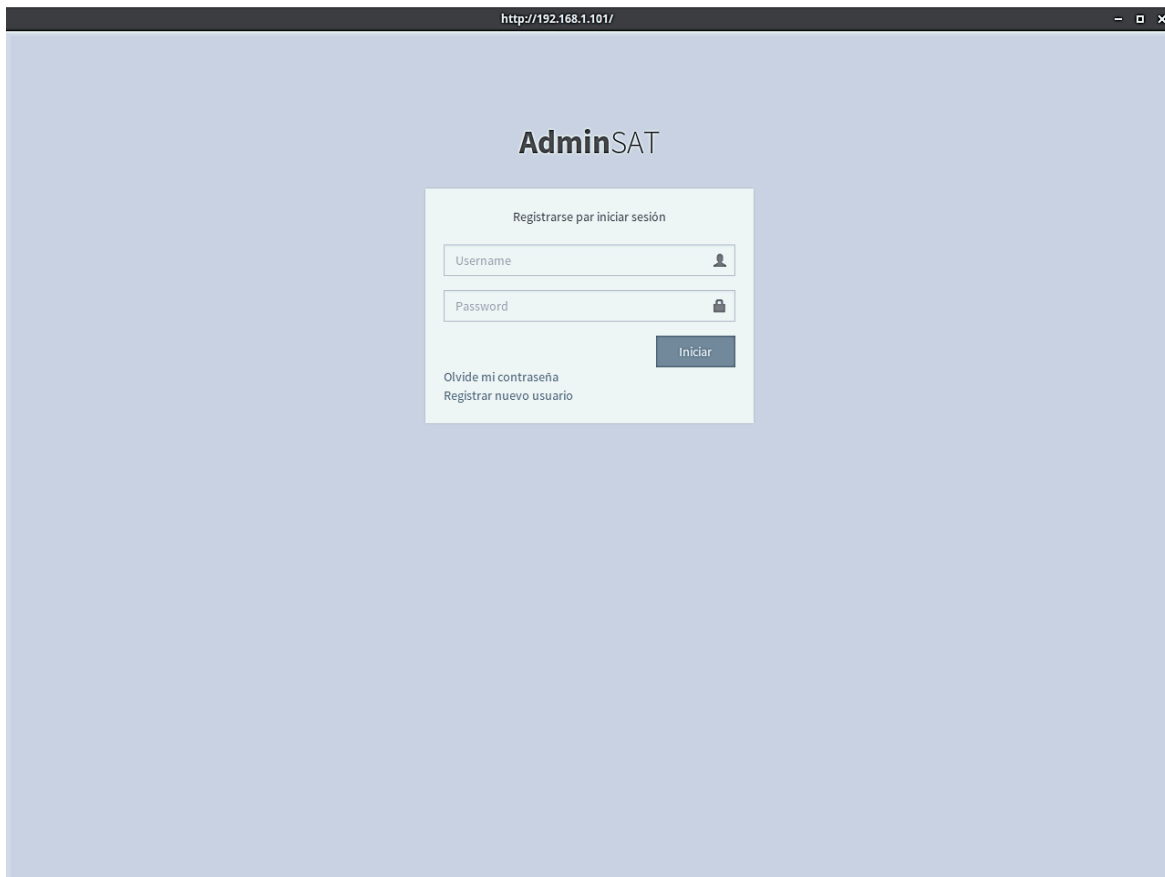


ILUSTRACIÓN 22. CORREO DE CONFIRMACIÓN DE REGISTRO


Cada vez que un usuario se registra en el sitio de la estación de alerta temprana se le notifica por correo electrónico la información de sus credenciales de registro.




http://192.168.1.101/

AdminSAT

Registrarse par iniciar sesión

Username 

Password 

Iniciar

[Olvide mi contraseña](#)
[Registrar nuevo usuario](#)

ILUSTRACIÓN 23. PÁGINA DE INICIO DE SESIÓN

Esta es la página de redirección que realiza el sitio para aquellos usuarios que desean iniciar sesión dentro de la estación de alerta temprana. Solo es necesario ingresar el nombre de usuario y la contraseña.

Si el usuario llegase a perder la información de su cuenta, el sistema le pedirá su correo electrónico de registro y se le enviará un correo donde se dan indicaciones de cómo recuperar la cuenta.

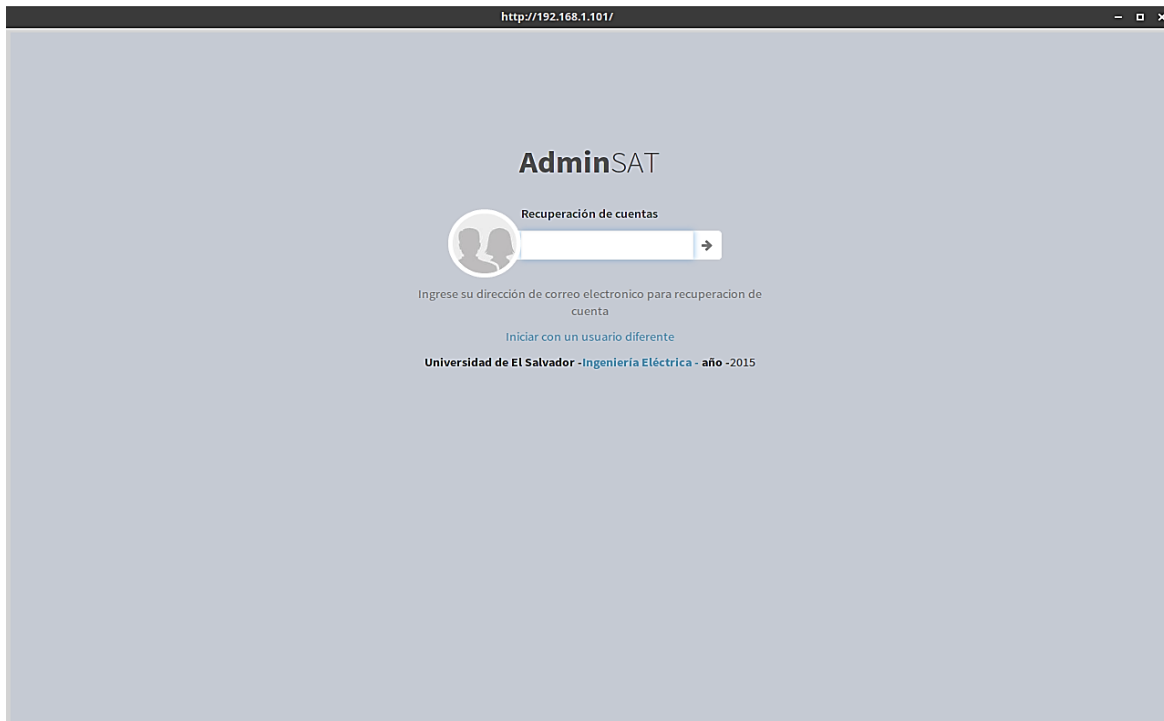


ILUSTRACIÓN 24. PÁGINA DE RECUPERACIÓN DE CUENTA

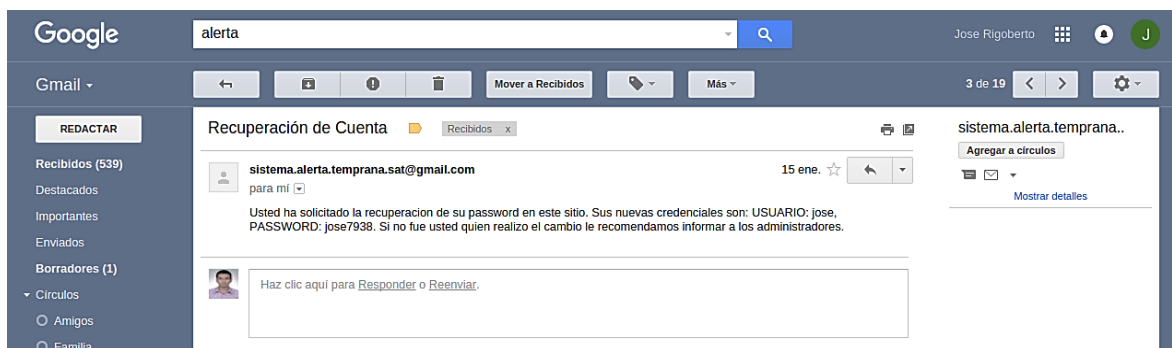


ILUSTRACIÓN 25. CORREO DE RECUPERACIÓN DE CUENTA

Luego de ingresar el correo de registro, el sitio envía un correo donde genera un nuevo password y el mismo nombre de usuario.

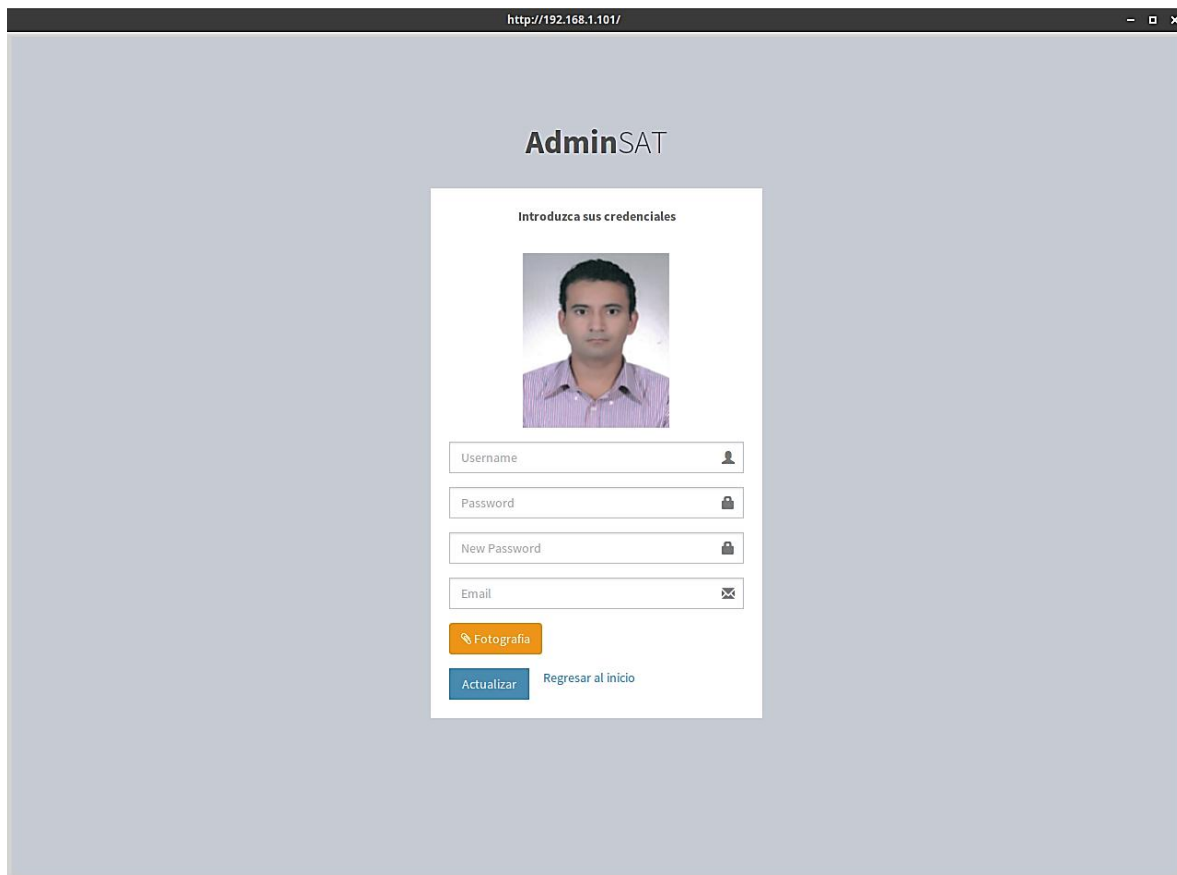


ILUSTRACIÓN 26. PÁGINA DE USUARIO REGISTRADO

Cuando el usuario ya se encuentra registrado puede modificar la información de su perfil. Cambio de correo, password, nombre de usuario y la fotografía.

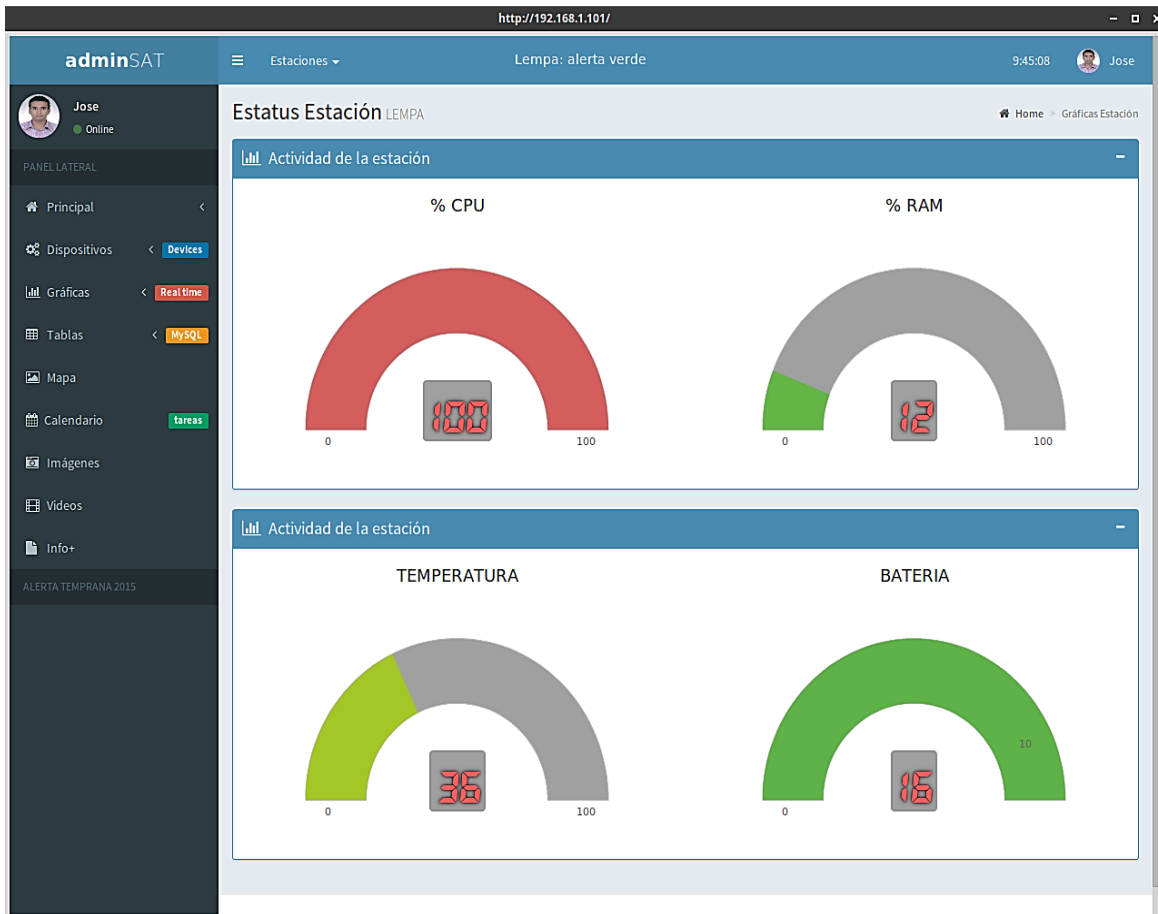


ILUSTRACIÓN 27. CONDICIONES DEL SAT

Se muestra en forma gráfica mediante 4 indicadores tipo círculo las últimas condiciones reportadas por la estación de alerta temprana.

Dependiendo de la condición así será el color de la gráfica, verde significa que el valor es el recomendado y rojo es un valor que ha llegado al límite máximo.

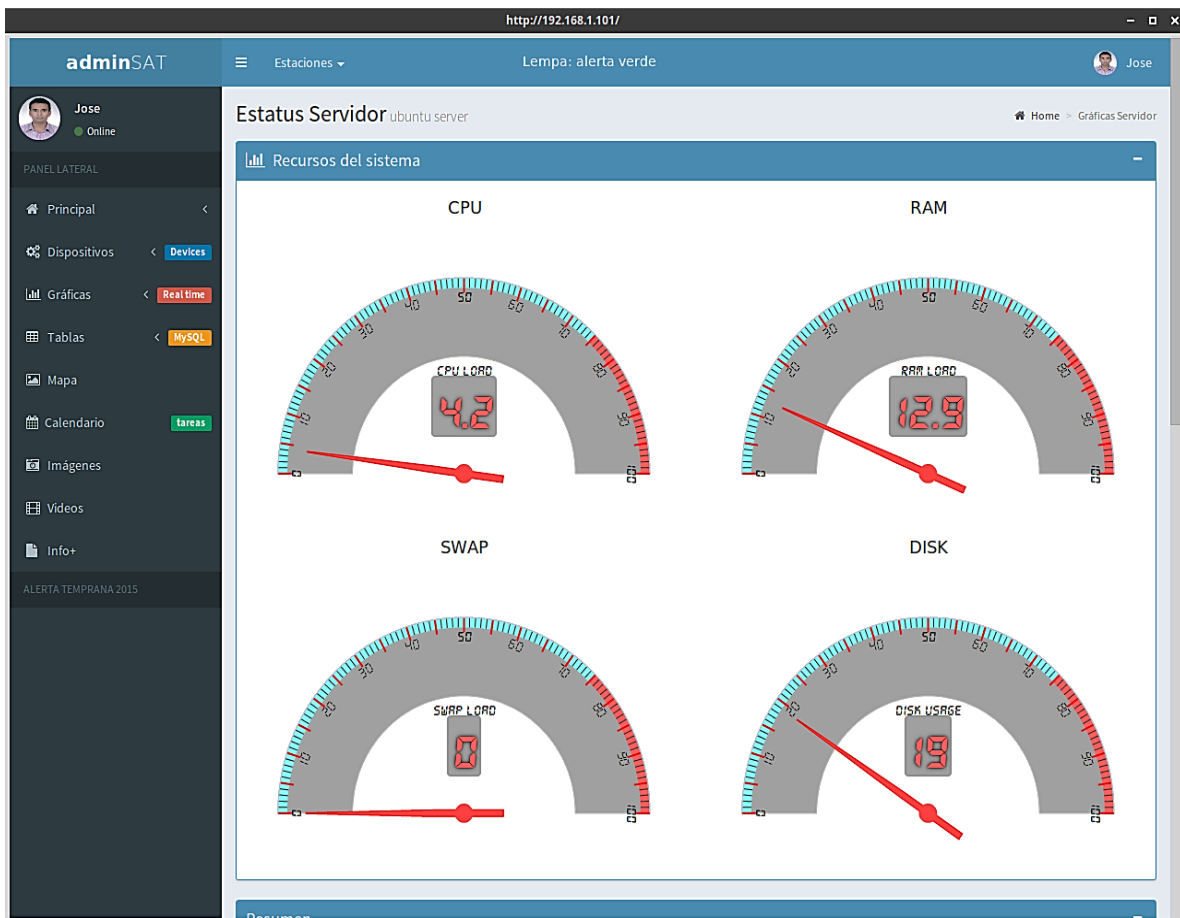


ILUSTRACIÓN 28. CONDICIONES DEL SERVIDOR

Se muestran las condiciones en las que se encuentra el servidor de la estación de alerta temprana mediante 4 indicadores tipo tacómetro, de igual forma la zona roja indica alta carga de trabajo del servidor.

The screenshot shows the adminSAT web interface. The top navigation bar includes the user name 'Jose' and the time '9:57:29'. The main content area is titled 'dispositivo Estación Lempa' and 'Preción Barométrica'. The device is identified as 'BMP180'. The page is divided into two columns: 'Descripción' and 'Características Eléctricas'.

Descripción

BMP180

General
Sensor de alta precisión, barómetro digital Bosch BMP180 de baja potencia. El BMP180 ofrece un rango de medición de presión de 300 a 1.100 hPa con una precisión de hasta 0,02 hPa en el modo de resolución avanzada. Se basa en la tecnología piezo-resistivo de alta precisión, robustez y estabilidad a largo plazo. Calibrado de fábrica, con los coeficientes de calibración ya almacenados en la memoria ROM.

Aplicaciones
Medición de presión barométrica

Más información
https://learn.sparkfun.com/tutorials/bmp180-barometric-pressure-sensor-hookup-?_ga=1.93469341.1071870409.1456388222

Características Eléctricas

Tensión de operación	3.30
Comunicación	I2C
Rango de operación	300hPa a 1100hPa
Operación	Indoor

Estado : True

Universidad de El Salvador -Ingeniería Eléctrica - año 2015

Version 0.01

ILUSTRACIÓN 29. PÁGINA DE INFORMACIÓN DE DISPOSITIVOS

En esta página se muestra la información del dispositivo que se utiliza para la medición de la presión barométrica, se puede obtener un documento PDF referencia del fabricante donde se muestra la información detallada.

Esta misma página también se encuentra disponible para los demás dispositivos de la estación de alerta temprana, por referencia solo se muestra para el sensor de presión barométrica.

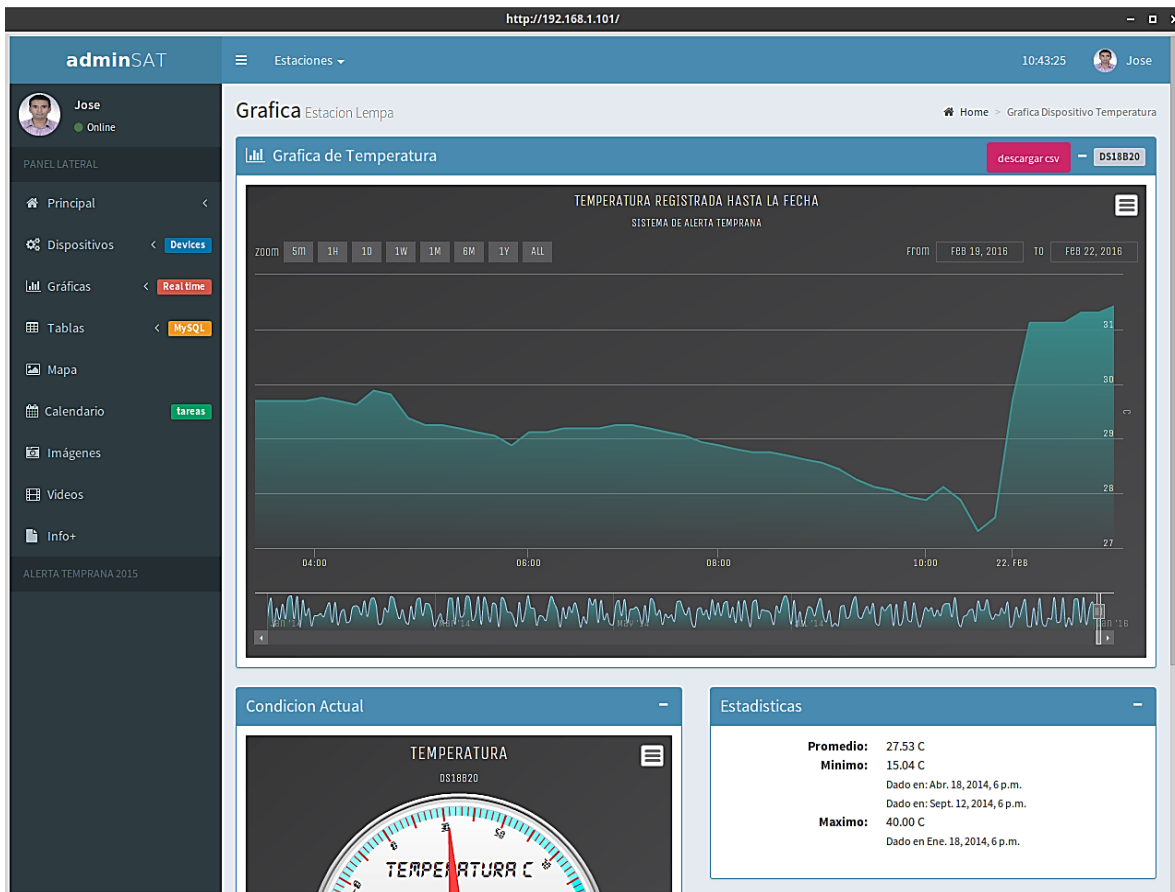


ILUSTRACIÓN 30. PÁGINA DE GRÁFICAS Y ESTADÍSTICAS

Esta es la página donde se muestra la actividad que ha registrado la estación de alerta temprana gráficamente. La gráfica es interactiva de la forma que el usuario puede arrastrar con el ratón la fecha de inicio y la fecha de fin y la gráfica se re-acomoda para mostrar solo esa franja de datos.

También la gráfica permite seleccionar rangos de tiempo predefinidos, para 5 minutos, una hora, un día, una semana, un mes, 6 meses, un año, o todos los datos registrados. Al seleccionar una franja de tiempo predefinida y al arrastrarla con el ratón es posible obtener condiciones iniciales o finales, por ejemplo seleccionar 5 minutos y arrastrarlos hacia la derecha, la gráfica mostrara los últimos 5 minutos registrados por la estación de

alerta temprana y en caso contrario los primeros 5 minutos de un día por ejemplo o la primer hora de lluvia, o 5 minutos antes de la tormenta, permitiendo así conocer con exactitud las condiciones antes, durante y después de un fenómeno natural en estudio.



ILUSTRACIÓN 31. TACÓMETRO DE REGISTRO

Al mismo tiempo que se generan los datos la página muestra en tiempo real mediante el tacómetro el último valor ingresado por la estación de alerta temprana hacia el sitio. El tacómetro muestra información además de la escala máxima y mínima en relación a los valores máximos y mínimos que la estación puede registrar para un sensor dado, en este caso el sensor de humedad.

La página de gráfica muestra también un cuadro donde se presenta el promedio de la magnitud registrada por el sensor. Se muestra el máximo y mínimo de los valores registrados así como también las fechas en que se dieron las mediciones.

Estadísticas	
Promedio:	27.53 C
Minimo:	15.04 C
	Dado en: Abr. 18, 2014, 6 p.m.
	Dado en: Sept. 12, 2014, 6 p.m.
Maximo:	40.00 C
	Dado en Ene. 18, 2014, 6 p.m.

ILUSTRACIÓN 32. CUADRO DE PROMEDIO, MÁXIMOS Y MÍNIMOS

The screenshot shows the adminSAT interface. The main content area displays a table titled 'Tabla Estacion Lempa' for 'Temperatura - DS18B20'. The table has two columns: 'Fecha - hora' and 'Temperatura Celsius'. The data rows show temperature readings from April 1st to April 13th, 2014, at 6 p.m. The interface includes a sidebar with navigation options like 'Principal', 'Dispositivos', 'Gráficas', 'Tablas', 'Mapa', 'Calendario', 'Imágenes', 'Videos', and 'Info+'. The footer contains 'Universidad de El Salvador - Ingeniería Eléctrica - año 2015' and 'Version 0.01'.

Fecha - hora	Temperatura Celsius
Abr. 1, 2014, 6 p.m.	30.07
Abr. 1, 2014, 6 p.m.	24.41
Abr. 10, 2014, 6 p.m.	28.19
Abr. 10, 2014, 6 p.m.	16.42
Abr. 11, 2014, 6 p.m.	15.32
Abr. 11, 2014, 6 p.m.	19.67
Abr. 12, 2014, 6 p.m.	30.66
Abr. 12, 2014, 6 p.m.	39.64
Abr. 13, 2014, 6 p.m.	21.83
Abr. 13, 2014, 6 p.m.	32.30

ILUSTRACIÓN 33. PÁGINA DE TEBLAS

La página de tablas muestra en forma de una tabla seccionada todos los datos registrados por la estación. En esta página se pueden seleccionar rangos de datos por cantidad y ordenarlos por fecha. Cambien es posible mediante esta página la búsqueda de una fecha para conocer el valor que se registró para dicha fecha o en forma inversa conocer la fecha en que se dio algún valor registrado en la base de datos.

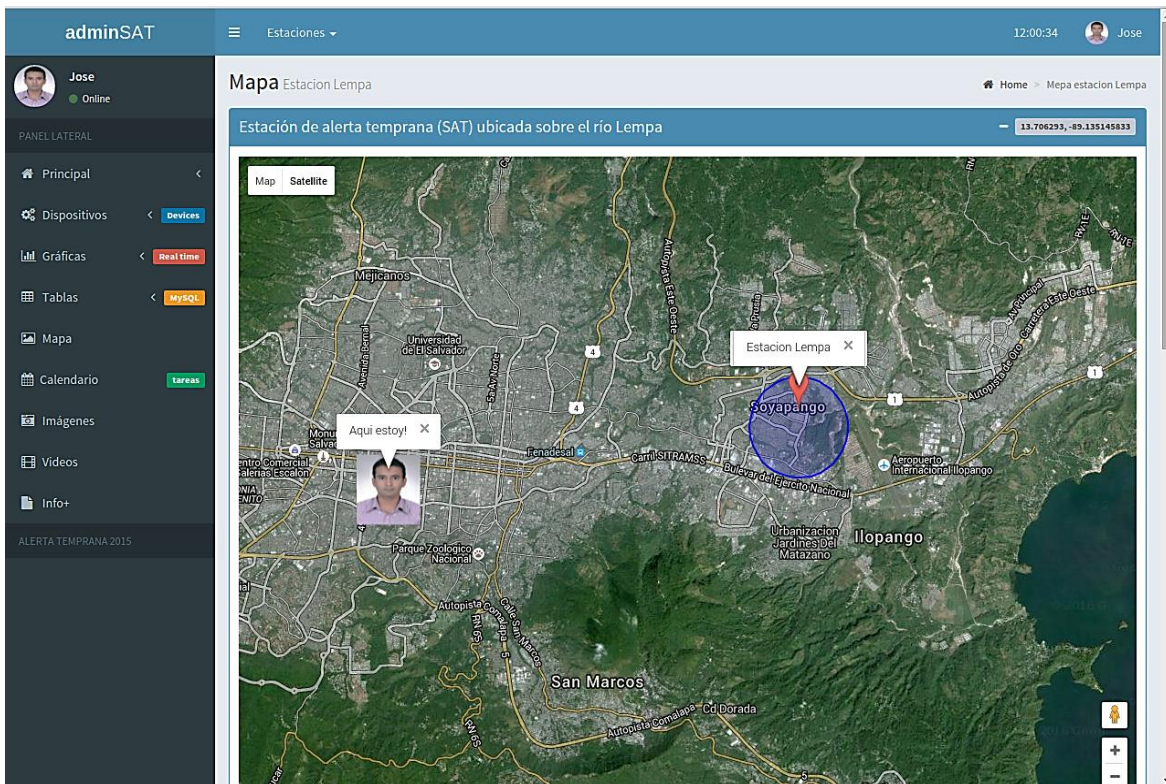


ILUSTRACIÓN 34. MAPA DE RIESGO

En la página de mapa se muestra la ubicación en el mapa de El Salvador donde está la estación de alerta temprana, el centro de control, donde se monitorea la estación y al usuario registrado. El círculo azul indica la zona de riesgo que cubre la estación de alerta temprana.

ILUSTRACIÓN 35. PÁGINA DE CALENDARIO DE ACTIVIDADES

En esta página se pueden agendar actividades a realizar en la estación de alerta temprana los eventos se muestran en color diferente al gris para actividades o eventos pendientes de cumplir o de color gris para eventos pasados.

Los usuarios registrados pueden visualizar el calendario de actividades y así estar al tanto de las actividades que realizan las autoridades con respecto a la estación de alerta temprana.

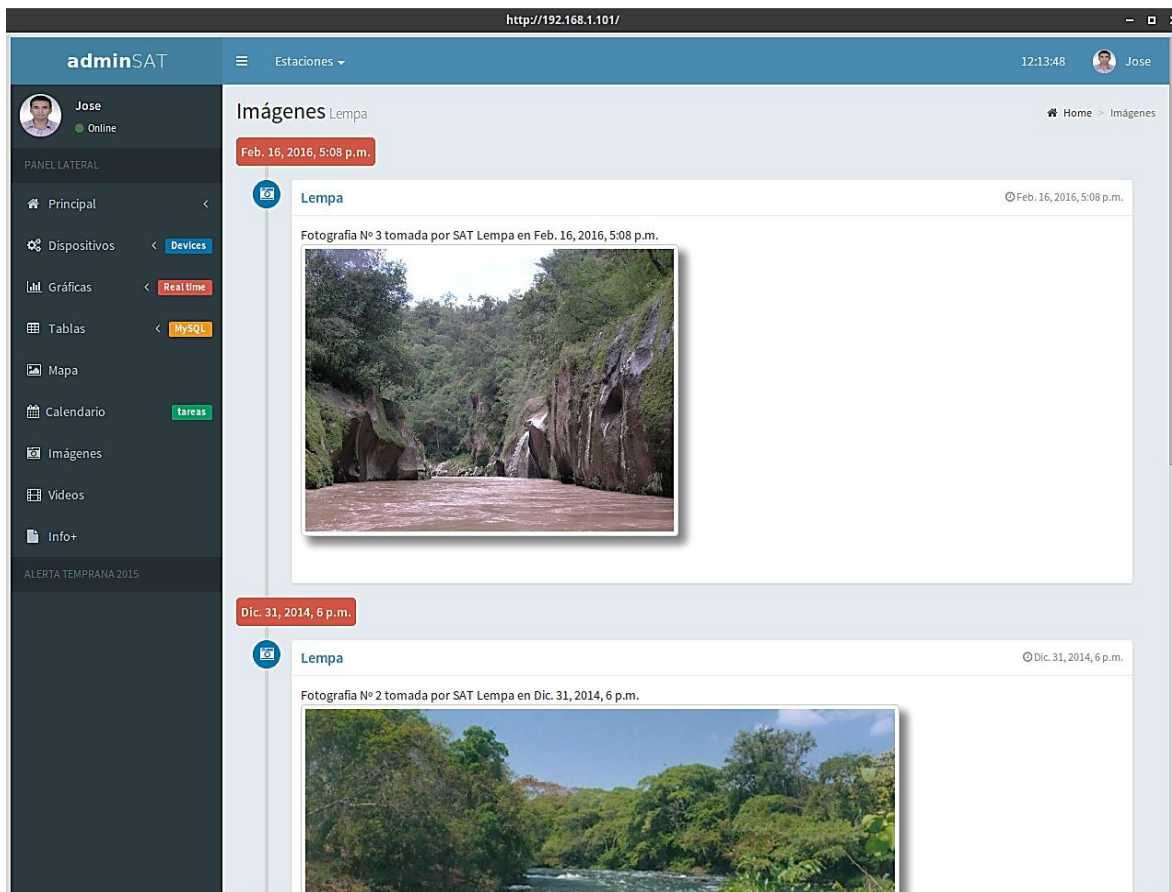


ILUSTRACIÓN 36. PÁGINA DE IMÁGENES

En esta página se muestran las fotografías que ha tomado durante la operación, la estación de alerta temprana. Las imágenes están clasificadas por fecha en que fueron tomadas y se muestran primero las últimas fotografías tomadas hasta llegar a la primera fotografía realizada.

Las fotografías se pueden descargar para generar reportes posteriores a un acontecimiento de la naturaleza.

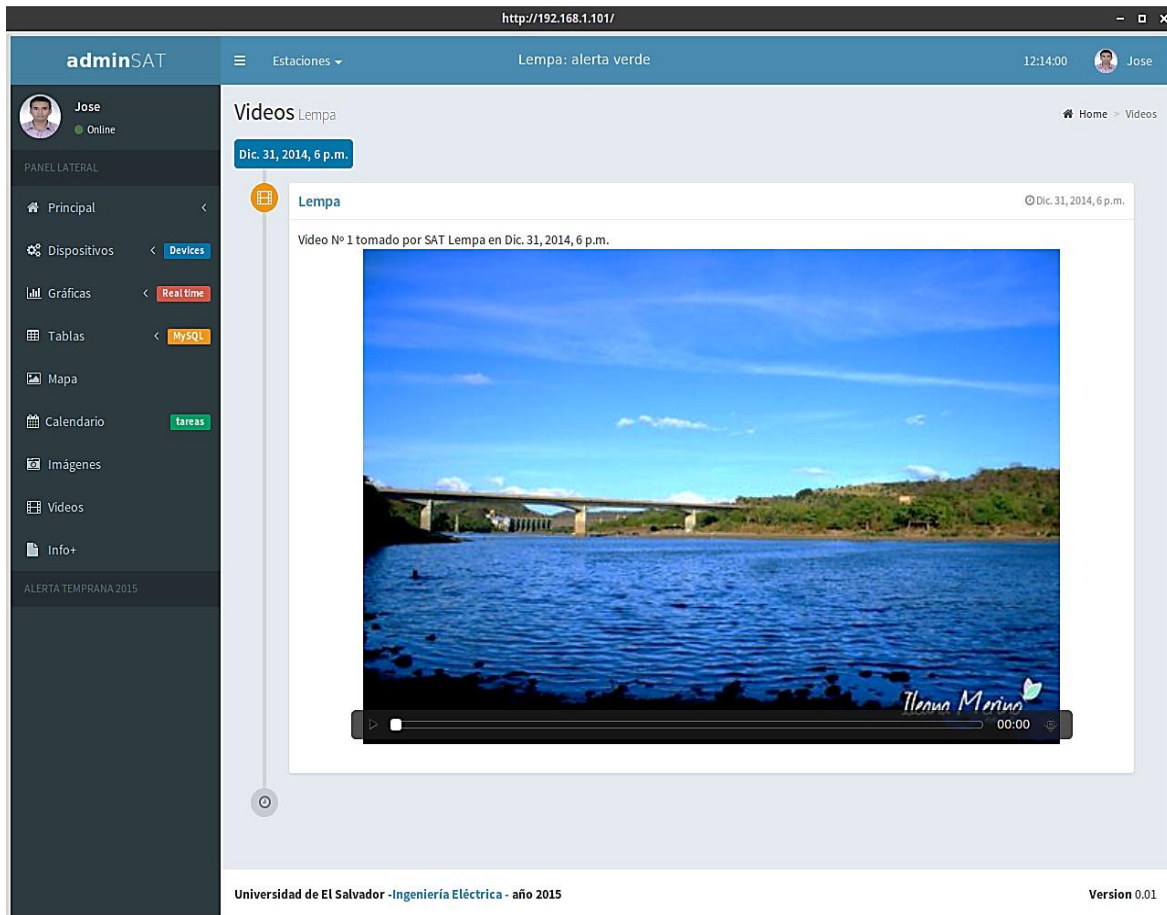


ILUSTRACIÓN 37. PÁGINA DE VIDEOS

La página de videos al igual que la página de imágenes muestra los videos tomados por la estación de alerta temprana durante el periodo de operación. De igual forma los videos se clasifican por fecha de captura y se muestran primero los últimos videos tomados por la estación hasta llegar al primer video capturado.

Los videos están en formato mp4 se pueden descargar de la página. Cada video tiene una duración de 1 minuto esto garantiza que un video no tenga más de 2 MB de tamaño y sean fácilmente observables.

The screenshot shows the AdminSAT web interface. The top navigation bar includes the user profile 'Jose' and the current station 'Lempa: alerta verde'. The main content area is titled 'estación Estación Lempa' and contains the following information:

- Alerta:** alerta verde
- Descripción:** Estación de alerta temprana (SAT) ubicada sobre el río Lempa
- Datos:**
 - IP: 192.168.43.96
 - Correo: lempa@gmail.com
 - Teléfono: 77177043
 - Coordenadas: 13.706293, -89.135145833
- Mapa:** A satellite map showing the location of Estación Lempa near Soyapango, with a red pin and a search box.
- Últimas condiciones:**
 - CPU:** 100%
 - RAM:** 12.9%
 - Batería:** 16.7%
 - Temperatura:** 36.8°C

ILUSTRACIÓN 38. PÁGINA DE INFORMACIÓN

La página de información muestra un poco más de información técnica acerca de la estación de alerta temprana, se muestra el teléfono, la dirección de correo, la IP, las coordenadas de la estación, las últimas condiciones registradas de sus recursos, y mini mapa de referencia

SAT Administracion Bienvenido/a, djangouser. Ver el sitio / Cambiar contraseña / Terminar sesión

Sitio administrativo

alertaApp Aplicacion	
Condicion	+ Añadir ✎ Modificar
Dispositivos	+ Añadir ✎ Modificar
Estaciones	+ Añadir ✎ Modificar
Eventos	+ Añadir ✎ Modificar
Fotografias	+ Añadir ✎ Modificar
Perfils	+ Añadir ✎ Modificar
Videos	+ Añadir ✎ Modificar

Autenticación y autorización	
Grupos	+ Añadir ✎ Modificar
Usuarios	+ Añadir ✎ Modificar

Acciones recientes	
Mis acciones	
✘ Nombre: Instalacion de SAT Rio Lempa, Inicio 2016-01-06 14:00:00+00:00, Final: 2016-01-06 11:00:00+00:00	
Evento	
✎ Nombre: Humedad, Modelo: DHT11, Estado: True	
Dispositivo	
✎ Nombre: Nivel, Modelo: HC-SR04, Estado: True	
Dispositivo	
✎ Nombre: Presion, Modelo: BMP180, Estado: True	
Dispositivo	
✎ Nombre: Temperatura, Modelo: DS18B20, Estado: True	
Dispositivo	
✎ Nombre: Precipitacion, Modelo: WS100, Estado: True	
Dispositivo	
✎ Nombre: Temperatura, Modelo: DS18B20, Estado: True	
Dispositivo	
✎ Nombre: Presion, Modelo: BMP180, Estado: True	
Dispositivo	

ILUSTRACIÓN 39. PÁGINA DE ADMINISTRACION DEL SITIO

El framework de Django proporciona un sitio de administración del sitio de alerta temprana. En este sitio que es parte del sitio de la estación de alerta temprana es donde se realiza el mantenimiento a la página de la estación de alerta temprana, aquí se añaden sensores, se modifican permisos de usuarios y sesiones, se agregan más estaciones de alerta temprana y se puede realizar un gestión mínima de la base de datos, el sitio permite la remoción de datos registrados, usuarios, estaciones de alerta temprana, dispositivos, eventos en el calendario, etc.

4.2 Alertas enviadas por la estación

Aquí se muestran las alertas enviadas por el sistema de alerta temprana, tanto las alertas enviadas por la red social de Twitter y las alertas enviadas o emitidas por mensajes de texto.

4.2.1 Tweets

The screenshot displays the Twitter profile of 'Alerta Temprana SAT' (@alert_sat). The profile header includes a yellow triangular warning icon with a red exclamation mark and the word 'ALERT' in red. Below the header, the profile name and handle are shown. The main content area is divided into 'Tweets' (7) and 'SIGUIENDO' (2). The tweets list several 'alerta verde' messages and one 'alerta amarilla' message, all from the account. The right sidebar features a 'A quién seguir' section with accounts like Dandovia, Bomberos El Salvador, and elsalvador.com, and a 'Tendencias' section with various trending hashtags.

ILUSTRACIÓN 40. ALERTAS EMITIDAS EN RED SOCIAL TWITTER

Para los usuarios que siguen la cuenta de Twitter de la estación de alerta temprana, estas alertas se pueden observar en la página de Twitter o también en la página de home de la estación de alerta temprana.

4.2.2 Mensajes de texto

Aquí se muestran las alertas generadas por mensajes de texto enviadas hacia los usuarios registrados en el directorio telefónico interno de la estación de alerta temprana.

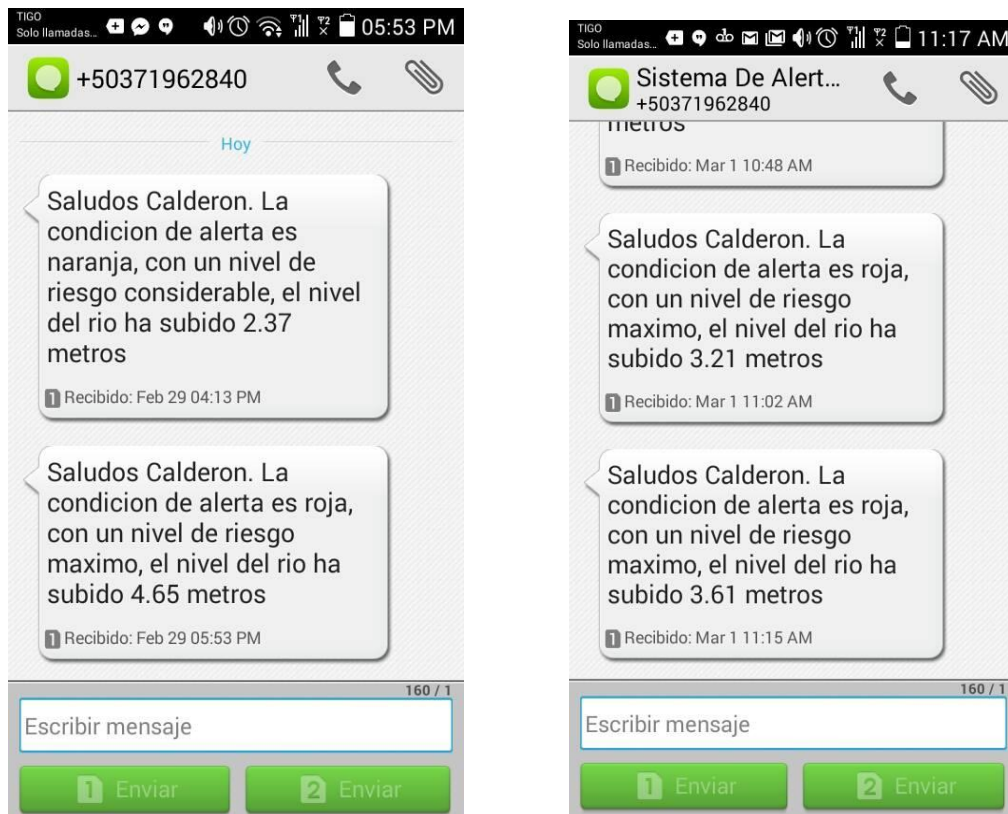


ILUSTRACIÓN 41. ALERTAS EMITIDAS POR MENSAJE DE TEXTO

Los mensajes de texto se envían cuando se realiza una actualización de la condición del río. Si la condición es de gravedad se envían mensajes de alerta mediante llamadas telefónicas y mensajes de texto a todos los usuarios registrados en el directorio telefónico interno de la estación de alerta temprana. Las llamadas de vos las realiza la propia estación de alerta temprana mediante un sintetizador de voz computarizado.

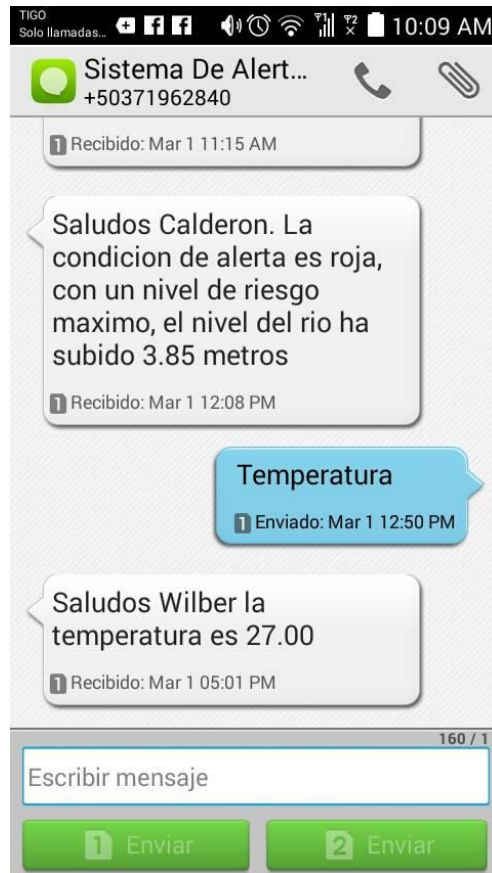


ILUSTRACIÓN 42. MENSAJE INFORMATIVO

Aquí se muestra un mensaje de texto enviado para la estación de alerta temprana hacia un usuario que envió un mensaje de texto bajo el formato establecido para conocer la temperatura medioambiental que reporta la estación de alerta temprana.

Conclusión

El diseño y elaboración de un sistema de alerta temprana es factible, utilizando las tecnologías hasta la fecha alcanzadas, ya que permite con la comodidad y en tiempo real la determinación de condiciones de riesgo, que alertan a las personas sobre la gravedad de las condiciones que se presentan en las cercanías de la estación de alerta temprana, previniendo así la pérdida de vidas humanas. Además un sistema de alerta temprana sistematizado permite un mayor control e información hacia la población y las autoridades, permitiendo así la pronta actuación ante las alertas generadas y las condiciones monitoreadas.

Dadas las condiciones la población puede llegar a ser menos vulnerable, con el conocimiento de las condiciones medioambientales, las alertas generadas y el riesgo reportado por la estación SAT.

Finalmente un sistema de alerta temprana de este tipo, permite el registro de los datos, para que a futuro se tomen acciones en base a condiciones registradas en el pasado que conllevaran a una situación potencialmente peligrosa y de esta forma actuar con el conocimiento y experiencia otorgada gracias al registro de las condiciones medioambientales.

Bibliografía

- Adafruit. (n.d.). *Touchscreen for Raspberry Pi*. Retrieved Agosto 2015
- Adrian Holovaty, J. K. (2015). *La guía definitiva de Django, desarrolla aplicaciones WEB de forma rápida y sencilla*. Django Software Corporation.
- Andres Marzal, I. G. (2009). *Introducción a la programación con python*. Universitat Jume.
- Bosch Sensortec. (n.d.). *BMP180*. Retrieved marzo 2015, from <http://cdn.sparkfun.com/datasheets/Sensors/Pressure/BMP180.pdf>
- Djangoproject. (n.d.). *Using Django*. Retrieved marzo 2015, from <https://docs.djangoproject.com/en/1.8/topics/>
- D-Robotics. (n.d.). *DHT11*. Retrieved marzo 2015, from <http://www.micropik.com/PDF/dht11.pdf>
- Duque, R. G. (2008). *Python para todos*.
- Electfreaks. (s.f.). *3G shield*. Recuperado el octubre de 2015, de http://www.electfreaks.com/wiki/index.php?title=3G_shield
- Electfreaks. (n.d.). *HC-SR04 Ultrasonic Module User Guide*. Retrieved marzo 2015, from http://www.electfreaks.com/store/download/product/Sensor/HC-SR04/HC-SR04_Ultrasonic_Module_User_Guide.pdf
- elinux. (n.d.). *RPi Distributions*. Retrieved Enero 2016, from elinux.org/RPi_Distributions
- elinux. (s.f.). *Rpi Low-level peripherals*. Recuperado el Julio de 2015, de elinux.org/Rpi_Low-level-peripherals
- Gay, W. (2014). *Raspberry Pi hardware reference*. Apress.
- Highcharts. (n.d.). *Highcharts*. Retrieved mayo 2015, from <http://www.highcharts.com/>
- Kleback, M. (n.d.). *Tutorial: Raspberry Pi GPIO Pins and Python*. Retrieved Julio 2015, from makezine.com/projects/tutorial-raspberry-pi-gpio-pins-and-python/
- Kuan, J. (2012). *Learning highcharts*. Packt publishing.
- Maxim Integrated. (n.d.). *DS18B20*. Retrieved marzo 2015, from <http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>
- Monk, S. (2010). *Raspberry Pi cookbook*. O'Reilly.
- Omnivision. (n.d.). *OV7670*. Retrieved octubre 2015, from <http://www.voti.nl/docs/OV7670.pdf>

- Point, T. (2013). *MySQL tutorial*. Tutorials Point.
- Point, T. (2014). *Bootstrap tutorial*. Tutorials Point.
- Realpython. (n.d.). *Getting Started with Bootstrap 3*. Retrieved abril 2015, from <https://realpython.com/blog/python/getting-started-with-bootstrap-3/>
- Reitz, K. (n.d.). *Requests Documentation*. 2016.
- Sarker, D. M. (2014). *Python network programming cookbook*. Packt publishing.
- Sparkfun. (n.d.). *Weather Sensor Assembly*. Retrieved marzo 2015, from <https://www.sparkfun.com/datasheets/Sensors/Weather/Weather%20Sensor%20Assembly.pdf>
- Tracker GPS con raspberry Pi*. (n.d.). Retrieved mayo 2015, from <http://fpaez.com/tracker-gps-con-raspberry-pi/>
- Ublox. (n.d.). *GPS.G6-HW-09005*. Retrieved abril 2015, from [https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_\(GPS.G6-HW-09005\).pdf](https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf)
- Ubuntu. (n.d.). *CronHowTo*. Retrieved enero 2016, from <https://help.ubuntu.com/community/CronHowto>
- Ubuntu documentation. (n.d.). *CronHowTo*. Retrieved noviembre 2015, from <https://help.ubuntu.com/community/CronHowto>
- UNESCO - CEPREDENAC. (2012). *Sistema de Alerta Temprana, 10 preguntas y 10 respuestas para la Comunidad Educativa de Panamá*. San José: UNESCO. Recuperado el Noviembre de 2015, de unesdoc.unesco.org/images/0022/002275/227596S.pdf
- Velasco S., R. F. (n.d.). A computer-assisted experiment for the measurement of the temperature dependence of the speed of sound in air.
- w3schools. (n.d.). *CSS tutorial*. Retrieved abril 2015, from <http://www.w3schools.com/css/default.asp>

Anexos

Anexo A: Cuadro desagregado de precios

Este cuadro muestra un estimado del costo de cada elemento utilizado en la estación de alerta temprana y el total de ellos.

Descripción	Precio unitario
Pines hembra soldar	\$2,50
Pines macho soldar	\$2,50
Regulador voltaje 3,3v	\$2,15
Buzzer	\$2,50
Kit header arduino	\$2,75
Cable ribbon GPIO	\$3,50
Raspberry Pi	\$57,00
Conector macho 13x2	\$2,75
Sensor presión BMP180	\$12,50
Sensor temperatura DS18B20	\$6,50
Expansor GPIO	\$8,00
Sensor humedad DHT11	\$13,00
GPS	\$23,00
3G Shield	\$79,00
Cámara OV7670	\$18,00
Cable expansion cámara	\$2,00
LCD táctil 3,5"	\$55,00
Sensor Ultrasonido HC-SRF04	\$5,00
Pluviometro WS2300	\$18,00
Placa cobre 20x20	\$9,00
Batería 7AH 12v	\$15,00
Ventilador 5v	\$4,50
Regulador 7805	\$2,25
Multiplexor CD4052	\$1,80
Micro SD 4GB	\$5,50
Resistores varios	\$1,00
Chasis	\$20,00
Total	\$374,70

TABLA 17 CUADRO DE PRECIOS

Anexo B: REQUESTS

Requests es una biblioteca de HTTP para Python, que permite enviar peticiones HTTP / 1.1, sin la necesidad de mano de obra. No hay necesidad de añadir manualmente las cadenas de consulta a las URL, o para formar o codificar sus datos POST. Keep-alive y la agrupación de conexiones HTTP son 100% automáticas.

Características

- Los dominios y direcciones URL Internacionales
- Mantenimiento de conexiones y agrupación de conexiones
- Las sesiones con persistencia de cookies
- Verificación SSL de estilo navegador
- Autenticación básica / Digest
- Elegantes cookies clave / valor
- Descompresión automática
- Decodificación automática de contenido
- Campos de respuesta Unicode
- Subida de archivos con copias
- HTTP (S) Soporte de proxy
- Tiempos de espera de conexión
- Streaming de descargas
- Soporte .netrc
- solicitudes fragmentadas
- Hilos de seguridad

Anexo C Wvdial

Wvdial es una utilidad que ayuda a realizar conexiones a Internet basadas en módem y que se incluye en algunas distribuciones de GNU/Linux importantes.

Wvdial es un marcador de Protocolo Punto-a-Punto: marca con un módem y comienza pppd en orden a conectar a Internet. La conexión comenzada con wvdial se puede dejar caer volviendo al terminal desde el que se comenzó y presionando Ctrl-C. Wvdial utiliza la biblioteca wvstreams.

Cuando comienza wvdial, primero carga su configuración de `/etc/wvdial.conf` que contiene la información básica sobre el puerto del módem, velocidad, y cadena init de inicialización, así como información sobre el ISP, como el número de teléfono, nombre de usuario y contraseña.

Para la comunicación 3G se utiliza un chip SIM de la compañía movistar por lo que la configuración del archivo `wvdial.conf` tiene la configuración del punto de acceso que la compañía proporciona.

```
[Dialer Defaults]
Modem = /dev/ttyUSB2
Modem Type = Analog Modem
Baud = 115200
ISDN = 0
Stupid mode = 1
Idle Seconds = 0
Init2 = ATZ
Init4 = AT+CGDCONT=1,"IP","internet.movistar.sv"
Phone = *99#>
Password = movistarsv
Username = movistarsv
Dial Command = ATD
```


Anexo D Protocolo de comunicación SPI

El Bus SPI (del inglés Serial Peripheral Interface) es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. El bus de interfaz de periféricos serie o bus SPI es un estándar para controlar casi cualquier dispositivo electrónico digital que acepte un flujo de bits serie regulado por un reloj (comunicación síncrona).

Incluye una línea de reloj, dato entrante, dato saliente y un pin de chip select, que conecta o desconecta la operación del dispositivo con el que uno desea comunicarse. De esta forma, este estándar permite multiplexar las líneas de reloj.

Muchos sistemas digitales tienen periféricos que necesitan existir pero no ser rápidos. La ventaja de un bus serie es que minimiza el número de conductores, pines y el tamaño del circuito integrado. Esto reduce el costo de fabricar, montar y probar la electrónica. Un bus de periféricos serie es la opción más flexible cuando se tienen tipos diferentes de periféricos serie. El hardware consiste en señales de reloj, data in, data out y chip select para cada circuito integrado que tiene que ser controlado. Casi cualquier dispositivo digital puede ser controlado con esta combinación de señales. Los dispositivos se diferencian en un número predecible de formas. Unos leen el dato cuando el reloj sube otros cuando el reloj baja. Algunos lo leen en el flanco de subida del reloj y otros en el flanco de bajada. Escribir es casi siempre en la dirección opuesta de la dirección de movimiento del reloj. Algunos dispositivos tienen dos relojes. Uno para capturar o mostrar los datos y el otro para el dispositivo interno.

El SPI es un protocolo síncrono. La sincronización y la transmisión de datos se realizan por medio de 4 señales:

- SCLK (Clock): Es el pulso que marca la sincronización. Con cada pulso de este reloj, se lee o se envía un bit. También llamado TAKT (en Alemán).
- MOSI (Master Output Slave Input): Salida de datos del Master y entrada de datos al Slave. También llamada SIMO.
- MISO (Master Input Slave Output): Salida de datos del Slave y entrada al Master. También conocida por SOMI.
- SS/Select: Para seleccionar un Slave, o para que el Master le diga al Slave que se active. También llamada SSTE.

Anexo E Protocolo de comunicación I2C

I²C es un bus de comunicaciones en serie. Su nombre viene de Inter-Integrated Circuit (Circuitos Inter-Integrados). La versión 1.0 data del año 1992 y la versión 2.1 del año 2000, su diseñador es Philips. La velocidad es de 100 kbit/s en el modo estándar, aunque también permite velocidades de 3.4 Mbit/s. Es un bus muy usado en la industria, principalmente para comunicar microcontroladores y sus periféricos en sistemas integrados (Embedded Systems) y generalizando más, para comunicar circuitos integrados mutuamente que normalmente residen en un mismo circuito impreso.

La principal característica de I²C es que utiliza dos líneas para transmitir la información: una para los datos y otra para la señal de reloj. También es necesaria una tercera línea, pero esta sólo es la referencia (masa). Como suelen comunicarse circuitos en una misma placa que comparten una misma masa esta tercera línea no suele ser necesaria.

Las líneas se llaman:

- SDA: datos
- SCL: reloj
- GND: tierra

Las dos primeras líneas son drenador abierto, por lo que necesitan resistencias de pull-up.

Los dispositivos conectados al bus I²C tienen una dirección única para cada uno. También pueden ser maestros o esclavos. El dispositivo maestro inicia la transferencia de datos y además genera la señal de reloj, pero no es necesario que el maestro sea siempre el mismo dispositivo, esta característica se la pueden ir pasando los dispositivos que tengan esa capacidad. Esta característica hace que al bus I²C se le denomine bus multimaestro.

Las transacciones en el bus I2C tienen este formato:

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

- El bus está libre cuando SDA y SCL están en estado lógico alto. En estado bus libre, cualquier dispositivo puede ocupar el bus I²C como maestro.
- El maestro comienza la comunicación enviando un patrón llamado "start condition". Esto alerta a los dispositivos esclavos, poniéndolos a la espera de una transacción.
- El maestro se dirige al dispositivo con el que quiere hablar, enviando un byte que contiene los siete bits (A7-A1) que componen la dirección del dispositivo esclavo con

el que se quiere comunicar, y el octavo bit (A0) de menor peso se corresponde con la operación deseada (L/E), lectura=1 (recibir del esclavo) y escritura=0 (enviar al esclavo).

- La dirección enviada es comparada por cada esclavo del bus con su propia dirección, si ambas coinciden, el esclavo se considera direccionado como esclavo-transmisor o esclavo-receptor dependiendo del bit R/W.
- El esclavo responde enviando un bit de ACK que le indica al dispositivo maestro que el esclavo reconoce la solicitud y está en condiciones de comunicarse.
- Seguidamente comienza el intercambio de información entre los dispositivos.
- El maestro envía la dirección del registro interno del dispositivo que se desea leer o escribir.
- El esclavo responde con otro bit de ACK
- Ahora el maestro puede empezar a leer o escribir bytes de datos. Todos los bytes de datos deben constar de 8 bits, el número máximo de bytes que pueden ser enviados en una transmisión no está restringido, siendo el esclavo quien fija esta cantidad de acuerdo a sus características.
- Cada byte leído/escrito por el maestro debe ser obligatoriamente reconocido por un bit de ACK por el dispositivo maestro/esclavo.
- Se repiten los 2 pasos anteriores hasta finalizar la comunicación entre maestro y esclavo.
- Aun cuando el maestro siempre controla el estado de la línea del reloj, un esclavo de baja velocidad o que deba detener la transferencia de datos mientras efectúa otra función, puede forzar la línea SCL a nivel bajo. Esto hace que el maestro entre en un estado de espera, durante el cual, no transmite información esperando a que el esclavo esté listo para continuar la transferencia en el punto donde había sido detenida.
- Cuando la comunicación finaliza, el maestro transmite una "stop condition" para dejar libre el bus.
- Después de la "stop condition", es obligatorio para el bus estar inactivo durante unos microsegundos.

El código del kernel de Linux para el soporte I2C está separado en varias piezas lógicas:

- I2C chip driver (maneja uno de los chips conectados al bus I2C, tanto si se comporta como maestro o como esclavo)
- I2C bus driver
- I2C algorithm driver

- I2C core (la parte genérica del subsistema de I2C)

Anexo F Protocolo de comunicación UART

UART, son las siglas en inglés de Universal Asynchronous Receiver-Transmitter, en español: Transmisor-Receptor Asíncrono Universal, es el dispositivo que controla los puertos y dispositivos serie. Se encuentra integrado en la placa base o en la tarjeta adaptadora del dispositivo.

Un UART dual, o DUART, combina dos UART en un solo chip. Existe un dispositivo electrónico encargado de generar la UART en cada puerto serie. La mayoría de las computadoras modernas utilizan el chip UART 16550, que soporta velocidades de transmisión de hasta 921,6 Kbps (Kilobits por segundo). Las funciones principales de chip UART son: manejar las interrupciones de los dispositivos conectados al puerto serie y convertir los datos en formato paralelo, transmitidos al bus de sistema, a datos en formato serie, para que puedan ser transmitidos a través de los puertos y viceversa.

Transmisión y recepción de datos serie

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

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

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

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

Anexo G Sentencias NMEA

La National Marine Electronics Association (NMEA) es una organización de comercio electrónico estadounidense que establece estándares de comunicación entre electrónica marina.

Estándares NMEA

- NMEA 0180
- NMEA 0182
- NMEA 0183
- NMEA 2000
- NMEA OneNote
- NMEA 0183

NMEA 0183 es una especificación combinada eléctrica y de datos entre aparatos electrónicos marinos y, también, más generalmente, receptores GPS.

El protocolo NMEA 0183 es un medio a través del cual los instrumentos marítimos y también la mayoría de los receptores GPS pueden comunicarse los unos con los otros. Ha sido definido, y está controlado, por la organización estadounidense National Marine Electronics Association.

Anexo H comandos AT

Los comandos AT son instrucciones codificadas que conforman un lenguaje de comunicación entre el hombre y un terminal modem.

En un principio, el juego de comandos AT fue desarrollado en 1977 por Dennis Hayes como un interfaz de comunicación con un modem para así poder configurarlo y proporcionarle instrucciones, tales como marcar un número de teléfono. Más adelante, con el avance del baudío, fueron las compañías Microcomm y US Robotics las que siguieron desarrollando y expandiendo el juego de comandos hasta universalizarlo. Los comandos AT se denominan así por la abreviatura de *attention*.

Aunque la finalidad principal de los comandos AT es la comunicación con modems, la telefonía móvil GSM también ha adoptado como estándar este lenguaje para poder comunicarse con sus terminales. De esta forma, todos los teléfonos móviles GSM poseen un juego de comandos AT específico que sirve de interfaz para configurar y proporcionar instrucciones a los terminales. Este juego de instrucciones puede encontrarse en la documentación técnica de los terminales GSM y permite acciones tales como realizar llamadas de datos o de voz, leer y escribir en la agenda de contactos y enviar mensajes SMS, además de muchas otras opciones de configuración del terminal.

Queda claro que la implementación de los comandos AT corre a cuenta del dispositivo GSM y no depende del canal de comunicación a través del cual estos comandos sean enviados, ya sea cable de serie, canal Infrarrojos, Bluetooth, etc. De esta forma, es posible distinguir distintos teléfonos móviles del mercado que permiten la ejecución total del juego de comandos AT o sólo parcialmente.

Notación de los comandos AT

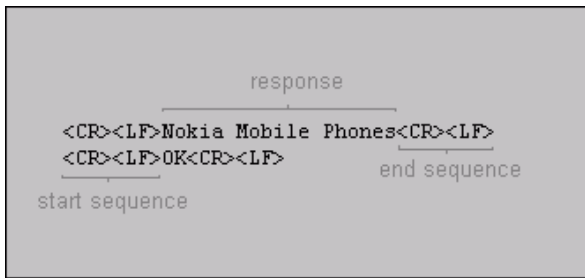
El envío de comandos AT requiere la siguiente estructura:

•Petición:



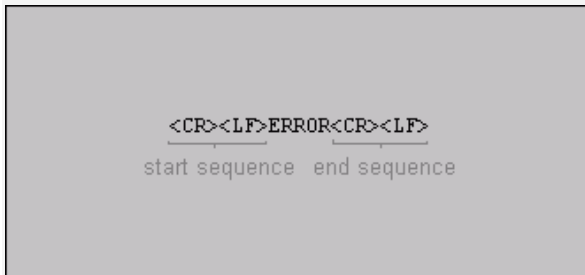
`<CR>` ... *Carriage return*

• Respuesta correcta:



<CR> ... Carriage return
<LF> ... Line feed

• **Respuesta incorrecta:**



<CR> ... Carriage return
<LF> ... Line feed