UNIVERSIDAD NACIONAL DE INGENIERIA

FACULTAD DE INGENIERIA ELECTRICA Y ELECTRONICA

DISEÑO E IMPLEMENTACIÓN DE UNA ESTACIÓN METEOROLÓGICA UTILIZANDO MICRO CONTROLADOR ARDUINO-RASPBERRY PI CON RADIO ENLACE

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ELECTRÓNICO

ELABORADO POR:

ALAND BRAVO VECORENA

ASESOR

CÉSAR AUGUSTO ATALAYA PISCO

LIMA - PERU

2016

DISEÑO E IMPLEMENTACIÓN DE UNA ESTACIÓN METEOROLÓGICA UTILIZANDO MICRO CONTROLADOR ARDUINO-RASPBERRY PI CON RADIO ENLACE

DEDICATORIA:

A Dios, a mis padres Darwin Oliver y Rosa Luz y a mis hermanos Evelyn Norka y Darwin.

AGRADECIMIENTO:

Quiero expresar mi gratitud a los docentes de ante grado de mi casa de estudios, por su apoyo en el enfoque de ingeniería en la presente tesis. Asimismo a mis alumnos y colegas de trabajo de la Universidad de Huánuco (UDH) y de la Universidad Nacional Hermilio Valdizán de Huánuco (UNHEVAL) por su entusiasmo y aliento constante. También mi agradecimiento al **Ing. César Rosas Echevarría** y al **Ing. Jaime Nuñez Mosqueira** por las largas y amenas horas de investigación. A todos los amigos que tuve y retuve, especialmente a aquellos que aún perduran en el tiempo y espacio.

SUMARIO

En la Tesis se diseña y construye un prototipo de estación meteorológica, utilizando micro controladores embebidos Arduino y RaspberryPi con interface web y base de datos, con enfoque de tecnología SCADA-WEB vía un radio enlace y un sistema integrado de panel solar con batería. En el proceso de ejecución del proyecto de Tesis, se han analizado diversas alternativas de solución a nivel de hardware y software, optando por el uso de tecnologías de código abierto, tales como: C++ para el micro controlador Arduino, base de datos SQLite, PHP para la página web y Python para la integración con la interface de adquisición de sensores y de la integración con el servidor OPC en el RaspberryPi (ver [1], [2] y [3]).

Se logró diseñar el radio enlace con tecnología Ubiquiti punto a punto, con la finalidad de contar con un sistema en tiempo real con una cobertura de hasta 15 Km. Desde el punto de vista del método de proceso de diseño de ingeniería, se realizó la validación de las especificaciones técnicas del prototipo. Asimismo, se verificó la consistencia de la estructura de datos utilizando datos reales recopilados del SENAMHI, dando aportes en el tema de integración tecnológica para una estación meteorológica. Se efectuó las pruebas de monitoreo de variables meteorológicas que han sido implementados en el proyecto de investigación, lográndose obtener los resultados esperados en base a la solución planteada, con el propósito de establecer una base tecnológica para un modelo de negocio de estaciones meteorológicas experimentales para investigación académica.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO DE TÉRMINOS

Término	Definición
SE	Software Engineering o Ingeniería de Software.
SCADA	Sistema de Supervisión, Control y Adquisición de Datos.
UML	Lenguaje de Modelado Unificado.
TIC	Tecnología de Información y Comunicación.
PHP	Lenguaje script de propósito general.
JVM	Máquina Virtual de Java.
W3C	World Wide Web Consortium.
TCP/IP	Protocolo de Control de Transmisión / Protocolo de Internet.
IEEE	Instituto de Ingenieros Electricistas y Electrónicos.
TIC	Tecnologías de la información y la comunicación.
TCP	Protocolo de Control de Transmisión.
IP	Protocolo de Internet.
OMM	Organización Meteorológica Mundial.
SQLite	Motor ligero de base de datos SQL.
SENAMHI	Servicio Nacional de Meteorología e Hidrología del Perú.
UDH	Universidad de Huánuco.
UNHEVAL	Universidad Nacional Hermilio Valdizán de Huánuco.
NanoBeam M5	Marca de un equipo de radio enlace punto a punto modelo M5.
MATLAB	Entorno de simulación de “Laboratorio de Matrices”
CONCYTEC	Concejo Nacional de Ciencia, Tecnología e Innovación Tecnológica.
PHP	Lenguaje de Pre procesador de Hipertexto.
Python	Lenguaje de programación de Phyton.
Matrikon	Fabricante de sistemas SCADA.
OPC	Estándar de Comunicación de Plataforma Abierta.

INTRODUCCIÓN

El propósito de la Tesis ha sido el diseño y construcción de una Estación Meteorológica, utilizando plataformas de código abierto mediante las tecnologías de Arduino-RaspberryPi para recolectar y digitalizar los datos de sensores de: temperatura, humedad, presión, ultravioleta, monóxido de carbono, dióxido de carbono, velocidad y dirección de viento y lluvia, de modo tal que permita personalizar la solución para incluir un radio enlace punto a punto de 15 Km, integrado a un panel solar con batería, conjuntamente con una base de datos SQLite, una interface de usuario web con PHP y la integración scada con Python.

El método de trabajo utilizado para la Tesis ha sido el método de diseño de ingeniería, con el cual se ha verificado las especificaciones técnicas de la solución planteada. Los alcances y limitaciones del estudio de la Tesis se circunscriben a un prototipo de una estación meteorológica, el cual ha sido desarrollado con éxito, sirviendo como punto de partida para una siguiente etapa de innovación, a nivel de producto final en la etapa de comercialización y venta, cuyo alcance no forma parte de la tesis. Para el desarrollo del presente trabajo se ha diseñado e implementado un prototipo en hardware y software de una estación meteorológica hecho a la medida, que reduce costos respecto a una estación comercial, que tiene como propósito establecer una base tecnológica para un modelo de negocio comercial de estaciones meteorológicas experimentales para investigación académica.

En el contenido del **Capítulo I** se desarrolla el análisis y descripción del problema, tales como: descripción del problema, objetivos de la tesis, evaluación del problema, limitaciones de la tesis y una breve síntesis de la tesis; el **Capítulo II** se refiere al marco teórico, donde se enfatiza los antecedentes y definiciones conceptuales, estudio de las alternativas de solución: arquitecturas de hardware de diversos fabricantes, componentes de software a nivel de integración web y scada, así como la formulación de las especificaciones técnicas; el **Capítulo III** comprende el diseño y construcción hardware-software del prototipo, así como la verificación de las especificaciones técnicas,

tales como: mensajería y comunicación de señales meteorológicas, algoritmos de predicción meteorológica, interoperabilidad web y scada; el **Capítulo IV,** trata sobre los resultados de la tesis, sus implicancias, así como las conclusiones y recomendaciones para trabajos futuros.

Finalmente, he querido dedicar algunas líneas para reconocer a personas e instituciones que han brindado su ayuda durante el desarrollo de la Tesis, me refiero a los alumnos y docentes colegas de la Universidad de Huánuco (UDH), en la persona del Ing. Jaime Nuñez Mosqueira, a los docentes colegas de la Universidad Nacional Hermilio Valdizán de Huánuco (UNHEVAL), al Ing. Cesar Rosas Echevarría, así como a los docentes de ante grado de la Universidad Nacional de Ingeniería (UNI) por su apoyo en el enfoque de ingeniería de esta Tesis.

CAPÍTULO I
ANÁLISIS Y DESCRIPCIÓN DEL PROBLEMA

Actualmente se vive una era Digital y del Conocimiento, en donde el capital humano debe jugar un papel importante para el desarrollo de las naciones y en especial de los países en vías de desarrollo como el nuestro, sin dejar de tener en cuenta que existe un mayor acceso a información que permite la investigación y desarrollo de nuevas tecnologías, hecho que debe ser aprovechado de la mejor manera para suscitar nuestro tan ansiado desarrollo. Bajo este contexto, la emergencia de los Sistemas Embebidos como parte de los Sistemas de Información, está cada día formando parte de las aplicaciones de gestión ambiental y de los recursos naturales. Es en este ámbito donde entra a tallar la integración de tecnologías y arquitecturas de ingeniería más complejas, para brindar un adecuado monitoreo y supervisión de variables ambientales.

1.1 Descripción del Problema

Existe necesidad de contar con estaciones meteorológicas experimentales para investigación académica en el territorio nacional, de modo que permita brindar una adecuada cobertura y calidad en la prestación de los servicios meteorológicos a los centros poblados del país.

1.2 Objetivo de la Tesis

El objetivo de la tesis es desarrollar el proceso de diseño de ingeniería de una estación meteorológica, que sirva como base tecnológica para un modelo de negocio comercial de estaciones meteorológicas experimentales para investigación académica.

1.3 Evaluación del Problema

En la actualidad el Gobierno planea ejecutar una mayor cobertura del servicio de gestión ambiental y de recursos naturales, manteniendo el presupuesto en este sector, motivo por el cual se hace necesario el ahorro de costos operativos, que permita reutilizar eficientemente los recursos del personal técnico y administrativo de las instituciones

involucradas en el tema, tales como el Ministerio del Ambiente MINAM, el Servicio Nacional de Hidro meteorología SENAMHI, Gerencia Regional de Recursos Naturales y Gestión Ambiental GRRNGA-GRH, Ministerio de Agricultura y Riego MINAG, entre otros. El diseño de la estación meteorológica, deberá contemplar los siguientes aspectos:

- Eficiente uso de los recursos informáticos. Debe tener un servidor web con base de datos que almacene y dé acceso a las señales meteorológicas digitalizadas, que permita la creación de un servicio de monitoreo remoto.
- Colaboración en la emisión del monitoreo y pronóstico ambiental. Debe contener múltiples usuarios para colaborar en un único pronóstico meteorológico, al tener un sistema digital de datos ambientales accesibles vía web.
- Esquema computacional de interoperabilidad sobre el estándar OPC Scada-DA, convirtiendo el contenido de la información meteorológica en una entrada digital que puede comunicarse con otros servidores o estaciones automáticas scada.

1.4 Limitaciones de la Tesis

El trabajo de investigación se restringe a un prototipo, cuyas limitaciones son las siguientes:

- El enfoque de los conceptos teóricos necesarios para la investigación aplicada han sido adecuados de modo tal, que permita esbozar con claridad los aportes del trabajo de experimentación materializados en la Tesis.
- No se ha cubierto todas las variantes de los algoritmos de pronóstico solo se ha cubierto una de sus variantes, en concreto el análisis de regresión lineal y múltiple.
- No se ha pretendido construir un servidor de aplicación web PHP y scada Python con una base de datos en un micro controlador, por el contrario se utiliza Apache y OpenOPC sobre Linux en el Raspberry-Pi/Arduino, para ahorrar tiempo en construir un sistema operativo desde cero.
- No se ha pretendido solucionar todos los problemas de digitalización de información de análisis meteorológico en el sector ambiental, sólo se ha demostrado la viabilidad técnica de ingeniería, así como las ventajas y desventajas en caso se utilice la tecnología para implementar el analizador computacional basado en los algoritmos de pronóstico con interface usb-rs232-bluetooth para micro controladores.

1.5 Síntesis de la Tesis

En esta tesis se ha realizado una investigación a nivel de hardware y software de una estación meteorológica a medida, para lo cual se hizo uso del método del proceso de diseño de ingeniería, sobre la cual se realizó el diseño electrónico simulado en Proteus

(ISIS-ARES) y construcción del prototipo del módulo de adquisición de datos vía los sensores de Temperatura (DS18B20), Humedad (HIH-4000), Presión (MPX4250A), Ultravioleta (ML8511), Monóxido (MQ7), CO2 (MG-811), Velocidad y Dirección de Viento, e intensidad de Lluvia (BH4TDV). Para el almacenamiento de los datos se utilizó la base de datos SQLite y para la publicación de los datos se utilizó el servidor web-scada con PHP/Python sobre el sistema operativo Linux embebido en la tarjeta RaspberryPi. Para la comunicación remota, se diseñó un radio enlace punto a punto sobre NanoBeam M5 Ubiquiti, integrado a un sistema de Panel Solar y Batería.

CAPÍTULO II
MARCO TEÓRICO

En este capítulo se presentan: los antecedentes, las definiciones conceptuales, así como las alternativas de solución al problema, para establecer las especificaciones técnicas de la estación meteorológica experimental. El marco teórico proporciona un conjunto de elementos que trabajan por separados, tales como: base de datos SQLite, Arduino, Raspberry Pi, sensores, sistema de energizado, sistema de radio enlace, etc. el resultado es un prototipo que amalgama gran parte de estos elementos en una estación meteorológica, con componentes de hardware y software con un flujo de información entre ambos componentes, que presenta al usuario una interface gráfica amigable y de fácil uso, que difiere de la pantalla de texto plano a nivel de micro controladores y el monitoreo serial-usb, al poder guardar datos en una base de datos, el sistema puede realizar una visualización sobre los datos históricos, así como un monitoreo en tiempo real de los datos actuales. Asimismo, se podrían desarrollar algoritmos de pronóstico avanzado utilizando un grupo de estaciones meteorológicas experimentales basado en pisos altitudinales y cubrir en cierta manera el déficit de información meteorológica para ajustar y mejorar los pronósticos actuales.

2.1 Antecedentes y Definiciones Conceptuales

Este tema de Tesis es fruto de una investigación realizado por el autor en la ciudad de Huánuco, durante el periodo Enero-2014 hasta octubre-2015, lográndose por ese entonces lo siguiente:

- Se ha revisado artículos y publicaciones relacionadas con el tema de la hidro meteorología y su aplicación en cuanto a técnicas de pronóstico se refiere. Se ha estudiado algunas variantes de los algoritmos usando R/MATLAB.
- Se ha investigado a fondo el conjunto de tecnologías C++, PYTHON, PHP y SQLite en Linux embebido sobre Raspberry-Pi/Arduino. Se ha logrado realizar interfaces gráficas web-scada usando herramientas de Software Libre.
- Se ha investigado a fondo las técnicas de ingeniería de software con énfasis en los

patrones de diseño, a través del framework PHP y PYTHON para establecer las pruebas de inter operatividad entre los módulos de hardware y software.

- Se ha tomado contacto con instituciones de gobierno relacionadas al tema, tales como el SENAMHI - Sede Huánuco, quienes ilustraron abundantemente en los aspectos hidro-meteorológicos que intervienen en el pronóstico de tiempo.

Desde el punto de vista científico, el sistema climático puede ser dividido en seis componentes [26]: atmósfera, hidrósfera, criosfera, superficie terrestre, biósfera y la antropósfera. Un modelo climático completo contiene descripciones físicas de cada uno de los componentes descritos y toma en consideración sus acoplamientos. Los primeros modelos climáticos fueron desarrollados para la predicción del clima a partir del año 1940, donde los procesos atmosféricos y de circulación, fueron las bases del desarrollo del modelo climático. Fue **Vilhelm Bjerknes** (1862 – 1951) el primero en notar que la predicción del clima era un problema de matemática y física, de modo que las ecuaciones de conservación de la masa, momento y energía, tenían que ser formulados para poder calcular la dinámica de circulación del clima. Estos son combinados con una ecuación de estado para un gas ideal, de modo que la atmósfera, evoluciona de una manera determinística, lo cual implica que dichos estados consecutivos del sistema, son enlazados mediante las leyes de la física. **Bjerknes** supone que un conocimiento suficientemente preciso de las leyes básicas y las condiciones iniciales, eran necesarios y suficientes para una predicción. Por lo tanto, adoptó la noción clásica de la previsibilidad de la naturaleza o el determinismo, desde el enfoque de **Laplace**. Sólo más tarde se hará evidente, sobre todo a través de la labor de **Edward Lorenz** en 1963, que la previsibilidad de la evolución de un sistema no lineal, en este caso: la circulación atmosférica, es de naturaleza limitada. **Bjerknes** fundó la "Escuela de Bergen" de la meteorología y ha producido contribuciones innovadoras al conocimiento de la ciclogénesis.

También es importante resaltar la contribución de **Lewis Fry Richardson** (1881-1953), quien fue el primero en formular y procesar un pronóstico numérico del tiempo. Los cálculos, se llevaron a cabo en el año 1917, los cuales se basaron en los datos de observación de 12 perfiles verticales de presión y temperatura en diferentes lugares de toda Europa, que fueron establecidos previamente por **Bjerknes**. **Richardson** publicó su resultado en el año 2007, con el famoso libro “Predicción del tiempo mediante Procesos Numéricos”. El problema era que las condiciones iniciales, en este caso, los datos de la presión superficial, contenían pequeños errores que se multiplicaron durante el procedimiento numérico y dieron lugar a fuertes tendencias con la presión. Esto apunta al

hecho de que las condiciones iniciales o la inicialización de tiempo y los modelos climáticos, es un problema de condiciones iniciales, de los cuales el modelador debe ser siempre consciente. No sólo las condiciones iniciales, sino también la formulación de ecuaciones de conservación es crucial, incluso los datos iniciales más precisos habrían conducido a la inestabilidad mediante las ecuaciones de **Richardson**, debido a que contenían los procesos físicos (olas gravitacionales), que desestabilizan la solución y hacen imposible una predicción a largo plazo.

Carl-Gustaf Rossby (1898-1957) también logró un gran avance al darse cuenta de que la conservación de la vorticidad era una limitación más robusta que el de los momentos. Este enfoque es adecuado para el sistema de la Tierra en rotación, ya que el efecto de Coriolis puede ser implementado de una manera natural. Las ondas planetarias (ondas de Rossby) aparecen en fluidos de rotación como la atmósfera y el océano. Por lo tanto la atmósfera y el océano responden a las perturbaciones (anomalías de la temperatura, inicio de la formación de aguas profundas, etc.) con la propagación de las ondas de Rossby que causan las corrientes que luego son capaces de modificar los estados del fondo. Las ondas de Rossby son fundamentales para la comprensión de los sistemas de tiempo en la atmósfera y la circulación a gran escala en el océano.

En los años 1940 y 1950 aparece el primer ordenador (**ENIAC, Electronic Numerical Integrator and Computer**), el cual se desplegó en Princeton para el Ejército de los Estados Unidos. El primer proyecto de meteorología, fue la predicción de una oleada de la tormenta en la costa Este americana. En 1955, el primer modelo integrado de circulación atmosférica simplificado de largo plazo, fue realizado por **Norman Phillips** (Phillips 1956). Esto marcó el comienzo de los modelos de circulación general que permitieron resolver las ecuaciones completas de flujo atmosférico. Además de la complejidad numérica, los estudios teóricos sobre los fundamentos del comportamiento dinámico de la atmósfera y el océano fueron avanzando notablemente. Entre estos, la conservación del momento y vorticidad en un fluido de rotación, implica términos no lineales en el sistema de ecuaciones. Además, en un contexto giratorio, tales como la Tierra, la fuerza de Coriolis provoca un acoplamiento de los componentes de los movimientos horizontales. Las no linealidades también son responsables de la previsibilidad finita de tal flujo como lo descubierto en el año 1963 por **Edward Lorenz** (1926-2008). En su artículo “Flujo No Periódico Determinístico”, **Lorenz** describe cómo los patrones de flujo a gran escala pueden conducir a un comportamiento caótico, esto condujo al descubrimiento de la Teoría del Caos.

A mediados de la década de 1960, casi 20 años después del desarrollo de los primeros modelos de la circulación en la atmósfera, se formularon modelos tridimensionales del océano (Bryan y Cox 1967). **Syukuro Manabe** (ver [25], [26], [27], [28]) encontró que para la investigación climática y atmosférica, los modelos oceánicos deben combinarse. Esto se consigue mediante el acoplamiento de forma dinámica de dos componentes. El primer modelo acoplado fue desarrollado a finales de la década de 1960 por **Suki Manabe** y sus colegas (Manabe y Bryan, 1969). Una dificultad particular fue las diferentes escalas de tiempo para la atmósfera y el océano. Un problema notorio fue que los flujos de calor y agua necesarios de la atmósfera y el océano, no eran compatibles, esto hizo necesaria la introducción de una corrección de flujo no físico, que se utilizó en la mayoría de los modelos por más de casi 30 años. Desde principios de 1990, se lograron mejoras significativas mediante la incorporación de otros componentes del sistema climático. Los modelos climáticos se han vuelto más completos. El ciclo del carbono, las formulaciones dinámicas de los tipos de vegetación, la química de las hojas de la atmósfera y de hielo, pertenecen a los componentes que se implementan actualmente en los modelos de circulación físicas existentes. En consecuencia, la modelización del clima se ha convertido en una ciencia interdisciplinaria.

Además de los modelos cada vez más detallados, también están siendo desarrollados, los modelos climáticos simplificados. Estos permiten el estudio de los problemas básicos de las ciencias del clima de una forma eficiente. El desarrollo y aplicación de modelos climáticos de complejidad reducida, a menudo llamados EMIC (Modelos del sistema Terrestre de Complejidad Intermedia), han hecho importantes contribuciones a la comprensión del sistema climático, en particular, en la interpretación cuantitativa de reconstrucciones paleo climáticas y simulaciones de conjunto del futuro cambio climático. Resaltan en estos temas, las contribuciones de **Kevin Hamilton**, **Wataru Ohfuchi** [13], Pukh Raj Rakhecha y Vijai P. Singh [17], en las aplicaciones a la hidro meteorología. También es importante recalcar los aportes de **Thomas Stocker** [26] en lo que concierne a los modelos climatológicos contemporáneos. Para mayor detalle de marco teórico, remitirse a las referencias bibliográficas del [11] al [40].

Por la complejidad del proyecto, se ha visto conveniente desarrollar los conceptos tecnológicos más importantes en lo que respecta a la capa de abstracción de hardware y software de la estación meteorológica, que se detalla a continuación:

Plataforma de Comunicación Abierta OPC, utiliza el enfoque cliente servidor para el intercambio de información. Un servidor OPC encapsula las fuentes del proceso de información, tales como: dispositivos PLC, DCS, IEDs y hace disponible la información a través de su interface. El cliente OPC se conecta al servidor OPC para acceder y consumir los datos disponibles. Con este enfoque, las aplicaciones consumen y proporcionan datos, ya sea como cliente o como servidor. En la **Fig. 2.1** se muestra un escenario típico cliente servidor OPC-SCADA. Es importante recalcar que los servidores clásicos OPC están basadas en las tecnologías COM y DCOM de Microsoft. La ventaja de este enfoque ha sido la disminución de trabajo técnico para definir las especificaciones de diversos APIs para diversas aplicaciones especializadas, sin la necesidad de definir un protocolo de comunicación específico. Tanto COM como DCOM proporcionan un mecanismo transparente para que un cliente pueda invocar a métodos en un objeto COM de un servidor, ejecutándose en el mismo proceso, en otro proceso o en otro nodo de red. Usando esta tecnología, se reduce el tiempo de desarrollo para maquinas con sistema operativo Windows.

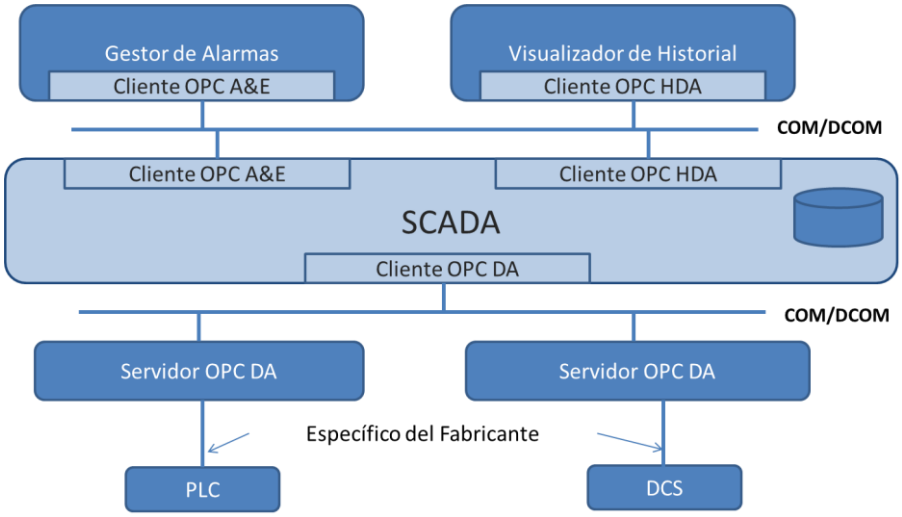


Fig. 2.1 Escenario Cliente Servidor OPC-SCADA.

En contra partida, la desventaja es que DCOM es difícil de configurar y no puede ser utilizado para comunicaciones internas, tampoco es interoperable en otras plataformas como Linux, requiriendo una interface de pasarela o Gateway como en el caso del servidor OPC de MatrikonOPC.

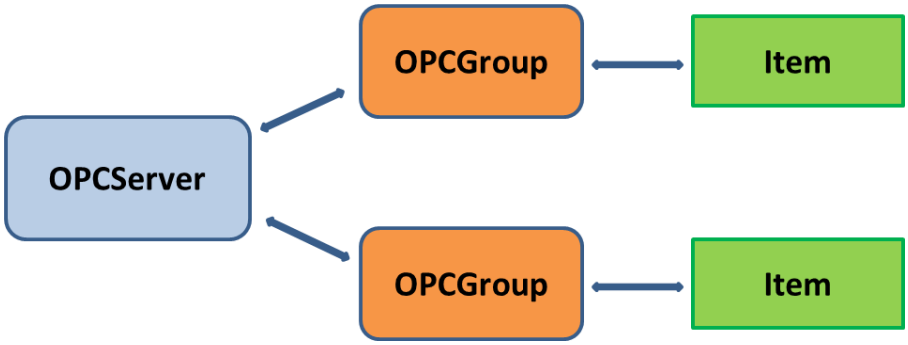


Fig. 2.2 Objetos creados por un Cliente OPC para acceder a los datos.

En la **Fig. 2.2**, se muestra la interface **OPC Data Access** que habilita la lectura, escritura y monitoreo de variables, conteniendo los datos actuales del proceso. El principal uso es de trasladar los datos en tiempo real desde los PLCs, DCSs y otros dispositivos, hacia las HMIs y otros dispositivos clientes. OPC DA es la interfaz más importante de la tecnología OPC. Los clientes OPC DA explícitamente seleccionan las variables (ítems OPC) que desean leer, escribir o monitorear en el servidor. El cliente OPC establece una conexión hacia el servidor mediante la creación de un objeto OPCServer. Este objeto que referencia al servidor desde el cliente, ofrece métodos para navegar a través de las direcciones de espacios jerárquicos, para encontrar los ítems y sus propiedades tales como: el tipo de dato y permisos de acceso. Para acceder a los datos, el Cliente agrupa a los ítems OPC con configuraciones idénticas, tales como el tiempo de actualización en un objeto OPCGroup. Cuando los ítems son agregados a un grupo, éstos pueden ser leídos o actualizados desde un cliente. Sin embargo, la manera preferida para el ciclo de lectura de datos por el cliente, es monitoreando los cambios en los valores de los ítems del Servidor. El cliente define una tasa de actualización en el grupo, conteniendo los ítems de interés. La tasa de actualización es usada en el servidor para la verificación cíclica de modificaciones en los valores. Después de cada ciclo, el servidor solo envía los valores modificados hacia el cliente. El OPC DA proporciona datos en tiempo real que podrían temporalmente no estar disponible, por ejemplo, cuando se interrumpe temporalmente la comunicación hacia un dispositivo. La tecnología del OPC clásico maneja esta situación mediante una casilla de indicador de tiempo y fecha llamado timestamp, así como una casilla de calidad de dato. Este indicador de calidad precisa si la data es exacta (buena), no disponible (mala) o desconocida (incierta). La interface **OPC Alarma y Eventos**, habilita la recepción de la notificación de eventos y alarmas. Los eventos son notificaciones únicas que informan al cliente acerca de la ocurrencia de un evento. Las alarmas son notificaciones que informan al cliente acerca de un cambio de la condición en el proceso.

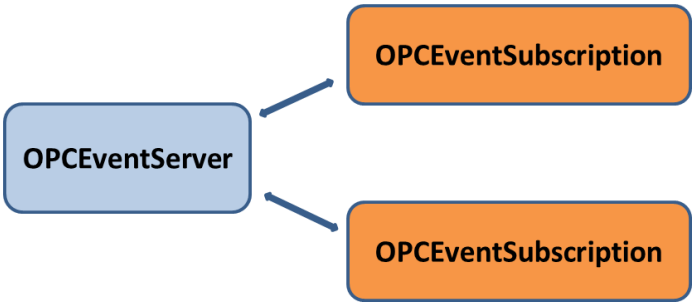


Fig. 2.3 Objetos creados por un Cliente OPC para recibir eventos.

Estas condiciones pueden ser, por ejemplo, el nivel de agua de un tanque. En este ejemplo un cambio en la condición podría ocurrir cuando se alcanza un nivel máximo, o se excede o disminuye por debajo de un nivel mínimo. Varias alarmas pueden incluir como requisito, que sean reconocidas por el usuario. Este reconocimiento también es posible vía la interface OPC A&E. Para recibir notificaciones, el cliente OPC A&E se conecta al servidor, para suscribirse a las notificaciones, y luego recibir todas las notificaciones lanzadas en el servidor. Para limitar el número de notificaciones, el cliente OPC puede especificar ciertos criterios de filtros. El cliente OPC primero se conecta mediante la creación de un objeto OPCEventServer en el servidor OPC A&E, luego genera un uso del servicio OPCEventSubscription para recibir los mensajes de eventos, tal como se aprecia en la **Fig. 2.3**.

El **OPC Historical Data Access**, provee acceso a los datos almacenados. Desde un simple sistema de registro hasta un complejo sistema SCADA, los archivos históricos pueden ser revisados de una manera uniforme. El cliente OPC se conecta mediante la creación del objeto OPCHDAServer en el servidor OPC HDA. Este objeto ofrece todas las interfaces y métodos para leer y actualizar datos históricos. Un segundo objeto OPCHDABrowser es definido para navegar sobre el espacio de direcciones del servidor OPC HDA. Existen tres maneras diferentes para la lectura de los datos históricos: El primer mecanismo lee los datos en bruto del archivo, en el que el cliente define una o más variables y el dominio del tiempo que se quiera leer. El servidor retorna todos los valores archivados durante el tiempo especificado, hasta el número máximo de valores definido por el cliente. El segundo mecanismo lee los valores de una o más variables para las marcas de tiempo específicos. El tercer mecanismo de lectura, calcula los valores agregados de los datos en la base de datos históricos, para el dominio de tiempo especificado para una o más variables. Los valores incluyen siempre la calidad y la marca de tiempo asociada. Además de los métodos de lectura, OPC HAD también define métodos para insertar, reemplazar y eliminar datos en la base de datos históricos.

El **Servidor Web** que utilizaremos en el proyecto será **Apache**, ya que es una distribución de libre acceso, cuya primera versión fue lanzada en el año 1995 y es portable a diversos sistemas operativos: Unix, FreeBSD, Linux, Solaris, Microsoft Windows. La ventaja es que permite múltiples lenguajes script: PHP, Perl, Tlc, Python, también soporta J2EE a través de Tomcat. Otra ventaja es que soporta host virtuales, es decir un mismo servidor físico para varias IPs y nombres de dominio. También permite crear nuevos módulos con el API de módulos de Apache, que trae una configuración sencilla en base a directivas que se editan en los ficheros de configuración: httpd.conf, Access.conf (en Linux), .htaccess, .htpasswd. En la **Fig. 2.4** se visualiza el detalle de la arquitectura empleada en el componente web de la estación meteorológica. El lenguaje de programación PHP, es un lenguaje script que se ejecuta en el servidor y da como resultado una página HTML. Permite generar páginas de contenido dinámico, también permite abrir, leer, escribir y cerrar archivos en el servidor. Permite utilizar bases de datos, formularios, cookies, encriptación y control de acceso de usuarios al sitio web.

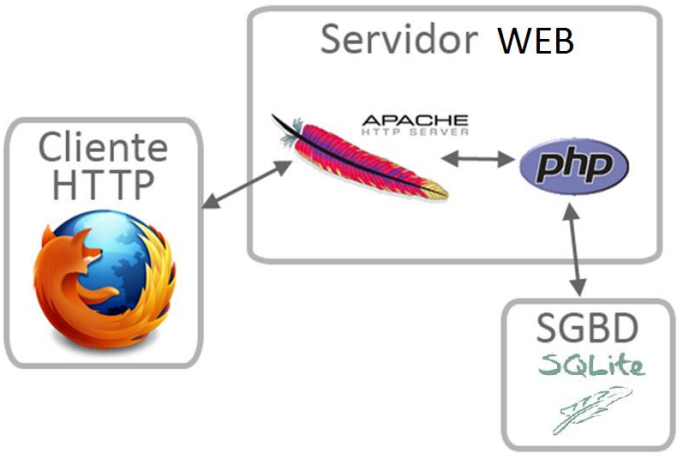


Fig. 2.4 Arquitectura Web Cliente-Servidor de Apache.

La configuración PHP se encuentra en el fichero php.ini y se puede visualizar su contenido con phpinfo().

Para los detalles electrónicos de los sensores remitirse al **Anexo A.3**, para los aspectos de hardware de micro-controladores al **Anexo A.1** y para detalles del Radio Enlace y del Sistema de Energía al **Anexo A.2**.

2.2 Alternativas de Solución al Problema

Al investigar en el tema, se encontró diversas alternativas tecnológicas que cumplían con los lineamientos generales para definir las especificaciones técnicas de diseño, cuyo

resumen se detalla en la **Tabla N° 2.1**:

Tabla N° 2.1 Análisis de las Alternativas de Solución.

Nivel de Abstracción	Alternativas de Solución	Análisis de la Selección
Hardware de adquisición de sensores	Tarjeta de adquisición propia Tarjeta de adquisición propietaria	Se ha decidido diseñar y construir una tarjeta de adquisición propia para integrar diversos sensores.
Control del Arduino	Programación en Asembler Programación en Wiring C++	Se ha decidido programar en C++ por ser más rápido en la detección de errores.
Control del RaspberryPi	Programación Python Programación C	Se ha decidido programar en Python por contar con librerías de comunicación serial-usb.
Persistencia de Datos	Base de Datos SQLite Base de Datos MySQL	Se ha decidido utilizar SQLite por ser más ligero que el MySQL y de uso libre.
Panel de Visualización Web-Scada	PHP – Python PHP – JAVA LABVIEW-Python LABVIEW-JAVA	Se descartó Labview por ser software de pago, se priorizó PHP – Python para integrar la visualización web-scada por contar con librerías OPC.
Sistema de Energizado	Panel Solar – Batería Panel Solar	Se ha decidido utilizar un sistema dual de Panel Solar y Batería para garantizar una operación de 24 horas.
Sistema de Radio Enlace	Radio enlace 15 Km - Bluetooth Celular GPRS - Bluetooth	Se ha decidido utilizar un radio enlace 15 Km con Bluetooth para comunicaciones cortas.

En la **Tabla N° 2.2** se detallan las alternativas de solución a nivel de Sistemas de Adquisición de Datos, Control y Supervisión SCADA (ver referencias [41] y [42]), cuyas ventajas y desventajas han sido tomados en cuenta al momento de formular las especificaciones técnicas para el diseño del prototipo de la Estación Meteorológica.

Desde el punto de vista de hardware, un sistema SCADA consiste de unidades terminales remotas (RTUs) recolectando datos de campo y enviando dicha información a una estación principal, vía un sistema de comunicación. La estación principal visualiza los datos recolectados y le permite al operador realizar tareas de control remoto. Desde el punto de vista de software, un sistema SCADA puede ser divido en dos tipos: propietarios y abiertos. Los fabricantes desarrollan su software propietario para comunicarse a su hardware, dicha solución es vendida como llave en mano.

El inconveniente de este enfoque es la fuerte dependencia del fabricante propietario del software hacia el usuario final. Por este motivo los sistemas de software abierto están cada día tomando más protagonismo, ya que permiten la interoperabilidad, es decir la habilidad de combinar diferentes fabricantes de equipos en un mismo sistema. De acuerdo con los diferentes requisitos dentro de las aplicaciones industriales, se han desarrollado tres principales especificaciones OPC: Acceso a Datos (DA), Alarma y Eventos (A&E) y Acceso a Datos Históricos (HDA). El acceso a los datos de proceso actual se describe en la especificación del DA, mientras que A&E describe una interfaz de información basado en eventos, que incluye el reconocimiento de las alarmas de proceso, y HDA describe las funciones para acceder a los datos archivados. Todas las interfaces ofrecen una manera de navegar a través del espacio de direcciones y provee información acerca de los datos disponibles. El estándar OPC utiliza un enfoque de cliente-servidor para el intercambio de información. Un servidor OPC encapsula la fuente del proceso de información tal como un dispositivo y hace que la información esté disponible a través de su interfaz.

Tabla N° 2.2 Alternativas de Solución a nivel de Integración SCADA.

Tipo de SCADA	Ventajas	Desventajas
SCADA Tipo Sensores a Panel	Es simple, sin CPU, RAM, ROM o programación de software. Los sensores están conectados a los medidores, conmutadores y luces en el panel. Es simple y barato adicionar un nuevo dispositivo como un conmutador o un indicador.	La cantidad de cableado se torna inmanejable para decenas de sensores, la cantidad y el tipo de datos son mínimos y rudimentarios. La reconfiguración del sistema se torna complicada. No existe monitoreo remoto.

SCADA Tipo PLC/DCS	La computadora puede almacenar gran cantidad de datos que pueden ser mostrados de acuerdo al requerimiento del usuario. Cientos de sensores pueden ser conectados al sistema. El operador puede incorporar simulaciones de datos reales, diversos tipos de datos pueden ser recolectados de las unidades terminales remotas o RTU.	Sistema más complejo que el tipo Sensores a Panel, se requieren diferentes habilidades de analista y programador. Todavía se tiene que lidiar con el cableado para cientos de sensores. El operador solo puede visualizar lo que realiza el PLC.
SCADA Tipo PC a IEDs	Se requiere un mínimo de cableado, el operador puede visualizar a nivel de sensor. Los datos recibidos desde el dispositivo incluyen: información de número de serie, modelo, etc. Todos los dispositivos son auto configurables, permitiendo rapidez en la instalación y el reemplazo. Reducción del espacio físico para los sistemas de adquisición de datos mediante los IEDs.	El uso de sistemas más sofisticados requiere emplear trabajadores capacitados. Los precios de los sensores se encarecen por la carencia del PLC en el diseño. Los IEDs están condicionados en los sistemas de comunicación a utilizar.

Para el diseño y construcción del prototipo, se ha utilizado un esquema de SCADA tipo PC a IEDs, bajo un enfoque de código abierto utilizando OpenOPC sobre Python, que es una arquitectura OPC-SCADA diseñado para ser utilizado con el lenguaje de programación Python. En nuestro caso, el dispositivo electrónico inteligente ha sido desarrollado utilizando la plataforma de RaspberryPi sobre Linux, que permitió utilizar esta librería para enviar los datos recolectados hacia el servidor remoto OPC, utilizando la librería python OPC-Client, de esta forma, la integración de la solución propuesta, es compatible con cualquier sistema SCADA que cumpla con el estándar OPC-DA. Para las pruebas de integración se ha utilizado el Servidor OPC del fabricante Matrikon, quien posee una versión de libre distribución del servidor OPC para fines de investigación académica.

En la **Tabla N° 2.3** se muestra la matriz de decisión de enfoque para el prototipo de la Estación Meteorológica, en el cual se ha buscado un escenario óptimo, tanto en los aspectos de hardware como de software.

Tabla N° 2.3 Matriz de Decisión de Especificaciones Técnicas.

Funcionalidad	Ventajas	Desventajas	Decisión
Arquitectura SCADA Abierta sobre OPC.	Posibilidad de acceso a código abierto.	Requiere personal especializado para manejo OPC.	Si, se escogió el enfoque OpenOPC para Linux.
Arquitectura Web Abierta sobre PHP y Apache en Linux.	Posibilidad de integrar web services para distribuir acceso.	Requiere uso de librerías para gráficos avanzada.	Si, se escogió usar la librería PHP Graphic Lib y jpGraph.
Arquitectura Embebida Desacoplada	Fácil depuración y mantenimiento del sistema integral, permite la comunicación Ethernet, serial y bluetooh.	Requiere mayores conocimientos para programar Python, PHP y web en Linux y C embebido para micro controlador.	Sí, se escogió Arduino para la digitalización de los sensores y Raspberry para el servidor web scada.
Arquitectura de Radio Enlace Punto a Punto	Permite asignar un ancho de banda dedicado para transferencia de datos.	Requiere línea de vista suficiente para el radio enlace.	Sí, se escogió el radio enlace punto a punto del fabricante Ubiquiti con alcance de hasta 15 Km.
Hardware de Adquisición de Sensores Básicos	Medición de Temperatura, Presión y Humedad	Una estación real requiere de múltiples parámetros.	Sí, se probó la integración básica de estos sensores vía tarjeta de adquisición.
Sensores Avanzados	Medición de Velocidad y Dirección de Viento, Monóxido, Dióxido e Intensidad de Luz	Requiere mayor diseño en hardware y software.	Sí, se integró a nivel de prototipo.
Utilitarios de comunicación Bluetooh, cámara	Posibilidad de realizar algoritmo de pronóstico utilizando imágenes.	Requiere desplegar a producción el prototipo diseñado.	Parcialmente, se logró la integración, a nivel de hardware.

Con la finalidad de dar aportes a nivel de implementación, en la **Tabla N° 2.4** se listan el detalle de las herramientas de desarrollo de software y hardware que se han utilizado en la Tesis.

Tabla N° 2.4 Herramientas de Desarrollo utilizadas en la Tesis.

Alcance	Herramientas	Descripción
Simulación	Matlab - R	Es un entorno de simulación que soporta las operaciones matriciales de regresión lineal simple y múltiple.
Geany para Aplicación Web	PHP y SQLite para Raspberry Pi	Es un entorno de desarrollo, donde se programado código para recolectar datos de los sensores y almacenarlos en una base de datos SQLite y publicados en un entorno web.
Simulación del entorno Hardware	Proteus	Es un entorno de simulación que permite verificar el diseño del hardware de adquisición de los Sensores con el micro controlador Arduino.
Aplicación Embebida	C++ para Arduino	Se utilizó C++ para el desarrollo del driver que adquiere los datos de los sensores para enviarlos a la tarjeta central del Raspberry Pi.
Aplicación OPC	Python OpenOPC	Se utilizó OpenOPC para inter operatividad SCADA con Python en Linux.

En el **Anexo B**, se ahonda en detalles acerca de los códigos de programación de la Estación Meteorológica, los cuales han sido tomados en cuenta al momento de realizar la construcción del prototipo que será desarrollada en el **Capítulo III**.

2.3 Formulación de las Especificaciones Técnicas

El diseño y construcción del prototipo de la Estación Meteorológica, deberá contemplar el cumplimiento de todas las especificaciones técnicas indicadas, de tal modo que la solución planteada, sea técnicamente viable para una posterior puesta en producción y pueda servir a instituciones: académicas, públicas o privadas, como es el caso del SENAMHI y las universidades. Para ello en la **Tabla N° 2.5** se listan las especificaciones técnicas óptimas que han sido elaboradas con la finalidad de establecer los requerimientos mínimos necesarios al momento de esbozar la solución.

Tabla N° 2.5 Especificaciones Técnicas de la Tesis.

Requerimiento	Especificación Técnica
Req. 1: Capacidad de comunicación remota.	ET1: Radio enlace punto a punto hasta un máximo de 15 Km con línea de vista directa.
Req. 2: Capacidad de operación las 24 horas.	ET2: Sistema dual: Panel Solar con salida 12 VDC y Batería de 12 VDC, con inversor 220 VAC. Capacidad de operación las 24 horas del día.
Req. 3: Capacidad de integrar sensores.	ET3: Tarjeta de adquisición de sensores Arduino UNO, con capacidad de integrar sensores de Temperatura, Presión, Humedad, Monóxido, Dióxido, intensidad de Luz, velocidad y dirección de viento.
Req. 4: Capacidad de interoperabilidad web-scada.	ET4: Micro controlador RaspberryPi con Linux, con servidor web Apache-PHP, Cliente OpenOPC-Python para SCADA y Base de Datos SQLite para registro de datos meteorológicos.
Req. 5: Capacidad de expansión de accesorios externos.	ET5: El diseño deberá permitir la posibilidad de habilitar una cámara fotográfica-video y comunicación inalámbrica vía bluetooth.
Req. 6: Capacidad de acceso al código fuente.	ET6: La solución deberá utilizar lenguajes de programación de código abierto: Lenguaje C++ para Micro controlador Arduino; Lenguaje Python para micro controlador RaspberryPi y OpenOPC; Lenguaje PHP para servidor web; Lenguaje SQL para Base de datos SQLite.
Req. 7: Capacidad de distribución física para uso de chasis.	ET7: El diseño deberá contemplar un esquema de sub sistemas para que la solución integral pueda desplegarse fácilmente sobre un chasis estándar para una estación meteorológica.

Es importante recalcar, que las especificaciones técnicas del proyecto, han sido esbozados para un contexto real para la ciudad de Huánuco, en un escenario de monitoreo para la laguna de Mancapozo, del Centro Poblado de Malconga. Sin embargo, estos requerimientos no pierden su generalidad ya que, en otros escenarios como la costa o selva, solo se necesitarán algunas adecuaciones para su uso en cuanto a los rangos de temperatura de operación, niveles de voltaje en la fuente de alimentación, entre otros.

CAPÍTULO III
DISEÑO Y CONSTRUCCIÓN DEL PROTOTIPO

En este capítulo, se mostrará el diseño y construcción de la Estación Meteorológica usando los requerimientos y las especificaciones técnicas esbozadas en el **Capítulo II**. Asimismo, se describirán detalles de la implementación del software del sistema, en sus componentes web y SCADA, así como del diseño y construcción del hardware usando el RaspberryPi-Arduino. Para comprobar el diseño electrónico se ha empleado la plataforma Proteus a efectos de integrar la data de la base de datos SQLite hacia el contenedor web de Apache y scada OpenOPC en Linux.

3.1 Elaboración del Diseño de la Estación Meteorológica

En la **Tabla N° 3.1** se detalla el Diseño de la Estación Meteorológica en base a las Especificaciones Técnicas. Esto incluye los siguientes sub sistemas de diseño electrónico a nivel de hardware:

- Sub Sistema de Energía.
- Sub Sistema de Transmisión de Datos.
- Sub Sistema de Control de Datos y Data Logger.
- Sub Sistema de Adquisición de Sensores.
- Sub Sistema de Medición de Viento y Lluvia.
- Sub Sistema de Medición de Luz Ultravioleta.

Tabla N° 3.1 Diseño de las Especificaciones Técnicas de la Tesis.

Especificación Técnica	Detalle de Diseño
ET1: Radio enlace punto a punto hasta un máximo de 15 Km con línea de vista directa.	Sub Sistema de Transmisión de Datos -Radio Enlace hasta 15 Km
ET2: Sistema dual: Panel Solar con salida 12 VDC y Batería de 12 VDC, con inversor 220 VAC. Capacidad de operación las 24 horas del día.	Sub Sistema de Energía -Panel Solar -Batería

ET3: Tarjeta de adquisición de sensores Arduino UNO, con capacidad de integrar sensores de Temperatura, Presión, Humedad, Monóxido, Dióxido, intensidad de Luz, velocidad y dirección de viento.	Sub Sistema de Adquisición de Sensores -T, P, H -CO2, CO Sub Sistema de Medición de Viento y Lluvia Sub Sistema de Medición de Luz Ultravioleta
ET4: Micro controlador RaspberryPi con Linux, con servidor web Apache-PHP, Cliente OpenOPC-Python para SCADA y Base de Datos SQLite para registro de datos meteorológicos.	Sub Sistema de Control de Datos y Data-Logger -RaspberryPi -Arduino UNO
ET5: El diseño deberá permitir la posibilidad de habilitar una cámara fotográfica-video y comunicación inalámbrica vía bluetooth.	Módulos complementarios -Bluetooth -Cámara
ET6: La solución deberá utilizar lenguajes de programación de código abierto: Lenguaje C++ para Micro controlador Arduino; Lenguaje Python para micro controlador RaspberryPi y OpenOPC; Lenguaje PHP para servidor web; Lenguaje SQL para Base de datos SQLite.	Entorno de Desarrollo -Python -PHP -C++ -SQL
ET7: El diseño deberá contemplar un esquema de sub sistemas para que la solución integral pueda desplegarse fácilmente sobre un chasis estándar para una estación meteorológica.	Enfoque modular vía sub sistemas - Sub Sistema de Transmisión de Datos - Sub Sistema de Energía - Sub Sistema Adquisición de Sensores - Sub Sistema de Medición de Viento y Lluvia - Sub Sistema de Medición de Luz Ultravioleta - Sub Sistema de Control de Datos y Data-Logger

En la **Fig. 3.1** se detalla cada uno de los sub sistemas del diseño electrónico de la estación meteorológica, con el detalle de la comunicación interna y externa de cada uno de los componentes. La Tarjeta de Adquisición de Sensores, contiene la etapa de

acondicionamiento de señales, internamente incorpora los sensores de Humedad, Presión, Temperatura y posee conectores para decodificar los datos provenientes de los sensores de dióxido, monóxido, Ultravioleta, Viento y lluvia.

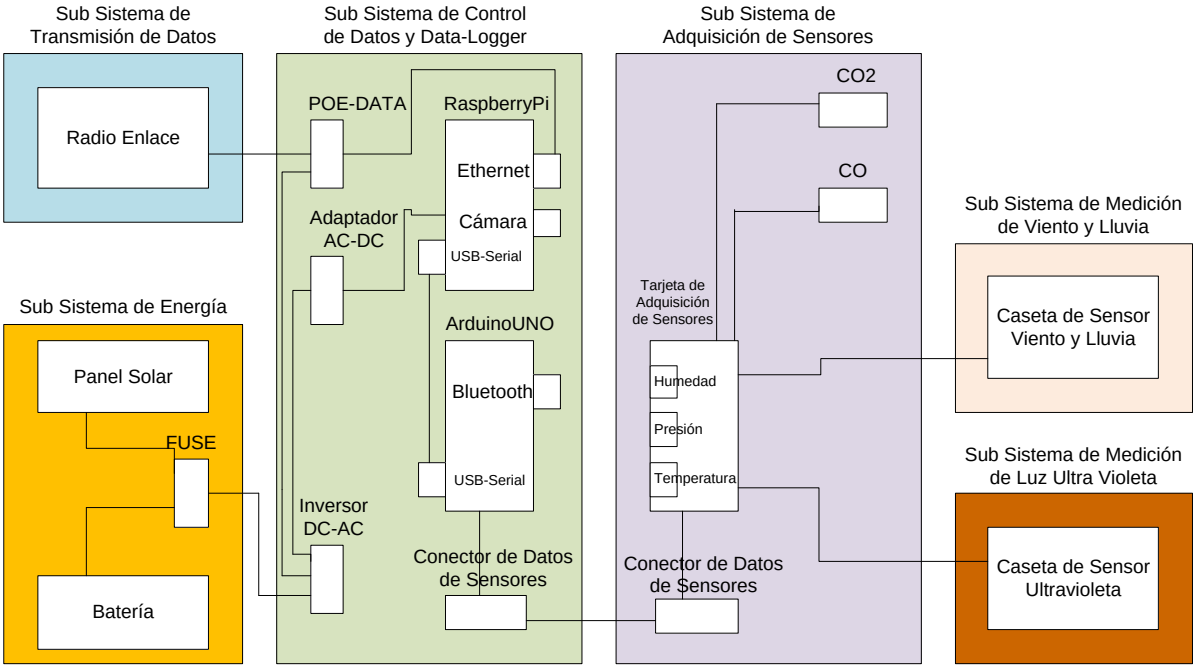


Fig. 3.1 Sub Sistemas del Diseño Electrónico de la Estación Meteorológica.

A continuación, se detallan los aspectos técnicos de los sensores a utilizar en el proyecto, así como el acondicionamiento de señal y los filtros a utilizar:

Sensor de Presión: El Sensor de Presión absoluta MPX4250A, tiene un rango de operación desde 2.9 PSI hasta los 36.3 PSI, con un rango de operación de temperatura desde -40°C hasta 125°C, con una precisión de +/- 1.5%, y una salida entre 0.2 V hasta 4.9 V.

Sensor de Temperatura: El Sensor de Temperatura DS18B20, funciona con un pin digital y posee un conversor incorporado de 9 a 12 bits de resolución para la adquisición de la temperatura. Se comunica con un pin digital, tierra, fuente y trabaja en un rango de temperaturas desde los -55°C hasta los 125°C y tiene una precisión de +/- 0.5°C sobre un rango de -10°C hasta los 85°C, utiliza el protocolo de comunicación 1-Wire Digital. El dispositivo sensor utiliza el comando T (0x44) con la librería OneWire, mediante la función ds.write(0x44), cuando el dispositivo sensor recibe esta instrucción, se activa ADC interno y digitaliza la temperatura y guarda la información en un registro. El tiempo de conversión varía de acuerdo a la resolución, entre 94 ms (9 bits de resolución) hasta los 750 ms (12 bits de resolución). Luego que está listo la conversión, el dispositivo es leído utilizando la función ds.read().

Sensor de Humedad: El Sensor de Humedad HIH-4000, mide la humedad relativa (%HR) y envía una señal analógica que es capturado por el puerto ADC del micro controlador, y gracias a una salida casi lineal de voltaje, es fácil procesar la información. Cuenta con una alimentación de 5 VDC y consume 200 uA.

Sensor de Ultravioleta: El Sensor de Ultra Violeta ML8511, tiene como salida un pin de voltaje digital que mide la intensidad de UV en mW/cm^2, internamente utiliza un amplificador operacional el cual convierte mediante un transductor de corriente a voltaje dependiendo de la intensidad de radiación ultravioleta. Se alimenta con una corriente de 0.1 uA, posee un rango de trabajo desde -20 °C hasta los 70 °C.

Sensor de Monóxido de Carbono (CO): Este sensor detecta las concentraciones de CO en el aire y genera una salida en voltaje analógico. El sensor puede medir concentraciones de 10 a 10,000 ppm. Puede operar a temperaturas desde -10 °C hasta los 50°C y consume menos de 150 mA, 5V.

Sensor de Dióxido de Carbono (CO2): El sensor CO2 utiliza una tarjeta MG-811 que posee un circuito de acondicionamiento de señal mediante un amplificador operacional y un circuito de calor para calentar el sensor, posee salida analógica y digital.

Sensor de Viento y Lluvia: El sensor BH4TDV permite capturar la velocidad y posición de viento y la lluvia vía dos conectores RJ11. Opera con voltajes de entre 3.3 V hasta los 16 V, incorpora circuitería para el acondicionamiento de la señal. En la **Fig. 3.2**, se visualiza el detalle del sensor.

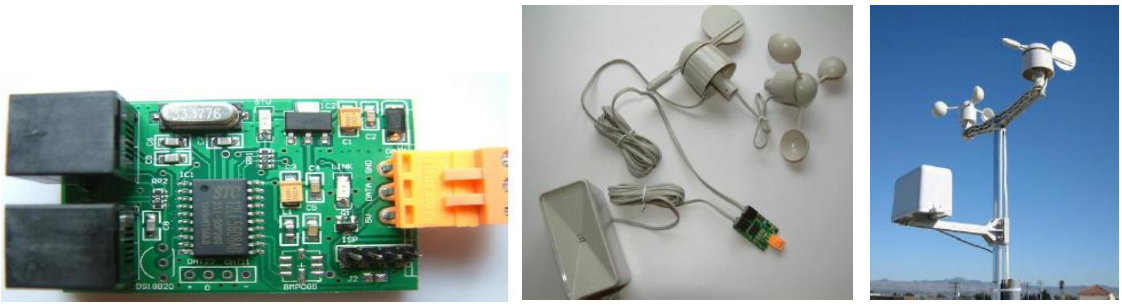


Fig. 3.2 Sensor de Velocidad y Dirección del Viento e Intensidad de Lluvia.

Sensor de Cámara: El sensor RaspberryPi Camera, permite obtener fotos y video con una resolución de 5 Mega pixeles en formato nativo, con soporte para video de 1080p30, 720p60, y 640x480p60/90, dichas imágenes podrían captar los momentos de lluvia para que, por ejemplo, posteriormente pueda utilizarse procesamiento digital de imágenes para calcular la intensidad de lluvia mediante el diámetro de las gotas de lluvia.

Sensor Bluetooth: El sensor Chip Bluetooth trabaja con un voltaje de 3.3 V, con una velocidad de transmisión de 1200, 2400, 4800, 9600, 19200, 38400, 57600 y 115200

baudios, por defecto viene configurado a 9600 baudios. Posee una corriente de trabajo de 8 mA, responde a comandos AT. Código de Pin 1234, Baud Rate: 9600, N, 8, 1. Permite enviar comandos a la Estación Meteorológica para tareas de calibración, operación y mantenimiento.

En la **Fig. 3.3** se muestra las capas de abstracción del sistema de la estación meteorológica, que incluye el análisis computacional para asistencia al pronóstico de clima en base a la información meteorológica usando el análisis de regresión lineal simple y múltiple, basado en el prototipo de la Estación Meteorológica que tiene un componente de hardware y un componente de software. Para el diseño del prototipo en hardware se ha usado la plataforma Proteus que permite verificar el diseño simulado de la Estación Meteorológica que almacena señales reales tomadas de los sensores de Temperatura, Presión y Humedad. A continuación, se muestran las capas de abstracción del proyecto:

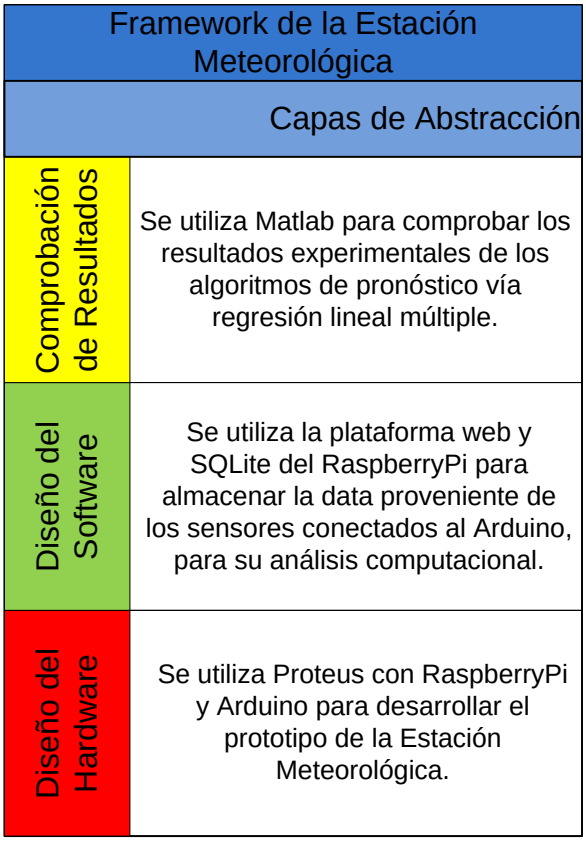


Fig. 3.3 Capas de Abstracción de la Estación Meteorológica.

Flujo grama de los Componentes de Software de la Estación Meteorológica: El proyecto incluye el desarrollo de tres módulos que contiene seis sub procesos de software, cuyo detalle se muestra en la **Tabla N° 3.2**. El módulo de adquisición de señales se ha implementado en el Arduino y vía comunicación USB se ha comunicado con el RaspberryPi mediante código Python, que transfiere los datos digitalizados en la

base de datos SQLite.

Tabla N° 3.2 Módulos y sub procesos del Sistema Meteorológico.

Módulo	**Sub Procesos**	**Descripción**
Módulo de Adquisición de Señales Meteorológicas	Sub Proceso de Recolección de Datos de Sensores en el Servidor Local RaspberryPI	Sistematización del recojo de la información en forma programada diariamente
	Sub Proceso de Captura y Guardado de Datos de Sensores en la Base de Datos	Recoge la información digital de los sensores y los guarda en la Base de Datos del Servidor Local Raspberry
Módulo de Predicción Meteorológica	Sub Proceso de Predicción Meteorológica	Implementación de los algoritmos de predicción
	Sub Proceso de Transferencia de Datos de Sensores desde el Servidor Local RaspberryPI hacia el Servidor Remoto	Transfiere y centraliza los datos desde el Servidor Local hacia el Servidor Remoto
Módulo de Reportes Web-SCADA	Sub Proceso de Visualización Web-SCADA en el Servidor Remoto de la Estación Meteorológica	Crea reportes web scada de las variables de los sensores de la Estación
	Sub Proceso de Visualización Web en el Servidor Local de la Estación Meteorológica	Crea reportes web recientes de las variables de los sensores de la Estación

En la **Fig. 3.4** se ha detallado la dependencia entre los módulos del sistema meteorológico, teniendo en cuenta que cada sub proceso está albergado en un componente de hardware y software específico. En la **Fig. 3.5** se ha detallado los módulos internos de la interface web. En el **Anexo B**, se incluye el código fuente de la programación en PHP, que permite implementar la funcionalidad que se ha diseñado para la estación prototipo.

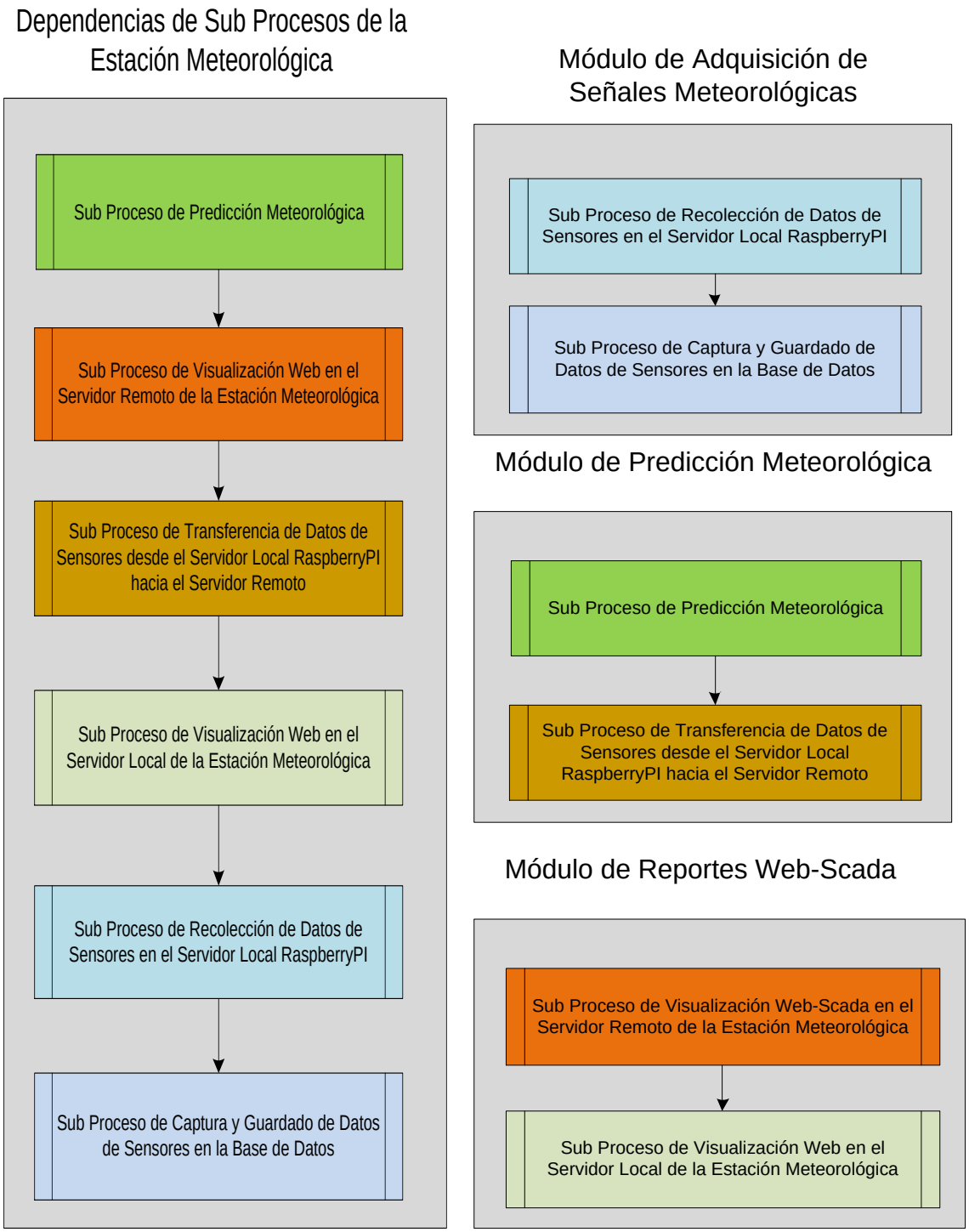


Fig. 3.4 Dependencia de los Módulos del Sistema Meteorológico.

Para el prototipo en software se ha dividido el sistema de la Estación Meteorológica en sub sistemas integrando varias tecnologías de código abierto.

Módulo de Adquisición de Señales Meteorológicas:

- Para recolectar las señales de los sensores de la Estación Meteorológica se utiliza la

tarjeta de adquisición de señales Arduino, que permite digitalizar las señales de los sensores: Temperatura (DS18B20), Humedad (HIH-4000), Presión (MPX4250A), Ultravioleta (ML8511), Monóxido (MQ7), CO2 (MG-811), Velocidad-Dirección de Viento y Lluvia (BH4TDV). Dicha información viaja a través de un radio enlace punto a punto Ubiquiti que se alimentará desde un Panel Solar ubicado en la zona de intervención.

- Para comprobar la consistencia del sistema, los datos adquiridos son almacenados en una base de datos SQLite sobre la plataforma Raspberry-Pi que se comunica con la tarjeta Arduino, con la finalidad de procesar los datos en forma remota a través de servicios web.
- Asimismo, el componente web tiene la posibilidad de comunicarse a un servidor SCADA vía conexión OPC, para el prototipo se utilizó el OpenOPC para comunicarse con el OPC Server del fabricante MatrikonOPC.
- A continuación se desarrolla a detalle los entornos de desarrollo utilizados para el proyecto (ver **Tabla N° 3.3**):

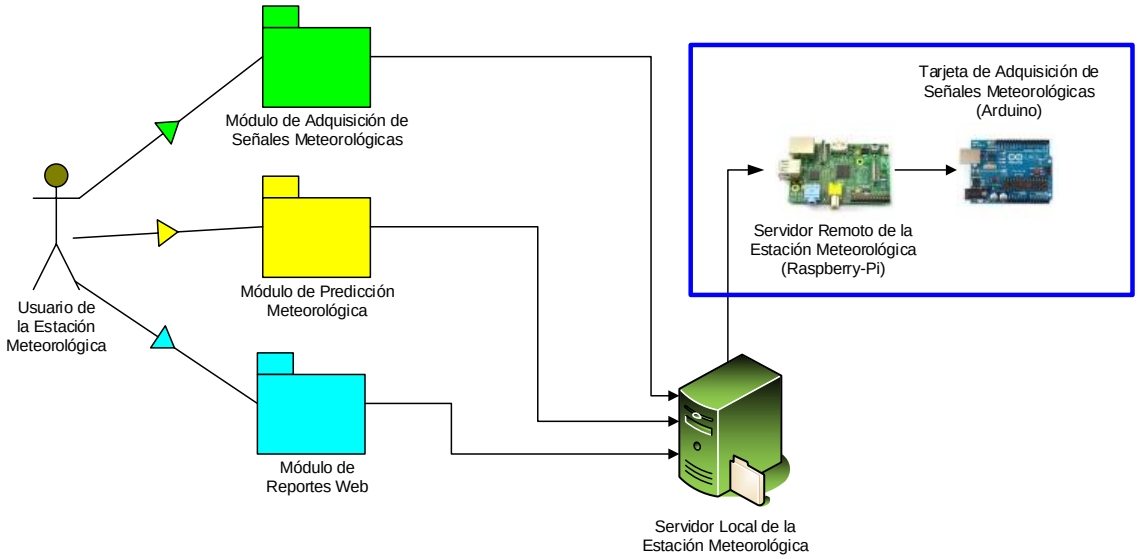


Fig. 3.5 Módulos internos del Sistema Web de la Estación Meteorológica.

Tabla N° 3.3 Descripción del Entorno de Desarrollo.

N°	Entorno de Desarrollo	Descripción / Lenguaje de Programación
1	SQLiteBrowser, para Modelado de Base de Datos SQLite.	SQL-RaspberryPI para almacenar los datos en forma persistente.

N°	Entorno de Desarrollo	Descripción / Lenguaje de Programación
2	PHPDesigner para programación en PHP.	PHP-RaspberryPI para publicar los datos vía web desde la base de datos SQLite.
3	Spyder para programación en Python.	Python-RaspberryPI, para transferir datos procesados del Arduino hacia la base de datos SQLite.
4	VI Script para programación del Cron en Linux.	CronLinux-RaspberryPI para transferir diariamente los datos almacenados desde el servidor local hacia el servidor remoto vía Radio Enlace, así como borrar datos para liberar espacio en disco.
5	RStudio para programación en lenguaje R con Excel.	Excel-R para procesar los datos utilizando algoritmos de predicción de datos. El lenguaje R incorpora librerías para análisis de regresión lineal simple y múltiple, con funcionalidad de publicación web.
6	Arduino IDE 1.6 para programación en C++ del Arduino.	Arduino-C++ para programar ArduinoUno que permite adquirir las señales desde los sensores hacia el micro controlador.
7	Utilitarios gratuitos de configuración remota.	Putty-WinSCP para configurar remotamente el Servidor RaspberryPI.

En la **Fig. 3.6** se ha descrito los pasos en la adquisición de los datos y los mecanismos de su procesamiento para la interface Web:

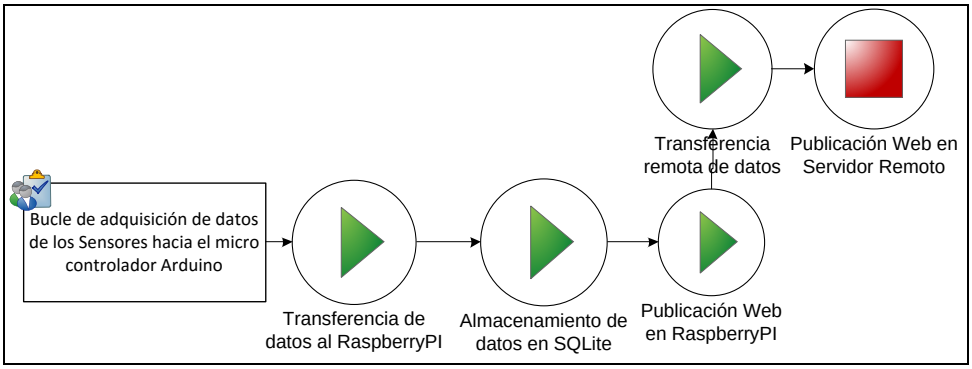


Fig. 3.6 Flujo de información desde los Sensores hacia la Web.

En la **Fig. 3.7**, se ha precisado el detalle del Módulo de Adquisición de Señales, que utiliza la tarjeta Arduino para digitalizar los datos provenientes de los sensores, los cuales son transferidos al sistema operativo Linux almacenado en la tarjeta RaspberryPi, vía código de Python por el puerto serial USB.

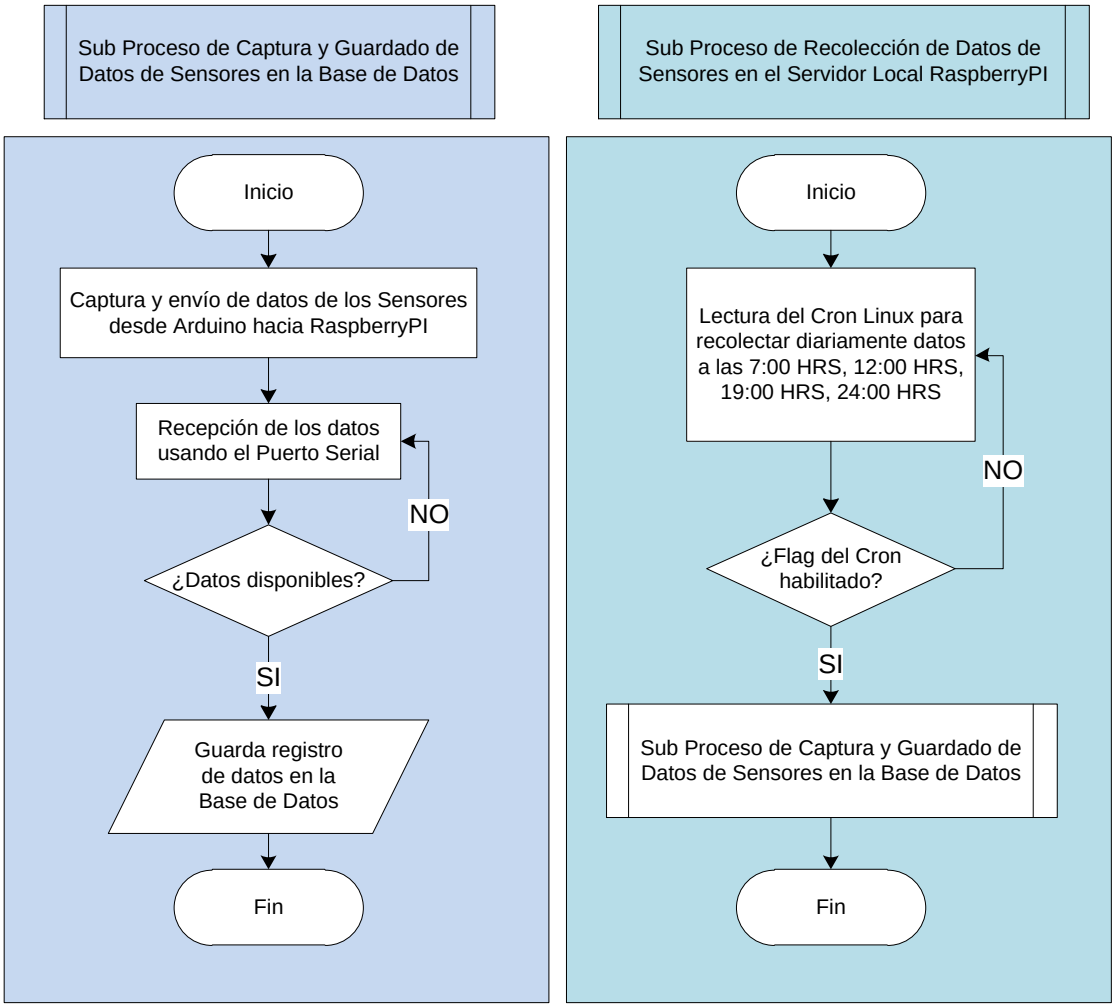


Fig. 3.7 Sub Procesos que integran el Módulo de Adquisición de Señales.

Módulo de Predicción Meteorológica:

- La data obtenida de la Estación Meteorológica ha sido procesada en Matlab-Excel-R extraída vía web de la Base de Datos, para aplicar los algoritmos de Regresión Lineal Simple y Múltiple que permiten pronosticar el clima.
- En la **Fig. 3.8**, se muestra un diagrama de flujo de datos (DFD) de los algoritmos que se han empleado para la predicción meteorológica:

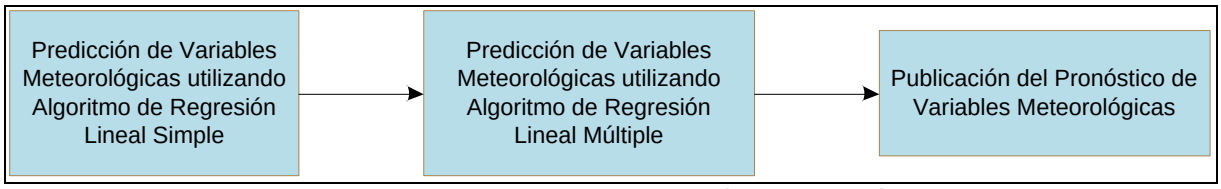


Fig. 3.8 Flujo de Datos para Predicción Meteorológica.

- Los algoritmos de predicción de temperatura han sido implementados en el proyecto de investigación, dando aportes en el tema en un contexto local, regional y nacional.
- En la **Fig. 3.9** se muestra a detalle el diagrama de bloques del módulo de predicción que utiliza los datos almacenados en la base de datos SQLite. Cabe precisar que la configuración del cron en Linux, obtiene 4 datos programados por día, a las 03 hrs, 07 hrs, 15 hrs y 19 hrs. Adicionalmente existe la posibilidad de recolectar datos en forma personalizada ingresando directamente una llamada a la función en Python. Para detalles referirse al **Anexo B**, sobre el código de programación.

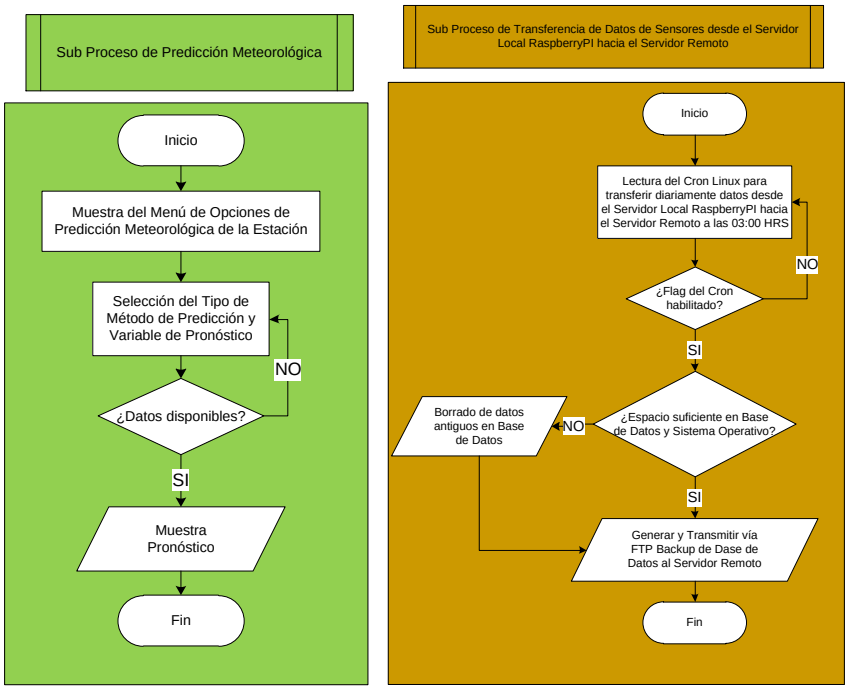


Fig. 3.9 Sub Procesos que integran el Módulo de Predicción.

Módulo de Reportes Web-Scada:

- La data obtenida de la base de datos SQLite del Raspberry-Pi ha sido almacenada en formato de texto plano y leído por el servidor web Apache con PHP, para generar reportes por perfil de periodos de tiempo y tipo de sensores. También la data es enviada a un Servidor OPC mediante el cliente OpenOPC con Python en Linux.
- En la **Fig. 3.10**, se muestran los sub procesos que integran el módulo de reportes web, en donde se han utilizado librerías de visualización gráfica en PHP, en particular

las librerías Jpgraph para la gráfica de la dirección y velocidad de viento, conjuntamente con la librería PHPGraphLib para las demás gráficas de los sensores, tales como: Temperatura, Presión, Humedad, Intensidad de Radiación Ultravioleta, Intensidad de Monóxido de Carbono, CO2, entre otros. También se utilizó la librería OpenOPC para leer y escribir datos al servidor OPC.

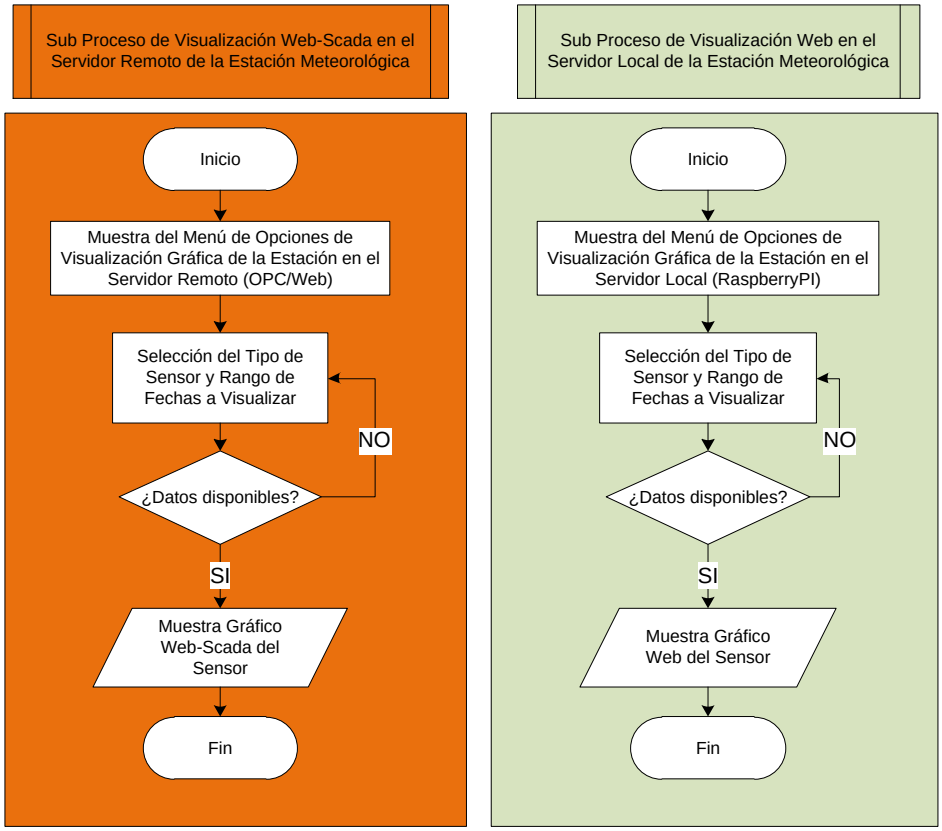


Fig. 3.10 Sub Procesos que integran el Módulo de Reportes Web-Scada.

En la **Fig. 3.11**, se muestra los detalles de diseño del chasis para la Estación Meteorológica, en donde se aprecia que en la parte superior se encuentra ubicado el plato de radio enlace; en la parte inferior por encima de 1 metro, se ubica el panel solar con la batería. También se distribuyó la caseta de sensores, una caseta para el sensor de viento y otra para la intensidad de luz. En la parte central se ubica la caseta de control y data logger. En resumen, en esta sección se ha realizado el diseño en base a las especificaciones técnicas, que han sido esbozadas en base a los requerimientos del usuario que ha sido elaborado en el **Capítulo II**. En un escenario real, se requiere además incluir aspectos de seguridad, tal como un para rayo, así como una cerca perimétrica. Se sobre entiende que estos aspectos están fuera del alcance de la tesis, por cuanto no está relacionado a los aspectos de ingeniería electrónica, sin embargo, se menciona para tenerlo en cuenta al momento de pasar a una etapa de producción y

operación.

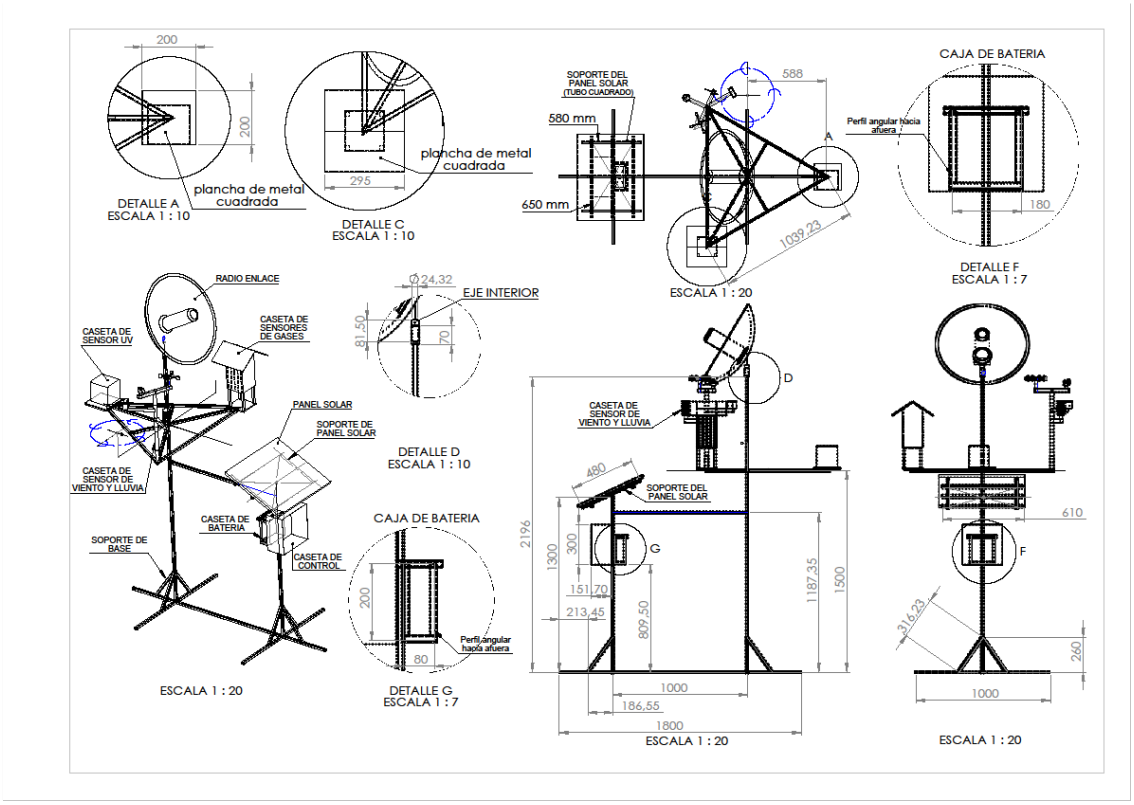


Fig. 3.11 Diseño del Chasis de la Estación Meteorológica.

3.2 Construcción de la Estación Meteorológica

El micro controlador **Raspberry-Pi** consta de un procesador de 256 RAM, con un slot GPIO de propósito general de entrada salida, salida de video y audio, leds, fuente de energía micro USB de 5 voltios, una interface HDMI de video digital de salida, puerto Ethernet, un slot para memoria SD y uno para USB. En la **Fig. 3.12** se detalla las interfaces de entrada salida en la plataforma de RaspberryPi.

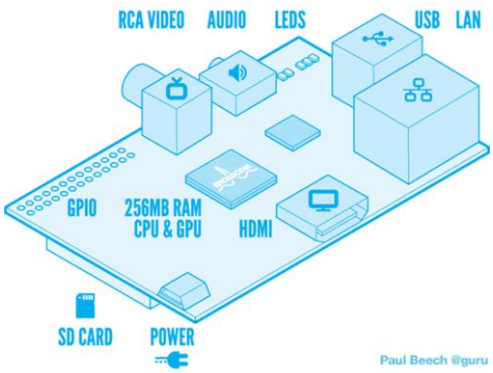


Fig. 3.12 Plataforma Open Hardware de Raspberry-Pi.

Para enviar los datos de los sensores digitalizados hacia el servidor OPC, se ha desarrollado el siguiente código en Python:

Rutina de Escritura SCADA en Python

```
def log_scada(self):
        # almacena los datos de los sensores en OPC SCADA Server
        host='192.168.1.45'
        OPCName='Matrikon.OPC.Simulation.1'
        opc = OpenOPC.open_client(host)
        opc.connect(OPCName, host)
        consulta = [('.Temperatura',self.temperaturaEstacion) ,
('.Presion',self.presionEstacion) , ('.Humedad',self.humedadEstacion),
('.Monoxido',self.monoxido) , ('.Dioxido',self.dioxido) , ('.IntensidadLuz',self.intensidadUV),
('.IntensidadLluvia',self.lluvia) , ('.VelocidadViento',self.velocidadViento) ,
('.DireccionViento',self.direccionViento)]
        print( consulta )
        opc.write(consulta)
        opc.close()
        print("escritura SCADA fue exitosa")
```

Rutina de Escritura en Base de Datos SQLite

```
def log_estacion(self):
# almacena los datos de los sensores en la base de datos
  conn=sqlite3.connect(dbname)
  curs=conn.cursor()
  consulta = "insert into sensor_data (timestamp, temperatura, presion, humedad,
monoxido, dioxido, intensidadUV, velocidadViento, direccionViento, lluvia,
temperaturaEstacion, presionEstacion, humedadEstacion) values ('"+
str(self.timestamp)+"','"+str(self.temperatura)+"','"+str(self.presion)+"','"+str(self.humedad)
+"','"+str(self.monoxido)+"','"+str(self.dioxido)+"','"+str(self.intensidadUV)+"','"+str(self.velo
cidadViento)+"','"+str(self.direccionViento)+"','"+str(self.lluvia)+"','"+str(self.temperaturaEst
acion)+"','"+str(self.presionEstacion)+"','"+str(self.humedadEstacion)+"')"
  print( consulta )
  curs.execute(consulta)
# guardar los cambios
  conn.commit()
```

```
  conn.close()
  print("insert SQLite fue exitoso")
```

Rutina de Programación del Cron en Linux de RaspberryPi

```
*/5 * * * * python /home/pi/OpenOPC-1.2.0/src/pruebaSensores_meteoroLinux.py
```

Rutina de Publicación Dashboard Web en PHP

```
<?php
include("charts4php/lib/inc/chartphp_dist.php");
$dbDb = "datos_Meteoro.db";      // Nombre de la base de datos
$dbTabla= "sensor_data";          // Nombre de la tabla
$bd = new SQLite3($dbDb);
$query="select * from ".$dbTabla." order by id DESC LIMIT 1;";
$results = $bd->query($query);
$dataArray=array();
while ($row = $results->fetchArray())
{
  $dbTime=$row[1];
  $dbT=$row[2];
  $dbP=$row[3];
  $dbH=$row[4];
  $dbM=$row[5];
  $dbD=$row[6];
  $dbU=$row[7];
  $dbLL=$row[10];
  $dbV=$row[8];
  $dbDV=$row[9];
}
$t = new chartphp();
$t->data = array(array($dbT));
$t->intervals = array(10,20,30,50);
$t->chart_type = "meter";
$t->title = "Temperatura";
$outT = $t->render("c1");

$p = new chartphp();
```

```
$p->data = array(array($dbP));
$p->intervals = array(600,800,900,1500);
$p->chart_type = "meter";
$p->title = "Presion";
$outP = $p->render("c2"); …
```

Rutina de Publicación del Histórico Web en PHP

```
<script type="text/javascript">
$(function ()
{
$('#tiporeporte').change(function()
{
var fecha1;
var fecha2;
var opcion;
fecha1=document.getElementById('date1').value;
fecha2=document.getElementById('date2').value;
opcion = $('input:radio[name="radio"]:checked').val();
datos=document.getElementById('datos').value;
image = document.getElementById('imagen1');
document.getElementById('UNO').style.display='block';
document.getElementById('TODOS').style.display='none';

switch($(this).val()) {
 case '1':
   image = document.getElementById('imagen0');
image.src="grafica.php?grafico=temperatura&fecha1="+fecha1+"&fecha2="+fecha2+"&o
pcion="+opcion+"&datos="+datos;
break;
 case '2':
    image = document.getElementById('imagen0');
image.src="grafica.php?grafico=presion&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion
="+opcion+"&datos="+datos;
 break;
   case '3': …
```

El micro controlador **Arduino UNO** es una plataforma open-hardware basada en una sencilla placa con entradas y salidas (E/S) tanto analógicas como digitales, consta de un micro controlador Atmega8, un chip sencillo y de bajo costo que permite el desarrollo de múltiples diseños. No requiere adquirir licencia alguna, ya que al ser open-hardware tanto su diseño como su distribución es libre para desarrollar cualquier tipo de proyecto.

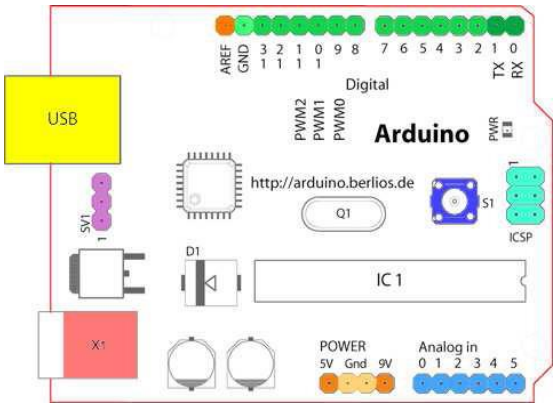


Fig. 3.13 Plataforma Open Hardware de Arduino.

En la **Fig. 3.13** se muestra el detalle de la plataforma open hardware de Arduino con sus respectivos pines de entrada salida. La Estación Meteorológica utiliza la tarjeta Raspberry-Pi para albergar la base de datos, así como el servidor web, el cual se conecta a la tarjeta Arduino que está conectado a los sensores del prototipo vía puerto serial-usb. En la **Tabla N° 3.4** se detallan los sensores utilizados en el prototipo del proyecto:

Tabla N° 3.4 Sensores utilizados en la Estación Meteorológica.

Sensor	Código	Características
Temperatura	DS18B20	Es un sensor de temperatura digital, que utiliza el protocolo de bus MAXIM de 1 cable tanto para recibir como transmitir datos.
Humedad	HIH-4000	Requiere fuente de 5 Vdc, agregando un divisor de voltaje a la salida se obtiene el rango de tensión necesario que la entrada analógica del Arduino necesita. Posee un rango de operación de temperatura entre -40°C y 84°C y una precisión +/- 3.5% HR.
Presión	MPX4250A	Requiere fuente de 5 Vdc, agregando un divisor de voltaje a la salida se obtiene el rango de tensión necesario que la entrada analógica del Arduino necesita. Posee un rango de 20 a 250 kPa (2.9 psi a 36.3 psi). Su

		rango de temperatura de trabajo oscila entre -40 °C hasta 125 °C.
Ultravioleta	ML8511	Tiene como salida un pin de voltaje digital, que mide la intensidad de UV en mW/cm2. Se alimenta con una corriente de 0.1 uA, posee un rango de trabajo desde -20 °C hasta los 70 °C .
Monóxido de Carbono	CO Board	Detecta las concentraciones de CO en el aire y genera una salida en voltaje analógico. El sensor puede medir concentraciones de 10 a 10,000 ppm. Puede operar a temperaturas desde -10 °C hasta los 50°C, y consume menos de 150 mA, 5V.
Dióxido de Carbono	MG-811	Mide el CO2, posee un circuito de acondicionamiento de señal mediante un amplificador y un circuito de calor para calentar el sensor, posee salida analógica y digital.
Viento y Lluvia	BH4TDV	Permite capturar la velocidad y posición de viento, y la lluvia, vía dos conectores RJ11. Opera con voltajes de entre 3.3 V hasta los 16 V, incorpora circuitería para el acondicionamiento de la señal.
Bluetooth	BT Board	El sensor Chip Bluetooth trabaja con un voltaje de 3.3 V, con una velocidad de transmisión desde 1200, hasta 115200 baudios, por defecto viene configurado a 9600 baudios. Posee una corriente de trabajo de 8 mA. Permite enviar comandos a la Estación Meteorológica para tareas de calibración, operación y mantenimiento
Cámara Digital	Cámara Raspberry Pi	Permite adquirir fotos y/o videos de las nubes donde está ubicado la Estación Meteorológica.

En la gráfica de la **Fig. 3.14**, se muestra los detalles de la interface de adecuación electrónica de la estación meteorológica, que incluye una interface de comunicación serial RS232 para el sensor de viento y lluvia, para mayor detalle revisar el **Anexo B**, que contiene el código fuente en lenguaje C para el Arduino, el cual se comunica con el código Python para la transferencia de datos digitalizados al RaspberryPi vía puerto RS232-USB.

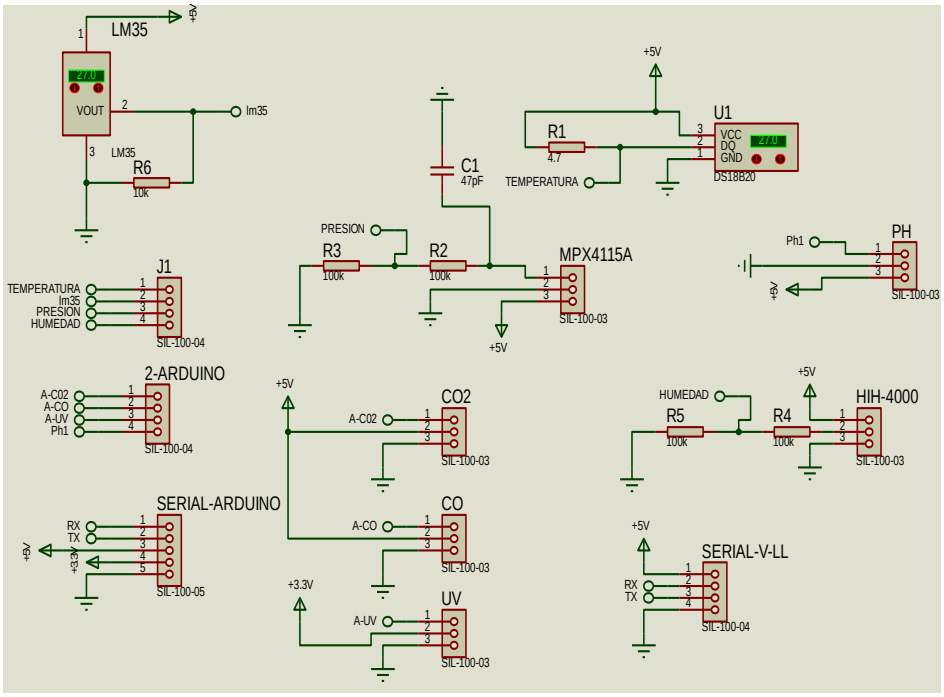


Fig. 3.14 Circuito de adecuación electrónica de la Estación Meteorológica.

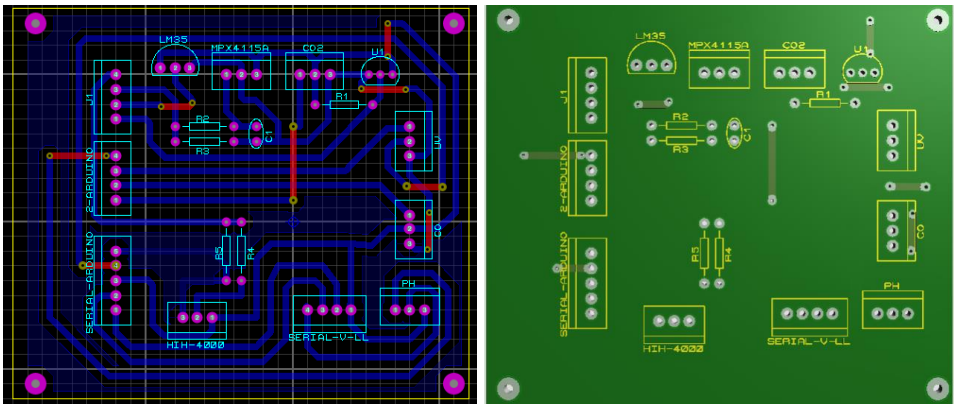


Fig. 3.15 Diseño de la Tarjeta de Adquisición de Sensores.

Código de Digitalización de Sensores del Arduino

```
void loop(void)
{
 if (Serial.available())
 {
  char sensor = Serial.read();
   if ((sensor == 'T') || (sensor == 't')) {
     Serial.print("");
     getBuffer(); //Inicio!
     Serial.print(Temperature());
```

```
} else if (sensor=='P'){
  Serial.print("");
  getBuffer(); //Inicio!
  Serial.print(BarPressure());
} else if (sensor=='H'){
  Serial.print("");
  getBuffer(); //Inicio!
  Serial.print(Humidity());
} else if (sensor=='M'){
  Serial.print("");
  Serial.print(readMONOXIDO(A0));
} else if (sensor=='C'){
     int percentage;
     float volts;
     Serial.print("");
     volts = readCO2(A3);
     percentage=readCO2Percentage(volts);
     if (percentage == -1) {
           //Serial.print( "<400" );
           Serial.print( "400" );
        } else {
           Serial.print(percentage);
        }
} else if (sensor=='U'){
     float uvLevel = readULTRAVIOLETA(A4);
     float outputVoltage = 5.0 * uvLevel/1024;
     float uvIntensity = mapfloat(outputVoltage, 0.99, 2.9, 0.0, 15.0);
} else if (sensor=='V'){
   getBuffer(); //Inicio!
   Serial.print("");
   Serial.print(WindDirection());
   Serial.print(",");
   Serial.print(WindSpeedMax());
   Serial.print(",");
   Serial.print(RainfallOneDay());
   Serial.print(",");
```

```
    Serial.print(Temperature());
    Serial.print(",");
    Serial.print(Humidity());
    Serial.print(",");
    Serial.print(BarPressure());
  } else  {
    Serial.print(sensor);
  }    }
```

Para las pruebas de validación del prototipo a nivel de componentes de software, se realizó el siguiente procedimiento:

1) Se comprobó la comunicación serial del Arduino hacia la PC en entorno Windows y luego en Linux vía puerto serial-usb.
2) Se verificó la integración Python para grabar los datos digitalizados de los sensores, desde el Arduino hacia la base de datos SQLite ubicado en el RaspberryPi.
3) Finalmente se validó la integración del sistema web con el sistema scada, utilizando python y la librería OpenOPC.

Para la validación de los componentes de hardware, previamente se utilizó el entorno de simulación Isis-Ares de Proteus, con el cual se validó el diseño electrónico y posteriormente se implementó el hardware de la tarjeta de adecuación electrónica, probando uno a uno cada sensor. Otro aspecto a considerar es el chasis del sistema, ya que existen estándares para la ubicación de la altura de los sensores, los cuales fueron considerados al momento de construir el prototipo. Durante la etapa de construcción del prototipo, se utilizaron los códigos de los fabricantes de los sensores, que luego han sido adaptarlos para la tarjeta Arduino, ya que se encontraron errores en su código de programación, tal es el caso del sensor de Lluvia y Viento BH4TDV, en el cual se realizó la siguiente mejora:

Código Mejorado del Sensor BH4TDV

```
void getBuffer() //Obtenemos el estado de los datos ambientales Viento-Lluvia
{
  int index;
  VSerial.flush();
  int x=0;
  while (VSerial.available() <= 0) {
    x=x+1;
```

```
    delay(1);
    if (x == 5000)
      goto msgError;
  }
    for (index = 0;index < 35;index ++)
  {
    delay(3);
    databuffer[index] = VSerial.read();
    }
    if (x == 5000) {
  msgError: Serial.println('Error');
  }
}
```

Código de Transferencia de Datos desde el Arduino hacia el RaspberryPi

```
# obtiene datos de los sensores desde el Arduino via USB Serial
def get_sensores(self):
  print("Abriendo el puerto serial...")
  arduino = serial.Serial('/dev/ttyACM0',9600)
  arduino.open()
  time.sleep(2.5) #Tiempo para conectar el puerto serial del arduino

  date=datetime.datetime.now()
  self.timestamp=date.strftime("%Y-%m-%d %H:%M:%S")
  print("Enviando datos al puerto serial...")
  arduino.write(bytes('t'.encode('ascii')))
  time.sleep(2.5)
  msg = arduino.read(arduino.inWaiting())
  self.temperatura = msg
  print "T = " + str(self.temperatura)
  arduino.write(comandoP)

  time.sleep(2.5)
  str(arduino.readline().decode('ascii'))
  msg = arduino.read(arduino.inWaiting())
  self.presion = msg
```

```
print "P = " + str(self.presion) …

arduino.write(bytes('V'.encode('ascii')))
time.sleep(2.5)
valor=""
valor = arduino.read(arduino.inWaiting())
print("Se recibio datos del puerto serial...")
datos=valor.split(',')
self.direccionViento = datos[0]
self.velocidadViento = datos[1]
self.lluvia = datos[2]
self.temperaturaEstacion = datos[3]
self.humedadEstacion = datos[4]
self.presionEstacion = datos[5]
arduino.close()
```

3.3 Validación de las Especificaciones Técnicas

Luego de haber realizado el diseño y construcción del prototipo en base a las especificaciones técnicas, se ha procedido a verificar cada componente a nivel de hardware y software. En esta sección se valida cada sub sistema mediante pruebas aisladas de cada módulo, y luego se realiza las pruebas de integración. En la **Fig. 3.16** se muestra el escenario de pruebas de integración del prototipo, en el cual se ha desplegado un Servidor OPC-SCADA para pruebas de lectura y escritura desde el Servidor Web del RaspberryPi que contiene el Cliente Scada OPC. Asimismo, se ha habilitado una computadora para pruebas de navegación hacia el servidor Web en el RaspberryPi.

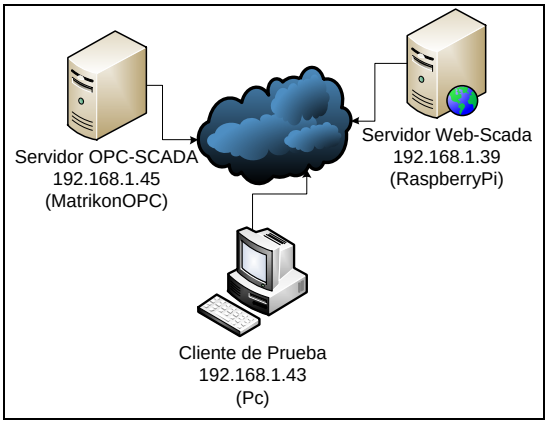


Fig. 3.16 Escenario de Pruebas de Integración del Prototipo.

En la **Fig. 3.17**, se muestra la visualización gráfica del código PHP que corresponde al

tablero de la interface web de los sensores, desde una computadora de desarrollo en Windows con navegador Chrome, que muestra los datos digitalizados en tiempo real de los sensores, con un tiempo de actualización de 5 minutos, que ha sido programado en el Crontab sobre Linux Apache en el micro controlador Raspberry Pi. Este enfoque ayudó a reducir los tiempos de desarrollo para comprobar la interface web, y la ventaja de que es transparente su migración, al contar con un esquema de servidor de aplicación estándar.

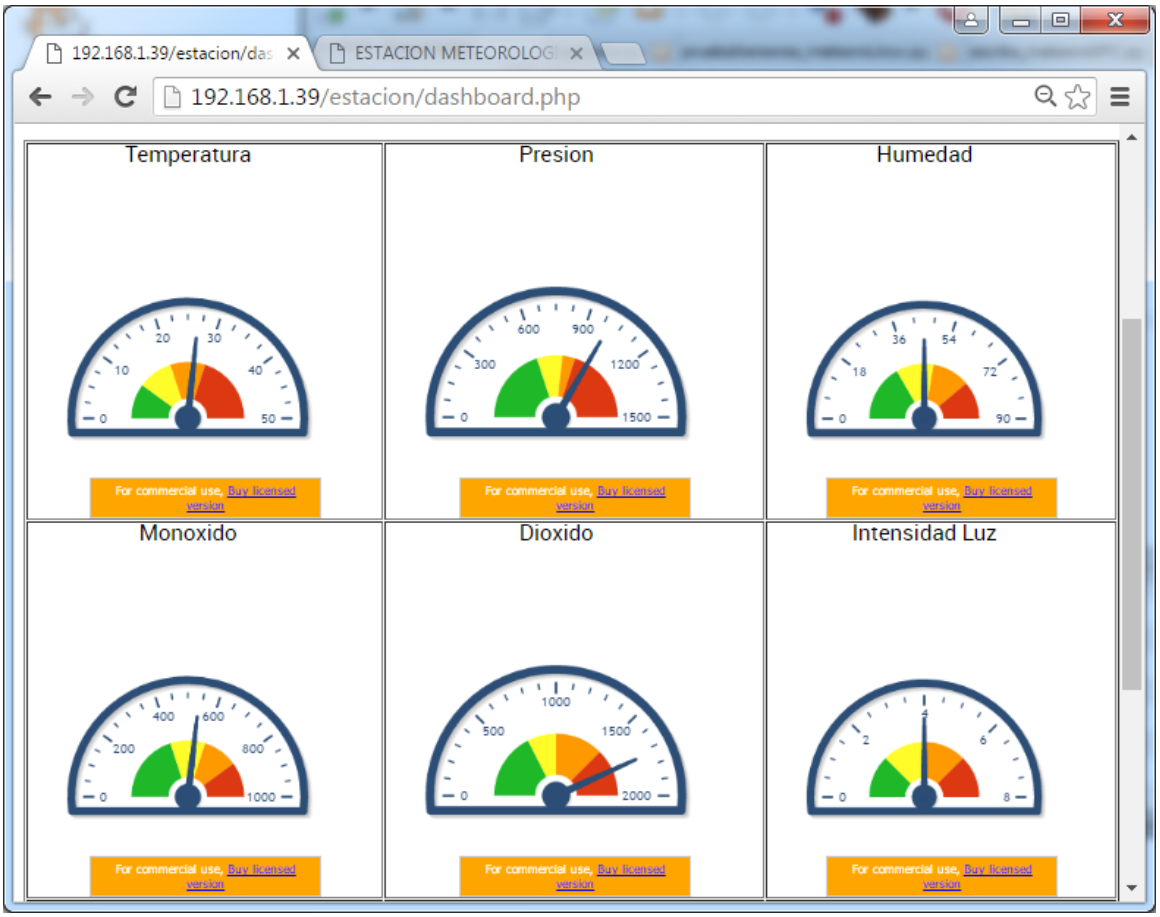


Fig. 3.17 Visualización del Dashboard de la Estación Meteorológica.

En la **Fig. 3.18**, se muestra el histórico de la interface web que contiene una casilla con el rango de fecha, así como tipo de gráfica, con la posibilidad de limitar el número de muestras a visualizar por parte del usuario. Se ha comprobado que el prototipo funciona y es de fácil uso a través de un navegador web o en un escenario más especializado, desde un servidor scada. La ventaja de haber utilizado tecnología de hardware libre y software libre, ha permitido su versatilidad e interacción entre diversos fabricantes tanto de sensores a nivel de hardware como de librerías de software, que fueron utilizados en las librerías de acceso a puertos seriales-usb, componente dashboard gráfico, etc.

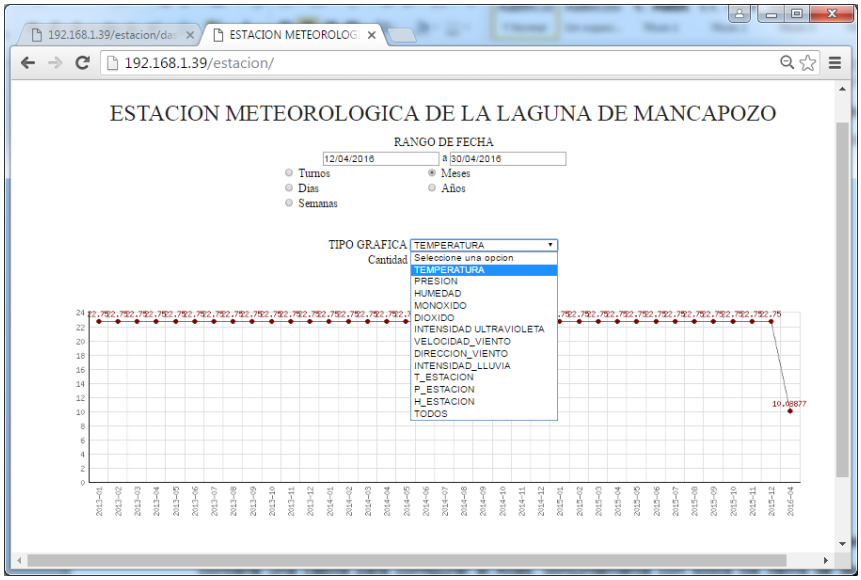


Fig. 3.18 Visualización del Histórico de la Interface Web.

En los resultados también se ha podido constatar que se ha logrado experiencia de investigación en cada una de las alternativas de solución que no fueron elegidas, sin las cuales no se hubiera podido llegar al resultado esperado. En la **Fig. 3.19** se muestra el entorno de programación web-scada sobre Linux, donde se ha utilizado Geany debido a su portabilidad entre sistemas operativos Windows y Linux.

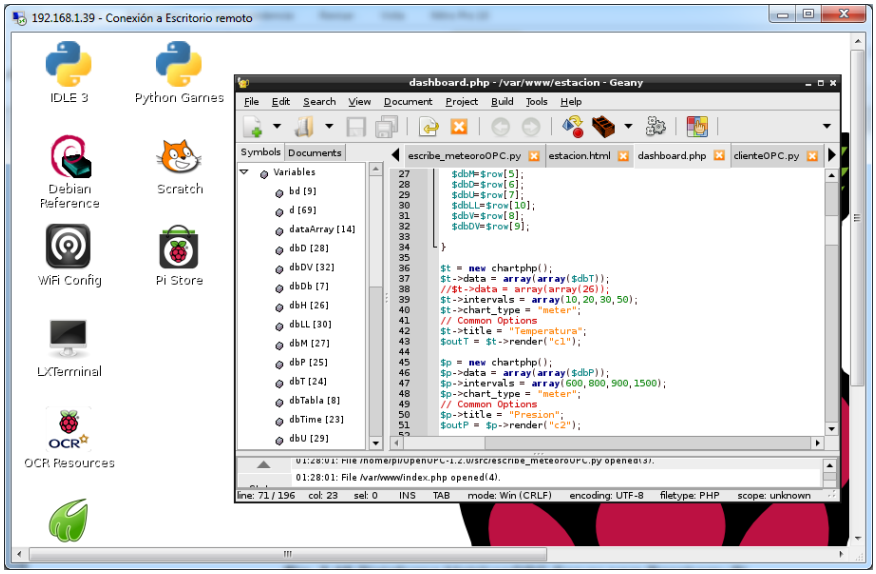


Fig. 3.19 Entorno de Programación Web-Scada del Raspberry-Pi.

En la **Fig. 3.20** se expone los nueve parámetros de sensores climatológicos en el Servidor Scada OPC de MatrikonOPC, los cuales son actualizados desde el módulo del OpenOPC Cliente programado sobre Python en Linux del RaspberryPi. Los pasos para

habilitar la funcionalidad scada en el sistema embebido, requiere previamente instalar las librerías de openOPC, el cual requiere una versión actualizada de Python, así como de la librería pyro, que permite acceder a objetos remotamente. También ha sido necesario habilitar en el Servidor OPC, el servidor Gateway OPC, para lo cual fue necesario registrar la librería de autentificación “regsvr32 “gbda_aut.dll” ” con permisos de usuario de administrador en el sistema operativo.

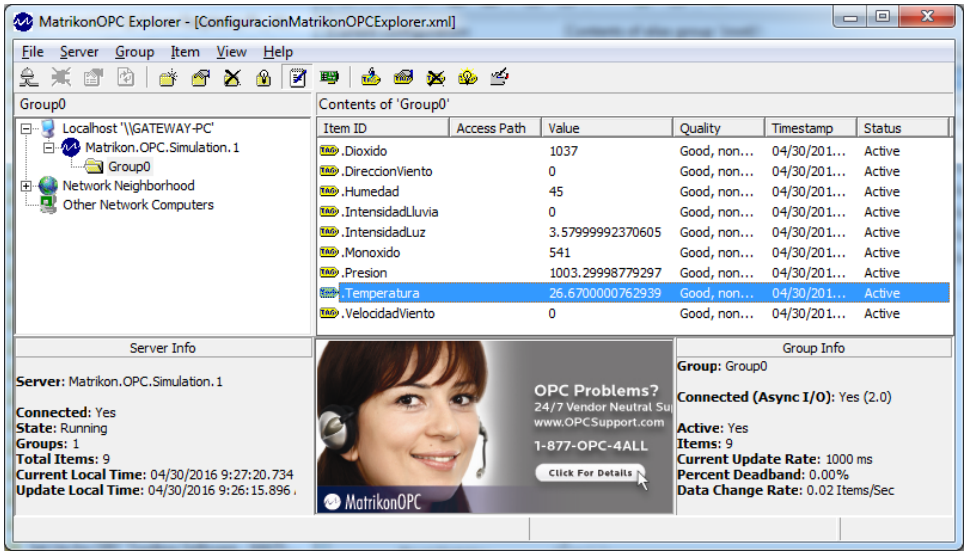


Fig. 3.20 Visualizador Scada MatrikonOPC Explorer.

Las pruebas de integración de los módulos han sido validadas satisfactoriamente, tal como se aprecia en la **Fig. 3.21** y **Fig. 3.22**, donde se ha guardado los datos digitalizados tomados de los sensores desde el RaspberryPi hacia el Servidor OPC mediante el módulo de la librería Python OpenOPC.

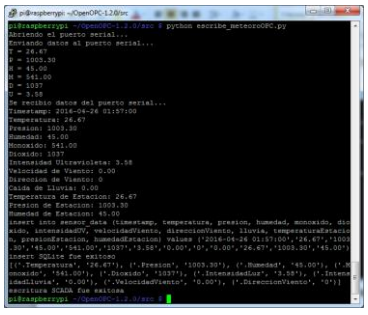


Fig. 3.21 Escritura Scada desde Raspberry-Pi con Putty.

En la digitalización de los datos de las variables monitoreadas, se ha utilizado Hiperterminal para conectar la Pc vía USB hacia el Arduino, y cutecom en entorno Linux,

también se validó las respuestas de los datos para cada escenario programado, tal como se aprecia en las figuras: **Fig. 3.22** y **Fig. 3.23**. Por ejemplo: al enviar T por el puerto serial, se recibe el valor de la temperatura leída por el sensor, al enviar P se recibe el valor tomado de la presión, al enviar H, se recibe el valor registrado por el sensor de humedad, y así sucesivamente.

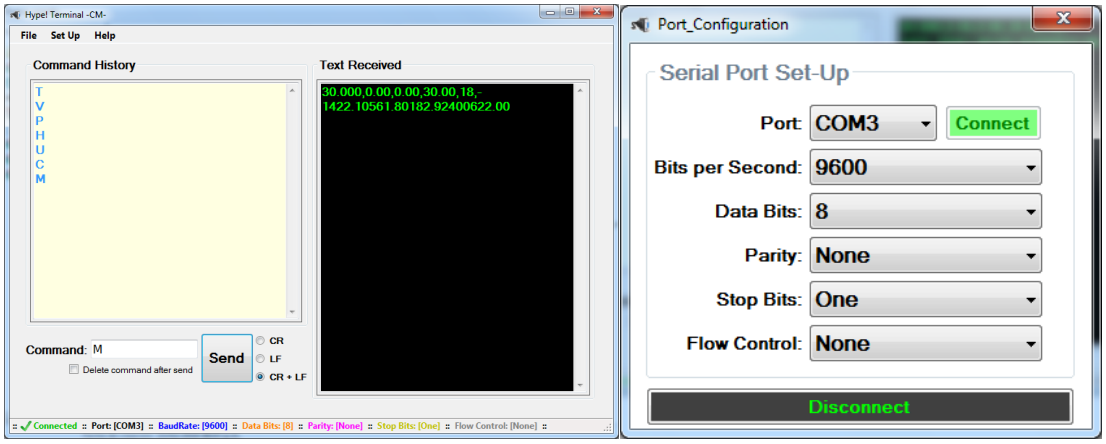


Fig. 3.22 Lectura de Sensores Arduino con Hiperterminal en Windows.

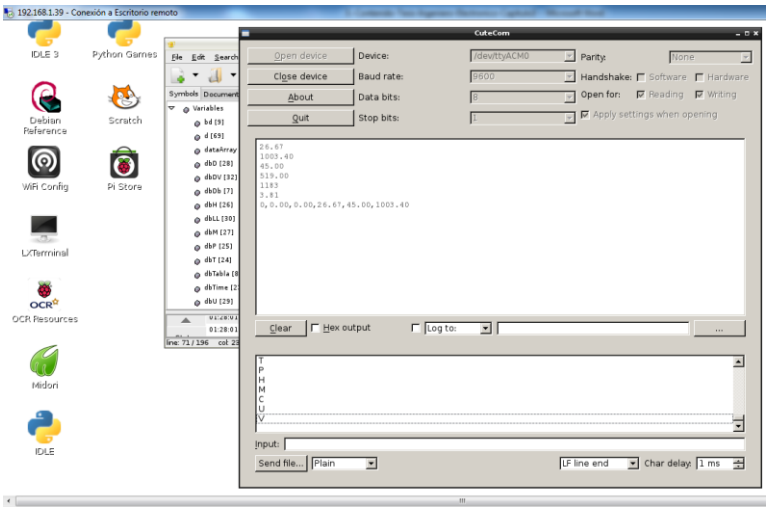


Fig. 3.23 Lectura de Sensores Arduino con Cutecom en Linux.

Cabe recalcar, que al momento de poner en operación real una estación meteorológica, se requiere calibrar cada uno de los sensores en la zona misma de despliegue, ya que los parámetros ambientales influyen en la operación de los sensores (vientos fuertes, temperaturas pronunciadas, rayos, etc.). Por ello, como el proyecto es un prototipo, la ventaja es que se posee acceso a los códigos fuente para ajustar los valores digitales al momento de desplegar la estación en un entorno real de aplicación. En la **Fig. 3.24**, se aprecia el test de comunicación de la base de datos, el servidor web y scada.

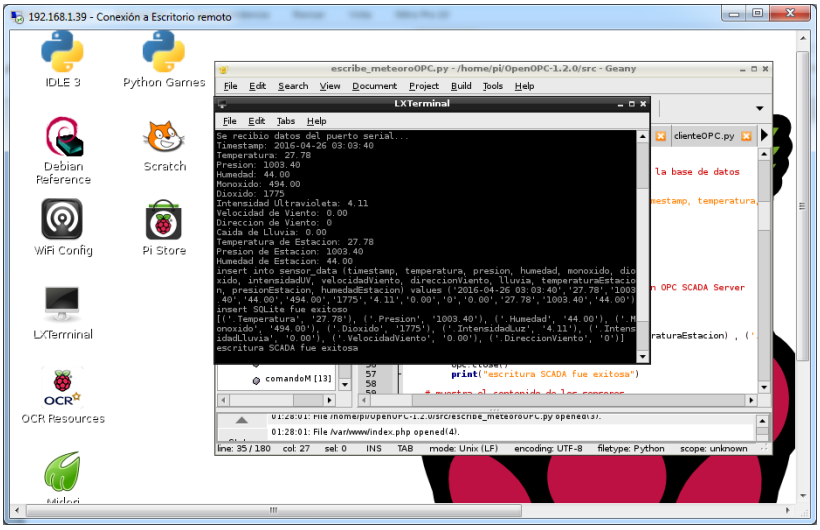


Fig. 3.24 Lectura de Sensores Arduino desde el Raspberry.

En la Tesis se ha utilizado un Radio Enlace punto a punto, con la tecnología NanoBeam modelo NBE-MS-19, que soporta los estándares: 802.11a, 802.11n/a/, que permite tasas de transmisión de datos de hasta 54 Mbps en caso se requiera procesar señales de video o fotográficas. El prototipo ha sido diseñado para un radio enlace punto a punto de hasta 15 Km, en particular para monitorear la Laguna de Mancapozo en la ciudad de Huánuco. En la **Fig. 3.25** se muestra la validación de cobertura utilizando Google Earth y las coordenadas de los puntos de origen y destino, con una distancia exacta de 11.41 Km.

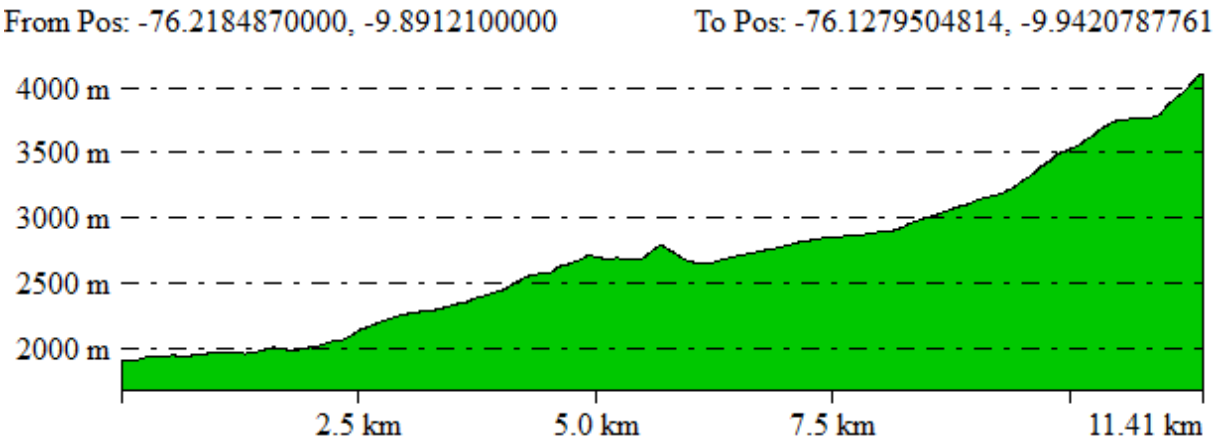


Fig. 3.25 Validación de la Línea de Vista del Radio Enlace punto a punto.

CAPÍTULO IV
RESULTADOS DE LA TESIS

4.1 Resumen de los resultados y sus implicancias

Para mostrar los resultados del trabajo de investigación, se ha dividido en dos partes: la primera orientada a demostrar la validez tecnológica de la ingeniería del proyecto mediante la validación de cada una de las especificaciones técnicas y la segunda orientada a comprobar el método de diseño del proceso de ingeniería que ha sido utilizado en esta tesis, con la finalidad de mostrar su pertinencia con enfoque de ingeniería. A continuación, se presenta el costo de inversión en Soles de los componentes utilizados en la elaboración del prototipo (ver **Tabla N° 4.1**):

Tabla N° 4.1 Presupuesto de Inversión del Prototipo.

Componente	**Costo en Soles S/.**
1 Placa Arduino UNO	80.00
1 Sensor DHT22 de Humedad y Temperatura	30.00
1 Sensor de Gas CO	20.00
1 Sensor de Gas CO2	40.00
1 Sensor LDR de luminosidad	5.00
1 Placa de Adquisición y Pruebas	60.00
1 Placa Raspberry Pi	80.00
1 Sensor de Lluvia y Viento	1100.00
2 Inversores	80.00
1 Kit Ubiquiti Radio Enlace 15 Km	600.00
2 Paneles Solares y 2 Baterías 12 VDC	800.00
1 Chasis de Metal	500.00
1 Fuente de Alimentación 5VDC	40.00
Costo Total	3435.00

En la **Tabla N° 4.2** se muestra los resultados de cumplimiento de las especificaciones técnicas de la tesis, que fueron desarrolladas en el **Capítulo III**, por lo cual se concluye que el prototipo ha cubierto satisfactoriamente los requerimientos planteados del proyecto.

Tabla N° 4.2 Resultados de Cumplimiento de las Especificaciones Técnicas.

Especificación Técnica	Detalle de Diseño	Resultado
ET1: Radio enlace punto a punto hasta un máximo de 15 Km con línea de vista directa.	Sub Sistema de Transmisión de Datos -Radio Enlace	Ok, Se utilizó el equipo NanoBeam M5 Ubiquiti.
ET2: Sistema dual: Panel Solar con salida 12 VDC y Batería de 12 VDC, con inversor 220 VAC. Capacidad de operación las 24 horas del día.	Sub Sistema de Energía -Panel Solar -Batería	Ok, Se utilizó un modelo redundante de panel solar y batería.
ET3: Tarjeta de adquisición de sensores Arduino UNO, con capacidad de integrar sensores de Temperatura, Presión, Humedad, Monóxido, Dióxido, intensidad de Luz, velocidad y dirección de viento.	Sub Sistema de Adquisición de Sensores -T, P, H -CO2, CO Sub Sistema de Medición de Viento y Lluvia Sub Sistema de Medición de Luz Ultravioleta	Ok, Se desarrollaron los códigos fuente para la digitalización de los sensores, así como la transmisión vía puerto serial-usb.
ET4: Micro controlador RaspberryPi con Linux, con servidor web Apache-PHP, Cliente OpenOPC-Python para SCADA y Base de Datos SQLite para registro de datos meteorológicos.	Sub Sistema de Control de Datos y Data-Logger -RaspberryPi -Arduino UNO	Ok, Los datos de los sensores se graban en la base de datos SQLite, así como en el Servidor OPC-Scada.
ET5: El diseño deberá permitir la posibilidad de habilitar una cámara fotográfica-video y	Módulos complementarios -Bluetooth -Cámara	Ok, se habilitó el diseño electrónico para adicionar la comunicación

comunicación inalámbrica vía bluetooth.		bluetooth y la cámara.
ET6: La solución deberá utilizar lenguajes de programación de código abierto: Lenguaje C++ para Micro controlador Arduino; Lenguaje Python para micro controlador RaspberryPi y OpenOPC; Lenguaje PHP para servidor web; Lenguaje SQL para Base de datos SQLite.	Entorno de Desarrollo -Python -PHP -C++ -SQL	Ok, Se comprobó que la solución utiliza lenguajes de programación de código abierto.
ET7: El diseño deberá contemplar un esquema de sub sistemas para que la solución integral pueda desplegarse fácilmente sobre un chasis estándar para una estación meteorológica.	Enfoque modular vía sub sistemas - Sub Sistema de Transmisión de Datos - Sub Sistema de Energía - Sub Sistema Adquisición de Sensores - Sub Sistema de Medición de Viento y Lluvia - Sub Sistema de Medición de Luz Ultravioleta - Sub Sistema de Control de Datos y Data-Logger	Ok, se comprobó que la solución propuesta, establece un esquema de diseño modular basado en sub sistemas.

Asimismo, con las pruebas del prototipo de la estación meteorológica, se concluye que es posible integrar sistémicamente a través de una plataforma de software, los algoritmos computaciones que permitan un monitoreo y pronóstico. Esto implica que es posible plasmar un sistema de almacenamiento de datos meteorológicos. En las gráficas de la **Fig. 4.1** y **Fig. 4.2** se muestran las pruebas de integración de los sensores y las pruebas de radio enlace respectivamente, para mostrar la viabilidad de la implementación de los casos de uso propuestos en el alcance de la Tesis.

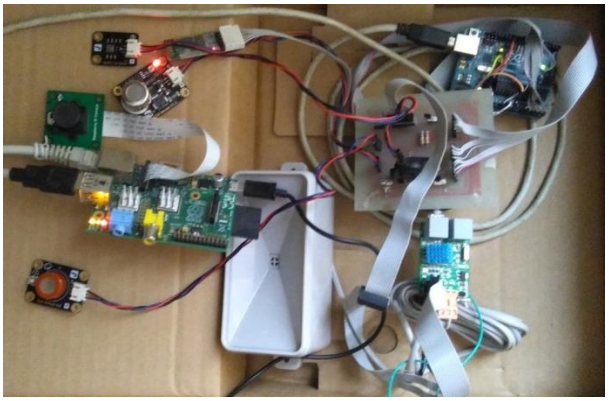


Fig. 4.1 Integración de la Tarjeta de Adquisición – Arduino - RaspberryPi.

Fig. 4.2 Pruebas de Radio Enlace en la Laguna de Mancapozo.

Como aporte complementario a la Tesis, se realizó un pronóstico de temperatura con la técnica regresión lineal simple y múltiple, cuyo detalle se encuentra en el **Anexo A.5**. También se llevó a cabo el acondicionamiento de la señal en la entrada de los sensores; y la detección de puntos significativos, que incluye los valores máximos y mínimos del día mediante la programación del Crontab Linux. En conclusión, a través del prototipo de la estación meteorológica basada en las señales de temperatura, humedad, presión, intensidad ultravioleta, monóxido, dióxido de carbono e intensidad de lluvia y dirección y velocidad de viento, se demostró la viabilidad técnica de la solución, sin embargo, queda pendiente la sistematización de más sensores en una etapa de producción, tales como: nieve, PH, etc. En la **Fig. 4.3** se muestra la etapa de validación de la interface de usuario del Panel de Control, utilizando el navegador Chrome y en la **Fig. 4.4** se muestra el chasis de la estación meteorológica que alberga todos los componentes del sistema; sistema de energizado, sistema de radio enlace, sistema de control, sistema de adquisición de datos, etc.

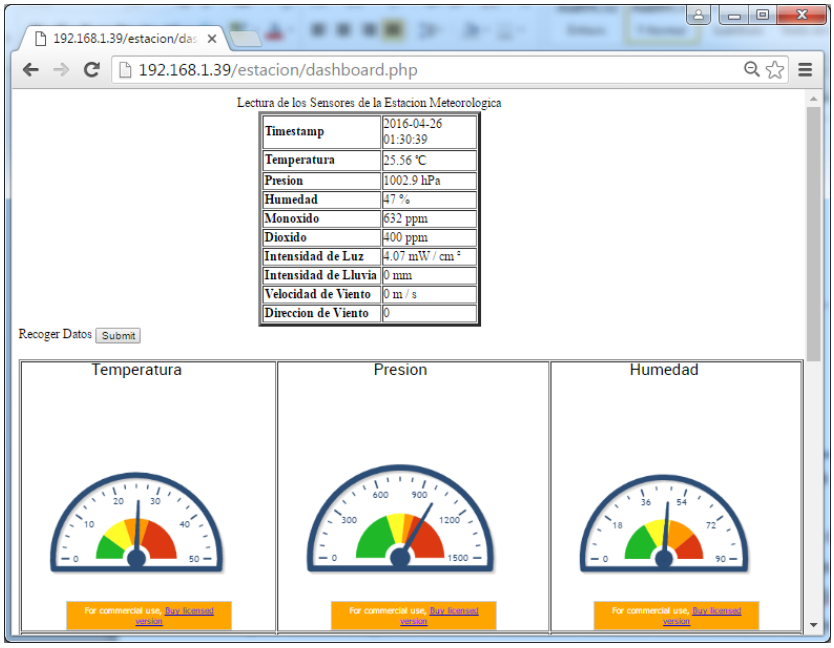


Fig. 4.3 Panel de Control Dashboard de la Estación Meteorológica.

Fig. 4.4 Vista Frontal y de Perfil del Chasis del Prototipo.

En la **Fig. 4.5** se muestra el modelado y visualización de la base de datos SQLite Studio, que es un entorno de programación de base de datos SQLite de fácil uso. Si bien es

cierto se pudo haber escogido una base de datos más robusta como MySQL u Oracle, pero debido a que la solución está sobre un micro controlador, es más recomendable utilizar una base de datos ligera como el SQLite, para no sobre cargar el consumo de CPU y espacio en disco.

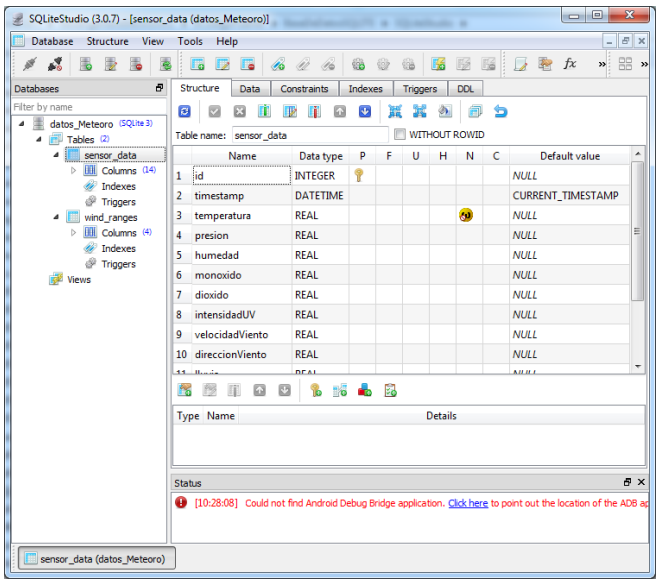


Fig. 4.5 Modelado de la Base de Datos SQLite Studio.

En la **Tabla N° 4.3**, se ha desarrollado la validación del **Método del Proceso de Diseño de Ingeniería**, para comprobar la consistencia del aspecto técnico de ingeniería, el cual ha sido validado en cada una de sus etapas, que se detalla a continuación:

Tabla N° 4.3 Validación del Método del Proceso de Diseño de Ingeniería.

Etapa	Detalle del Método	Evaluación
Definición del Problema	¿Cuál es el problema? Necesidad de contar con estaciones meteorológicas experimentales para investigación académica. ¿Por qué es importante solucionarlo? Permitirá ahorrar costos operativos y mejorar el servicio meteorológico.	Ok, Mediante el diseño y construcción del prototipo de la Estación Meteorológica, se pudo probar la viabilidad tecnológica de la solución propuesta. En el **Capítulo I** se esbozó el análisis y descripción del problema.
Investigación Temática	En el **Capítulo II**, se estudió la evolución histórica de las técnicas de predicción	Ok, Se revisó los avances temáticos realizados a la

	meteorológica, así como las alternativas de solución al problema planteado.	fecha, así como las alternativas de soluciones existentes.
Especificaciones Técnicas	En el **Capítulo III**, se realizó el análisis y diseño a nivel de hardware y software, poniendo énfasis en los sub sistemas de energía, transmisión de datos, control y data logger, adquisición de sensores, medición de viento y lluvia, así como medición de luz ultravioleta.	Ok, Se logró realizar las especificaciones técnicas de los sensores, así como de los componentes de hardware y software de la estación meteorológica.
Alternativas de Solución	En la **sección 2.3** se esbozaron diversas alternativas de solución para la plataforma web-scada del proyecto.	Ok, Se analizaron diversos escenarios de solución.
Selección de la Mejor Solución	En el **Capítulo III** se realizaron los diagramas de flujo de los componentes de la alternativa seleccionada.	Ok, Se desarrolló el diseño del prototipo, se realizaron los diagramas de flujo.
Desarrollo de la Solución	En la **sección 3.2** se esbozaron detalles de la construcción del prototipo en base al diseño detalladas en la **sección 3.1** para el cumplimiento de las Especificaciones Técnicas desarrolladas en la **sección 2.3**.	Ok, Se realizó la construcción del prototipo utilizando los diagramas de diseño.
Construcción del Prototipo	Se construyó una versión prototipo de la solución, permitiendo verificar el flujo de la solución integral propuesta.	Ok, Se realizó el proto tipado de la solución en RaspberryPi y Arduino.
Testeo y Rediseño	Se realizaron varias interacciones y rediseños para obtener la solución final, cuyo detalle técnico fue plasmado en el código fuente del proyecto.	Ok, Se realizaron las pruebas y el rediseño, concluyendo la versión del código final en el Anexo B.
Comunicación de Resultados	Se comunicó los resultados del proyecto a los miembros del jurado en la versión Perfil y versión final de la Tesis.	Ok, Se redactó el informe de Perfil y el informe final de la Tesis.

4.2 Logros obtenidos de la Tesis

A continuación, se citan los logros obtenidos en la Tesis:

- A nivel de marco metodológico, se ha utilizado el Método del proceso de diseño de ingeniería, orientado a la ingeniería del proyecto.
- A nivel de marco teórico, se ha repasado los aspectos históricos de la evolución de los sistemas automáticos para la meteorología, así como la evolución de los sistemas SCADA, realizando una comparación y luego una selección del mejor escenario web-scada para el prototipo realizado.
- Se ha elaborado las especificaciones técnicas del prototipo de la estación meteorológica en base a los diagramas de flujo tanto de hardware como software.
- Se ha construido con éxito el prototipo de la Estación Meteorológica, que incluye el sistema de energizado de panel solar con batería, el radio enlace, así como los sensores de temperatura, presión y humedad, los cuales fueron digitalizados vía el micro controlador Arduino - Raspberry.
- Se ha aplicado la técnica de regresión lineal simple y múltiple para realizar pronóstico de las señales meteorológicas digitalizadas.
- Se ha manejado la tecnología de la plataforma web con PHP y base de datos SQLite para el sistema embebido Raspberry Pi, que incluye la interconexión vía Ethernet con el Radio Enlace punto a punto del fabricante Ubiquiti, junto con el panel solar y el sistema de baterías.
- Se ha manejado la tecnología del microcontrolador Arduino que adquiere las señales de los sensores y los transmite al Raspberry Pi.
- Se ha logrado realizar la integración de la solución con el cliente python OpenOPC, que permitió cumplir con el requerimiento de integración SCADA a nivel de OPC Server, enviando los datos de los sensores hacia el servidor SCADA-OPC del fabricante MatrikonOPC.
- Se ha logrado captar el interés de los actores involucrados, tales como: SENAMHI-Huánuco, Gobierno Regional Huánuco, Autoridades del Centro Poblado de Malconga, distrito de Amarilis, universidades, entre otros.

En la **Tabla N° 4.4** se citan las ventajas del prototipo del sistema desarrollado, en concreto:

Tabla N° 4.4 Ventajas del Prototipo de la Estación Meteorológica.

Descripción	Ventajas	Mejoras Futuras
Simulación Electrónica de las Interfaces de Adquisición de los Sensores basado en un microcontrolador Arduino sobre Proteus.	Se pueden adicionar diversos tipos de sensores: viento, radiación solar, etc., de modo que permita realizar mejoras continuas.	Se podría agregar sensores para aplicaciones de pronóstico geo referenciado de clima para el sector agrario del centro poblado de Malconga.
Estación Meteorológica para procesado de los sensores de Temperatura, Humedad y Presión con Raspberry Pi.	Reducción en costos de infraestructura y de operación y mantenimiento por permitir acceso al código fuente.	En el presente proyecto solo se utilizó sensores cuyos datos se almacenan en BD SQLite y OPC-Scada, queda por investigar otros escenarios de integración.
Pronóstico de Clima usando el algoritmo de Regresión Lineal Simple y Múltiple sobre MATLAB – R - Excel.	Apoyo en el diagnóstico y proyección de clima usando técnicas computacionales.	En el presente proyecto se utiliza una base de datos SQLite, pero podría mejorarse su desempeño vía MySQL ampliando el tamaño de memoria SD.

CONCLUSIONES Y RECOMENDACIONES

Después de realizar el prototipo de la Estación Meteorológica, se ha llegado a lo siguiente:

Conclusiones

1) Se ha demostrado la viabilidad técnica de diseño y construcción de una estación meteorológica con arquitectura de código abierto, que contempla aspectos de hardware y software para el monitoreo climatológico, para su posterior uso en el sistema de gestión ambiental y de recursos naturales por parte de las instituciones involucradas en este servicio.
2) Al comparar con soluciones comerciales existentes, el costo de inversión está por debajo de éstas, sin embargo, el costo de operación y mantenimiento requiere un monto adicional debido a que se requiere personal especializado para realizar mejoras en forma progresiva en los códigos de programación.
3) En cuanto a la plataforma Web-Scada y de base de datos SQLite, se ha probado satisfactoriamente el desempeño del sistema embedido Raspberry Pi - Arduino, con lo cual, se ha demostrado con éxito que es posible utilizar esta plataforma de código abierto, para monitorear y almacenar señales meteorológicas digitalizadas, de modo que posteriormente puedan servir como data de análisis para probar algoritmos avanzados de pronóstico, haciendo una combinación de varias técnicas.
4) Se ha demostrado que es posible adicionar módulos a medida, como es la integración scada o de monitoreo climatológico; sin embargo, en un contexto regional, se requiere un cluster de estaciones meteorológicas para poder ajustar y mejorar los algoritmos de pronóstico.
5) Durante las pruebas de validación de los componentes de hardware y software, ha sido útil esbozar un esquema sistemático de testeo que permitió reducir los tiempos de desarrollo. De otro modo se hubiera tomado más tiempo en la integración de los sub componentes.

6) En el supuesto escenario comercial del prototipo, se tiene una solución a medida que va a requerir cierta dependencia con el proveedor para la instalación, capacitación, operación y mantenimiento del sistema, lo cual ofrece la oportunidad de tener la exclusividad en los servicios de pos venta.

Recomendaciones

A continuación, se detallan las recomendaciones para trabajos futuros:

1) Para un proyecto futuro se sugiere el desarrollo de un sistema embebido basado en micro controladores, que incorpore toda la gama de sensores de modo que permita verificar la robustez de la solución propuesta, para un análisis más integral, se podría por ejemplo agregar sensores de PH, nieve, etc., de modo que permita un monitoreo integral multi variable.
2) Para continuar la investigación de esta Tesis, podrían darse mejoras en los siguientes escenarios:
   - Mejora en los Sensores de la Estación Meteorológica: Por ejemplo, se podría plantear una investigación para construir sensores propios multi variables.
   - Mejora en el Chasis de la Estación Meteorológica: Por ejemplo, se podría investigar en el chasis auto reconfigurable multi propósito para costa, sierra y selva, en escenarios móviles (vía drones) o estáticos.
   - Mejora en los Algoritmos de Pronóstico: utilizando técnicas de computación suave tales como Redes Neuronales, Algoritmos de Control Neuro Difuso, etc.
   - Mejora en el entorno del lenguaje de programación desde un entorno textual hacia un entorno gráfico tales como: Visuino, Scratch, entre otros.

ANEXO A
ESPECIFICACIONES TÉCNICAS

A.1 Descripción del Hardware del Raspberry Pi y Arduino

En este anexo se presentan los conceptos de las tecnologías de los micros controladores de la familia Arduino-Atmel y Raspberry-Pi-ARM Cortex.

Características generales del Arduino

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en hardware flexible y fácil de usar. Está Basado en una tarjeta con un microcontrolador que permite conectar sensores, actuadores y otros elementos mediante sus entradas y salidas, analógicas y digitales. Al ser Open-Hardware, tanto su diseño, como su distribución son de libre acceso. Es decir, puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin necesidad de adquirir ninguna licencia. Las características más relevantes de la plataforma Arduino son las siguientes:

- La filosofia OpenSource – Codigo Abierto- que lo sustenta.
- La comunidad formada a su alrededor.
- La sencillez del lenguaje de programación.
- El hardware de bajo costo.
- Microcontrolador: ATMega328
- Voltaje de operación: 5V DC
- Voltaje de alimentación: 7 – 12V DC
- Pines digitales I/O: 14 (6 Con PWM)
- Pines entrada analógica: 6
- Interfaz de programación: USB
- Frecuencia del Reloj: 16 Mhz

Arduino cuenta con un entorno de desarrollo nativo creado en Java, por lo que es multiplataforma. Los Shields o tarjetas de expansión son módulos fabricados por terceros que se pueden apilar encima de la placa Arduino y le proporcionan una funcionalidad

determinada. Por ejemplo:

- Conexiones inalámbricas.
- Control de sensores y motores.
- Lectura y escritura en memorias.

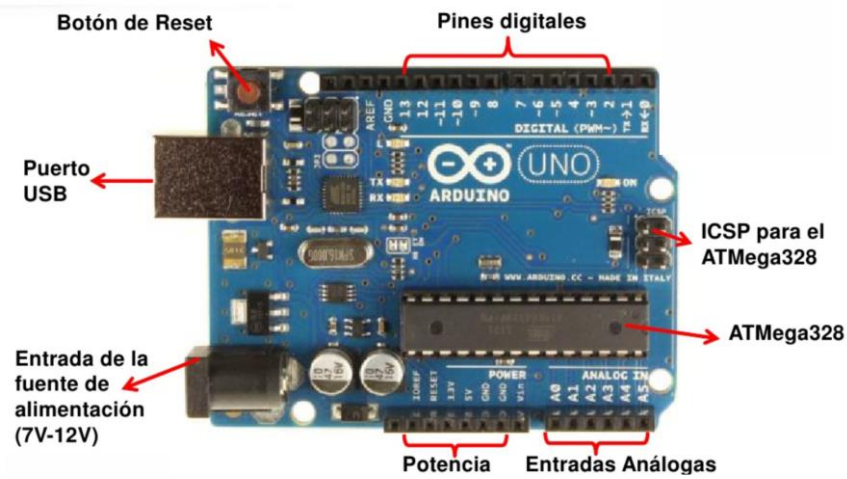


Fig. A.1 Componentes del Microcontrolador Arduino.

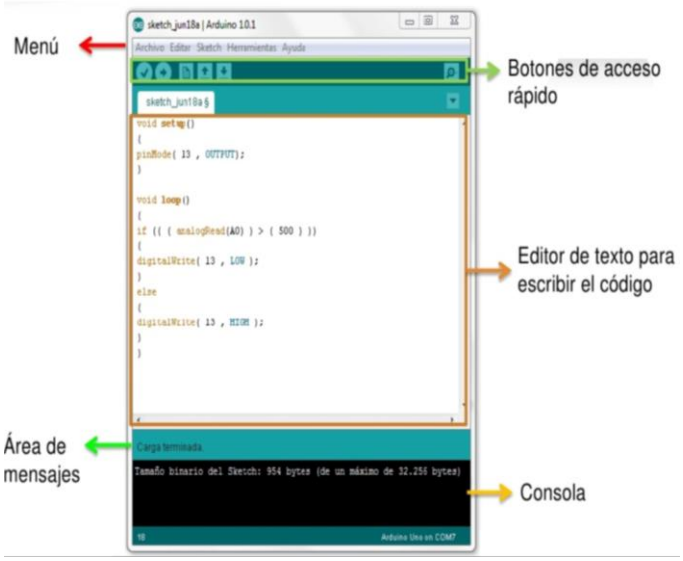


Fig. A.2 Entorno de Programación del Arduino.

Características generales del Raspberri Pi

Raspberry Pi es un computador de bajo costo sobre Linux, que llegó con la idea de revolucionar el sector educativo y que, en muy poco tiempo, se ha convertido, junto a Arduino, en un exponente del hardware libre y en la base de un buen número de

proyectos.

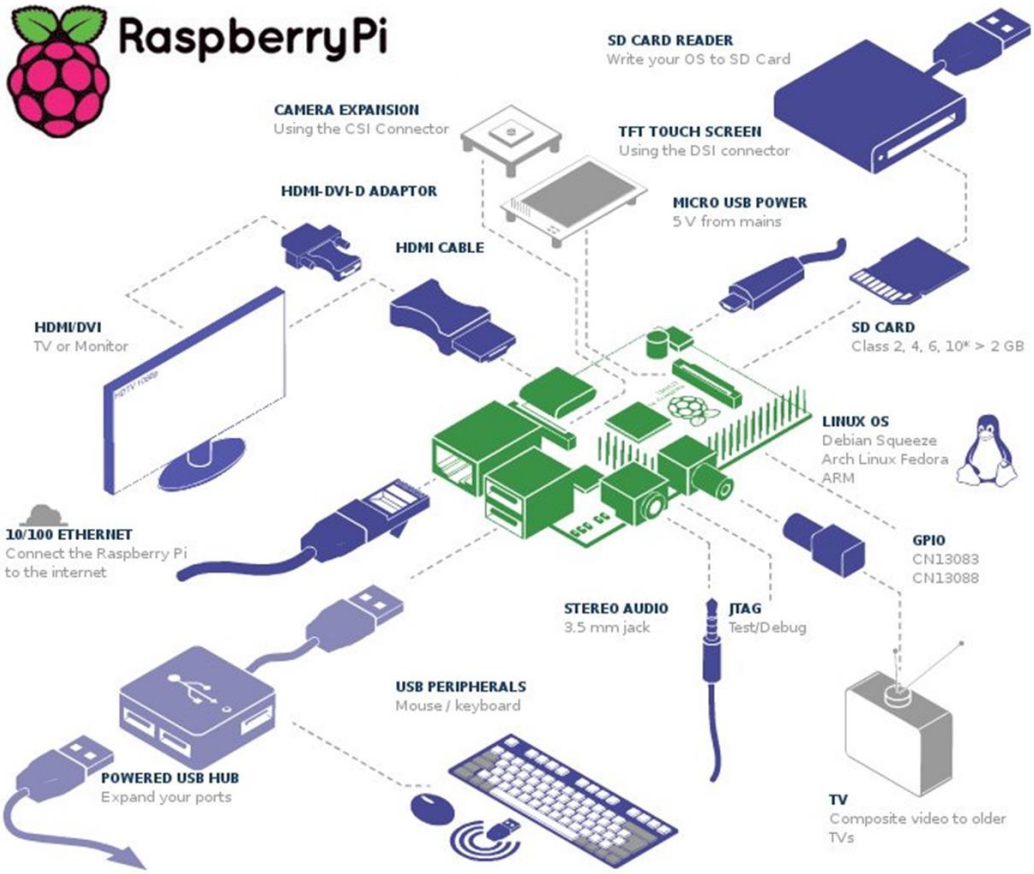


Fig. A.3 Componentes del Raspberry Pi.

La tarjeta Raspberry Pi consta de las siguientes características:

- Dual core ARM cortex-A7 processor, NEON, VFPv4, 512KB L2 cache.
- Mali400mp2, OpenGL ES GPU.
- 1GB DDR3 @480MHz.
- HDMI 1080p Output.
- 100M Ethernet.
- 4Gb Nand Flash.
- 2 USB Host, 1 micro SD slot, 1 SATA.
- 96 extend pin including I2C, SPI, RGB/LVDS, CSI/TS, FM-IN, ADC, CVBS, VGA, SPDIF-OUT, R-TP.
- Soporta Android, Ubuntu y otras distribuciones de Linux.

A.2 Descripción del Radio Enlace, Panel Solar y Batería

Fig. A.4 Detalle de los Componentes del Radio Enlace Ubiquiti NanoBeam M5.

El **Radio Enlace** NanoBeam M tiene un diseño compacto todo-en-uno con un ancho de haz uniforme, eficiente y un procesador más rápido. Este nuevo Ubiquiti NanoBeam M incorpora un sistema de sujeción a los 3 ejes (X, Y, Z) para permitir una mayor precisión en los enlaces Wi-Fi de larga distancia. La antena y el rendimiento de este nuevo equipo han sido mejoradas para proporcionar un mayor rendimiento. Existen diversos Modos de Operación: Punto de Acceso, Estación & Repetidor (WDS). Señalización propietaria: AirMax (MIMO TDMA). Sistema airOS 5. Tecnología Gigabit Ethernet (1,000 Mbps). Potencia de Salida: 400 mW. Ancho de Banda: Hasta 300 Mbps. Canal ajustable de 5 hasta 40 MHz. Antena MIMO parabólica con 25 dBi de ganancia. Doble Polaridad Simultánea (Vertical y Horizontal, 2x2). Seguridad: WEP, WPA, WPA2 y MAC ACL. Alineación de antenas Visual y Audible (Software). Temperatura: -40°C a 75°C. Alimentación: 24 Vcc, 0.5 A (incluye PoE Gigabit).

El **Panel Solar** está constituido por varias células iguales conectadas eléctricamente entre sí, en serie y/o en paralelo, de forma que la tensión y corriente suministrada por el panel se incrementa hasta ajustarse al valor deseado. La mayor parte de los paneles solares se construyen asociando primero células en serie hasta conseguir el nivel de tensión deseado y luego asociando en paralelo varias asociaciones serie de células para alcanzar el nivel de corriente deseado. Además, el panel cuenta con otros elementos a parte de las células solares, que hacen posible la adecuada protección del conjunto frente

a los agentes externos; asegurando una rigidez suficiente, posibilitando la sujeción a las estructuras que lo soportan y permitiendo la conexión eléctrica.

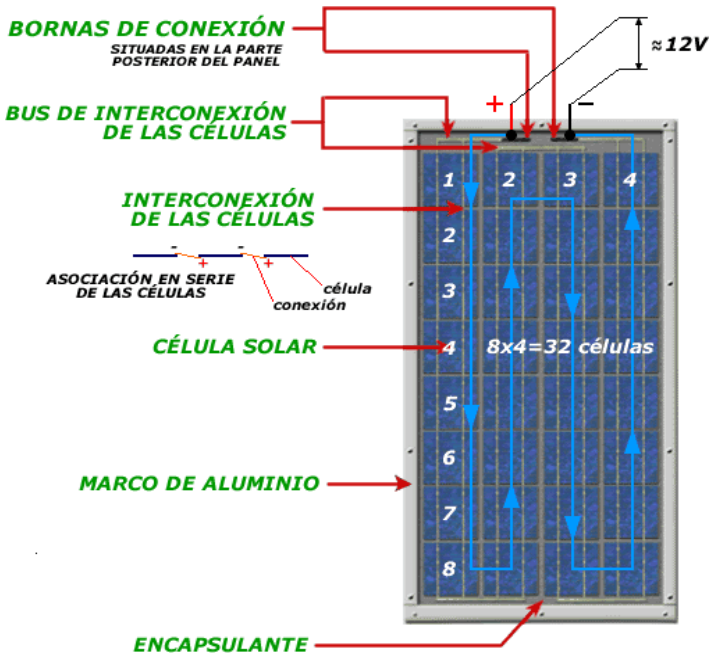


Fig. A.5 Estructura del Panel Solar Fotovoltaico.

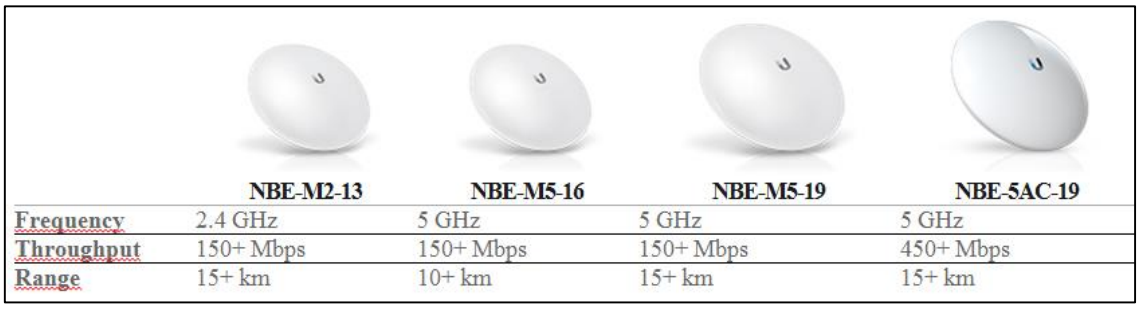
Los elementos de un panel solar fotovoltaico son los siguientes:

- **Cubierta exterior de cara al Sol.** Es de vidrio que debe facilitar al máximo la transmisión de la radiación solar. Se caracteriza por su resistencia mecánica, alta transitividad y bajo contenido en hierro.
- **Encapsulante**. De silicona o más frecuentemente EVA (etilen-vinil-acetato). Es especialmente importante que no quede afectado en su transparencia por la continua exposición al sol, buscándose además un índice de refracción similar al del vidrio protector para no alterar las condiciones de la radiación incidente.
- **Protección posterior**. Igualmente debe dar rigidez y una gran protección frente a los agentes atmosféricos. Usualmente se emplean láminas formadas por distintas capas de materiales, de diferentes características.
- **Marco metálico**. De Aluminio, que asegura una suficiente rigidez y estanqueidad al conjunto, incorporando los elementos de sujeción a la estructura exterior del panel. La unión entre el marco metálico y los elementos que forman el módulo está realizada mediante distintos tipos de sistemas resistentes a las condiciones de trabajo del panel.
- **Cableado y borneas de conexión**. Habituales en las instalaciones eléctricas, protegidos de la intemperie por medio de cajas estancas.

- **Diodo de protección**. Su misión es proteger contra sobre-cargas u otras alteraciones de las condiciones de funcionamiento del panel.

Por lo general, los Panel solares tienen entre 28 y 40 células, aunque lo más típico es que cuenten con 36. La superficie del panel o modulo puede variar entre 0.1 y 0.5m2 y presenta dos bornes de salida, positiva y negativa, a veces tienen alguna intermedia para colocar los diodos de protección. Normalmente, los paneles utilizados, están diseñados para trabajar en combinación con baterías de tensiones múltiplo de 12V, como se detalla en la sección dedicada al acumulador.

Análisis y Diseño de Cobertura de Radio Enlace de la Estación Meteorológica de la Laguna de Mancapozo: El equipo NanoBeam M5 Ubiquiti, está disponible para bandas de frecuencia de 2.4 GHz y 5 GHz, utiliza la tecnología airMax que implementa un acceso múltiple por división de tiempo (TDMA), que permite que cada cliente pueda enviar y recibir datos usando slots de tiempo pre determinado mediante un controlador de punto de acceso (AP). Asimismo posee inteligencia de calidad de servicio QOS, dando prioridad a la voz y video. En la **Fig. A.6**, se detallan las diversas familias NanoBeam, con la finalidad de seleccionar el modelo que cubra con los requerimientos mínimos de distancia y ancho de banda.



	NBE-M2-13	NBE-M5-16	NBE-M5-19	NBE-5AC-19
Frequency	2.4 GHz	5 GHz	5 GHz	5 GHz
Throughput	150+ Mbps	150+ Mbps	150+ Mbps	450+ Mbps
Range	15+ km	10+ km	15+ km	15+ km

Fig. A.6 Cobertura de Radio Enlace de la familia NanoBeam.

En base al cálculo realizado de radio enlace, se requiere contar con un modelo que tenga un alcance con línea de vista superior a los 11.412 Km, por lo cual se decidió utilizar el modelo M5-19, que tiene un alcance superior a los 15 Km, con una tasa de transmisión de 150 Mbps y con una frecuencia de radio enlace en la banda libre de 5 GHz. Esto implica que se pueden transmitir señales de datos, voz y video. Asimismo, se verificó la cobertura con el software de Google Earth y Global Mapper, conjuntamente con la aplicación de Radio Mobile. En las gráficas de la **Fig. A.7, Fig. A.8, Fig. A.9, Fig. A.10,** y **Fig. A.11,** se muestran detalles del análisis y diseño del radio enlace.

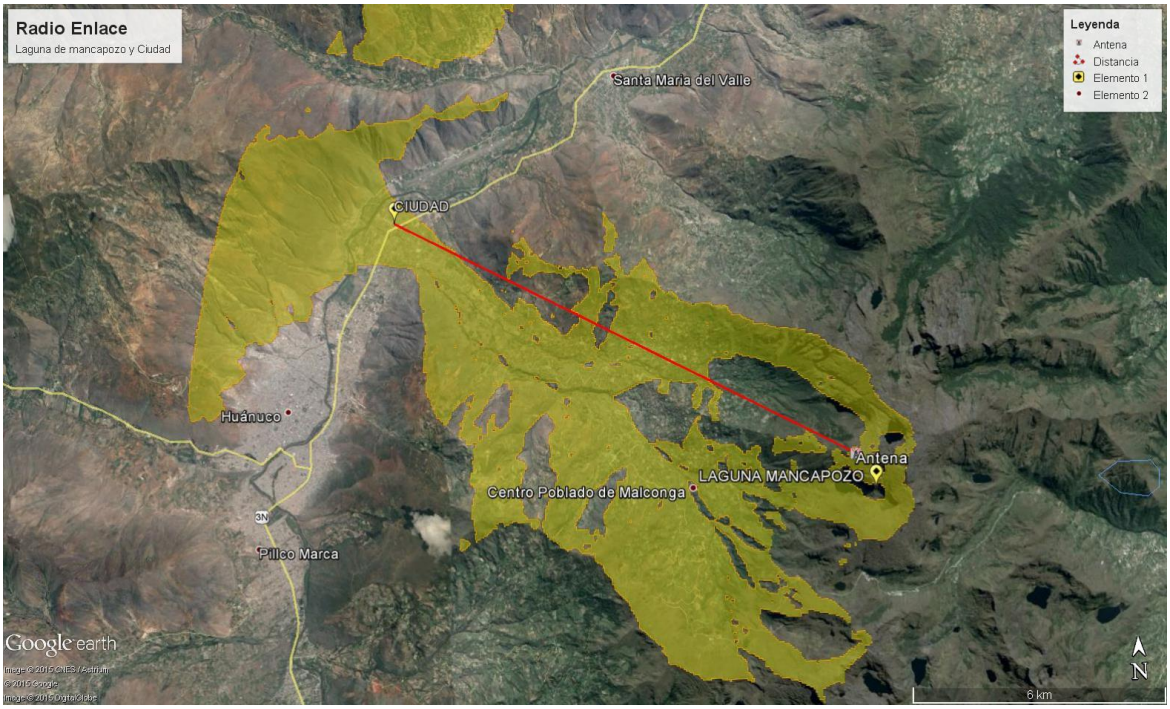


Fig. A.7 Áreas de Cobertura del Radio Enlace de la Estación Meteorológica.

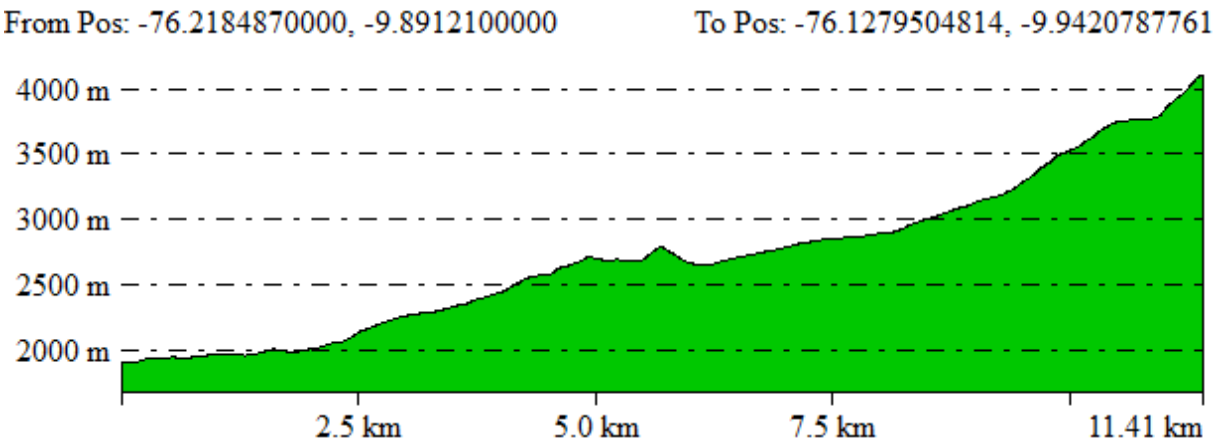


Fig. A.8 Análisis de la línea de vista del radio enlace punto a punto.

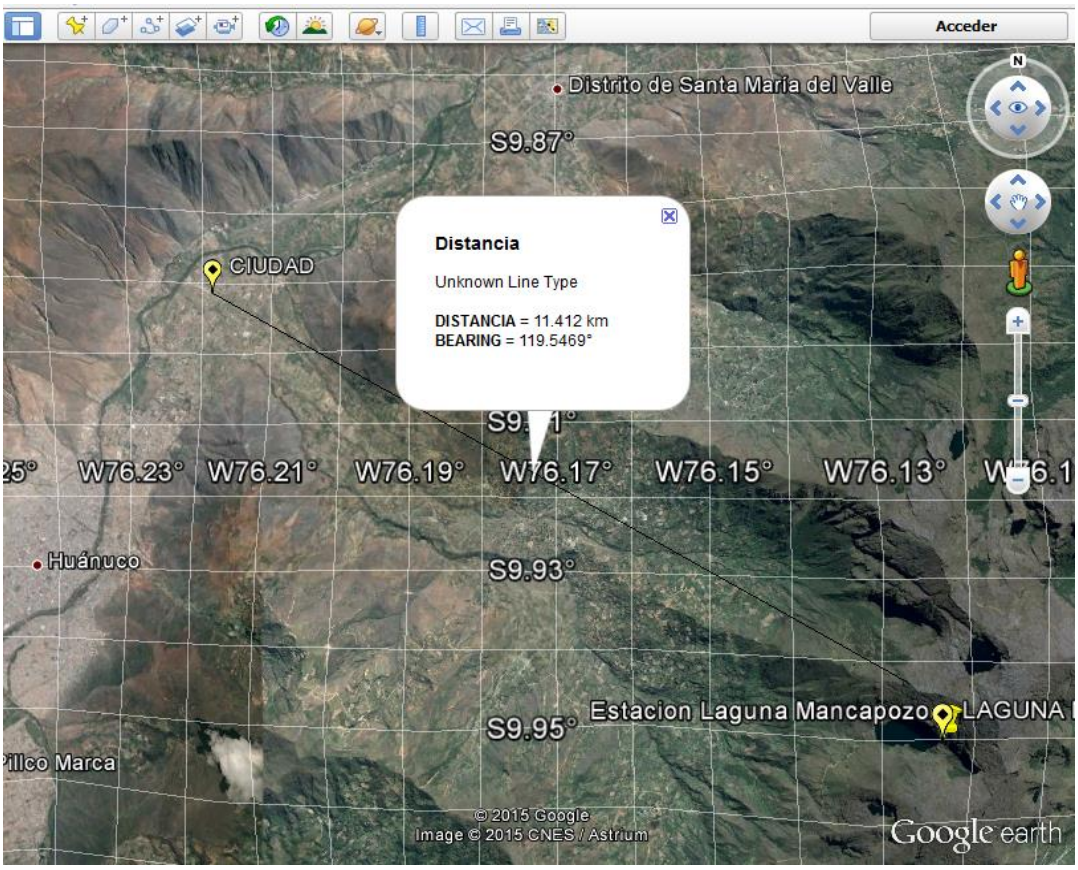


Fig. A.9 Geo referencia del Radio Enlace en Google Earth.

.

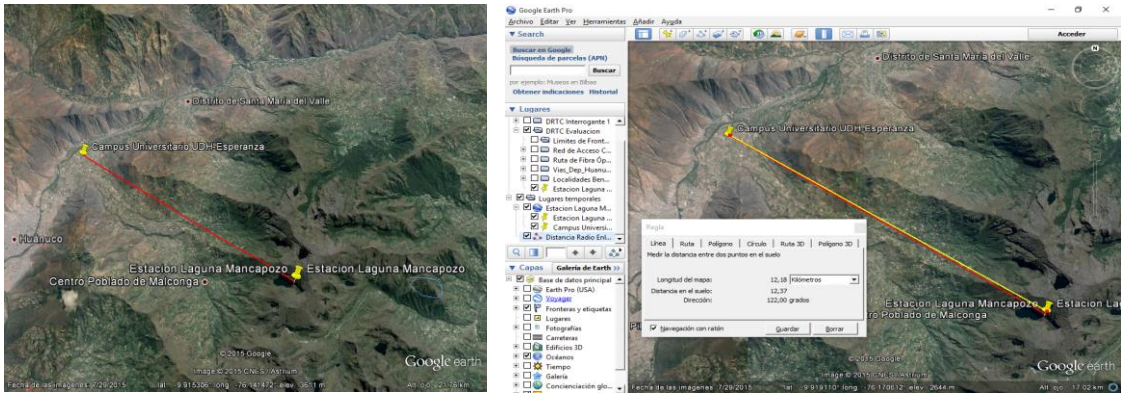


Fig. A.10 Cálculo de la distancia del Radio Enlace.

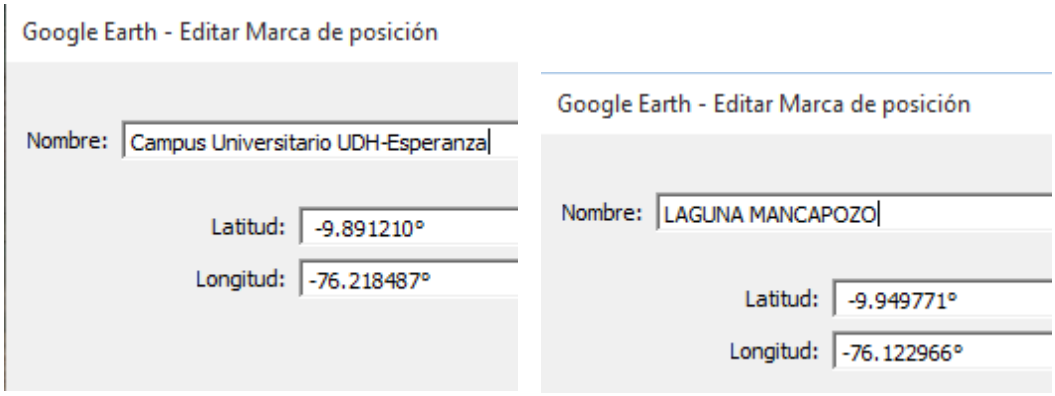


Fig. A.11 Coordenadas geo referenciales de los nodos de Radio Enlace.

Los estándares tecnológicos para la transmisión de datos, está supeditada a la capacidad del ancho de banda y el medio de transmisión. En la Tesis se ha utilizado un Radio Enlace punto a punto, con la tecnología NanoBeam modelo NBE-MS-19, que soporta los estándares: 802.11a, 802.11n/a/, que permite tasas de transmisión de datos de hasta 54 Mbps, en caso se requiera procesar señales de video o fotográficas.

Gestión de Riesgos de la Estación Meteorológica: A fin de establecer mecanismos de contingencia ante riesgos naturales o fallas en los equipos de radio enlace, se plantea el siguiente esquema:

- **Fallas Energéticas:** Se ha previsto el uso de un Panel solar para captar la energía durante el día y de una batería para ser utilizado durante la noche; este sistema es dual auto configurable, que permite cargar la batería durante el día para su uso en la noche. Asimismo, se utiliza un inversor de corriente DC a AC 220Vac para energizar la tarjeta RaspberryPI, quien mediante el puerto USB-Serial energizará al Arduino.

- **Fallas en el Radio Enlace:** Se ha previsto el uso de una base de datos en la estación conjuntamente con un servidor web incorporado en el RaspberryPI, de modo tal que si falla el radio enlace, los datos recolectados serán temporalmente almacenados para su posterior transferencia al Servidor Remoto, cuando se restablezca el radio enlace. La ventaja de utilizar RaspberryPI es que posee un sistema operativo Linux, con arquitectura de sistema operativo monolítico, adecuado para estas aplicaciones embedidas. El uso del Cron en Linux permitirá realizar copias de respaldo en el servidor local y transferirlo al servidor remoto, en forma periódica, así como establecer alertas de espacio en disco y uso de CPU.

A.3 Descripción de los Sensores de la Estación Meteorológica

En este apartado vamos a describir las características de los sensores utilizados en el prototipo de la estación meteorológica.

Sensor de Temperatura: DS18D20

El termómetro digital DS18B20 del fabricante DALLAS SEMICONDUCTOR provee lecturas de temperatura desde 9 a 12-bits (configurable) los cuales indican la temperatura del dispositivo. La información es enviada desde el DS18B20 sobre una interface de un solo cable 1-Wire, de modo que solamente se requiere un cable y tierra para ser conectado con el micro controlador del Arduino. La potencia para leer, escribir y realizar conversiones de temperatura puede ser derivada desde la misma línea de datos sin necesidad de insertar una fuente de datos externa. Cada DS18B20 contiene un único número de serie, de modo tal que pueden coexistir en el mismo bus del cable 1-Wire. Esto permite poner varios sensores en diferentes lugares. Entre las aplicaciones más comunes figuran: mediciones de temperatura dentro de edificaciones, equipos o máquinas, monitoreo y control de entornos, etc. A continuación, se detalla las características más importantes:

- Requiere un solo bus de cable 1-Wire.
- Posee capacidad de múltiple conexión para simplificar las aplicaciones de mediciones de temperaturas distribuidas.
- No requiere componentes externos.
- Puede ser energizado desde la misma línea de datos.
- El rango de potencia se encuentra entre 3.0V a 5.5V.
- Requiere que la fuente de energía repose en cero.
- Tiene un rango de medición desde -55°C hasta +125°C. Cuyo equivale en Fahrenheit es de -67°F hasta +257°F.
- Posee una precisión de ±0.5°C desde -10°C hasta +85°C.
- Posee una resolución programable desde 9 hasta 12 bits
- Permite una conversión digital de temperatura de 12 bits en 750 ms.
- Permite configurar alarmas de temperatura no volátil.
- El comando de búsqueda de alarmas, identifica y direcciona los dispositivos cuyas temperaturas están fuera de los limites programados (condición de alarma de temperatura).
- Las aplicaciones incluyen controles termostáticos, sistemas industriales, productos de consumo, termómetros, o cualquier sistema sensitivo de temperatura.

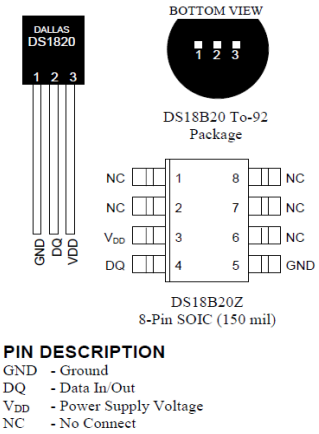


Fig. A.12 Pines del Termómetro Digital DS18B20.

Sensor de Humedad: HIH-4000-001

El sensor de humedad de la serie HIH-4000 del fabricante Honeywell, está diseñado especialmente para grandes volúmenes. Permite una entrada directa al micro controlador u otro dispositivo, mediante la salida del voltaje lineal del sensor. Con una corriente típica de drenaje de solamente 200 micro amperios, la serie HIH-4000 es ideal para sistemas operados por batería con un bajo drenaje. Los datos de calibración del sensor están disponibles.

La serie HIH-4000 entrega buena calidad en la adquisición de la humedad relativa (HR) a bajo costo, con tecnología de soldado SIP (Single In-line Package). Está disponible en dos modos de configuración. Su uso incluye aplicaciones de meteorología, secado, limpieza, petróleo y diversos entornos químicos. Este pequeño sensor proporciona una tensión analógica de salida que se puede conectar directamente con la entrada al convertidor ADC de cualquier controlador. Esta salida lineal se incrementa en 30.680mV por cada %RH, a partir de un offset de 0.958V (a 0% de RH la salida es de 0.958V). Entre sus características podemos citar las siguientes: Salida lineal de tensión respecto al %RH, Bajo consumo, Alta resolución, Tiempo rápido de respuesta.

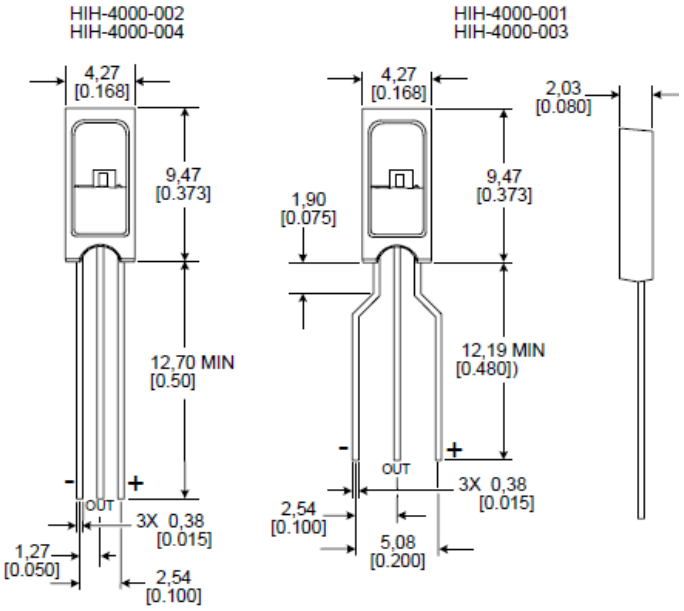


Fig. A.13 Pines del Sensor de Humedad HIH4000-001.

En la **Fig. A.14** se muestra el detalle de las características del sensor de humedad dados por el fabricante Honeywell.

Parameter	Minimum	Typical	Maximum	Unit
Interchangeability (best fit straight line)	–	–	–	–
0 % to 60 %	-5	–	5	%RH
60 % to 100 %	-8	–	8	%RH
Interchangeability (2nd order curve)	–	±3.5	–	%RH
Accuracy[1] (best fit straight line)	–	±3.5	–	%RH
Accuracy (2nd order curve)	–	±2.5	–	%RH
Hysterisis	–	3	–	%RH
Repeatability	–	±0.5	–	%RH
Settling time	–	–	70	ms
Response time (1/e in slow moving air)	–	15	–	s
Stability[2] (@ 50 %RH)	–	±1.2 (per year)	–	%RH
Stability[3] (@ 50 %RH)	–	±0.5 (per year)	–	%RH
Voltage supply	4	–	5.8	Vdc
Current supply	–	–	500	µA
Voltage output (1st order fit)	$V_{out}=V_{supply}(0.0062(\text{sensor RH})+0.16)$			
Voltage output (2nd order curve fit)	$V_{out}=0.00003(\text{sensor RH})^2+0.0281(\text{sensor RH})+0.820$, typical @ 25 ºC			
Temperature compensation	$V_{out}=(0.0305+0.000044T-0.0000011T^2)(\text{Sensor RH})+(0.9237-0.0041T+0.000040T^2)$, T=Temperature in ºC			
Operating temperature	-40[-40]	See Figure 1.	85[185]	ºC[ºF]
Operating humidity	0	See Figure 2.	100	%RH
Storage temperature	-40[-40]	–	125[257]	ºC[ºF]
Storage humidity	See Figure 2.			%RH

Fig. A.14 Características del Sensor de Humedad HIH4000.

Sensor de Presión: MPX4250AP

El sensor de presión de la serie MPX4250 del fabricante Freescale Semiconductor, está diseñado para adquirir la presión absoluta del aire. La serie de los transductores piezo resistivos MPX42500A, es un sensor de presión de silicio monolítico diseñado para una

amplia gama de aplicaciones, particularmente aquellos que emplean micro controladores o micro procesadores con entradas A/D. Este transductor combina técnicas avanzadas de micro maquinas, metalización de láminas delgadas y procesamiento bipolar para proporcionar una salida de señal analógica proporcional a la presión aplicada. Las principales características del sensor son las siguientes:

- Error máximo de 1.5% en el intervalo 0° hasta 85°C.
- Especialmente diseñado para sistemas de control de motores para medir la presión absoluta.
- Posee compensación de temperatura entre -40° hasta +125°C.
- Ofrece una reducción en peso y volumen para módulos híbridos existentes.
- Posee un empaquetado exterior duradero, ideal para aplicaciones no automovilísticas.

Characteristic		Symbol	Min	Typ	Max	Units
Differential Pressure Range(1)		P_{OP}	20	—	250	kPa
Supply Voltage(2)		V_S	4.85	5.1	5.35	V_{DC}
Supply Current		I_O	—	7.0	10	mAdc
Minimum Pressure Offset(3) @ V_S = 5.1 Volts	(0 to 85°C)	V_{OFF}	0.133	0.204	0.274	V_{DC}
Full Scale Output(4) @ V_S = 5.1 Volts	(0 to 85°C)	V_{FSO}	4.826	4.896	4.966	V_{DC}
Full Scale Span(5) @ V_S = 5.1 Volts	(0 to 85°C)	V_{FSS}	—	4.692	—	V_{DC}
Accuracy(6)	(0 to 85°C)	—	—	—	±1.5	$\%V_{FSS}$
Sensitivity		$\Delta V/\Delta P$	—	20	—	mV/kPa
Response Time(7)		t_R	—	1.0	—	msec
Output Source Current at Full Scale Output		I_O+	—	0.1	—	mAdc
Warm-Up Time(8)		—	—	20	—	msec
Offset Stability(9)		—	—	±0.5	—	$\%V_{FSS}$

Fig. A.15 Características del Sensor de Presión MPX4250A.

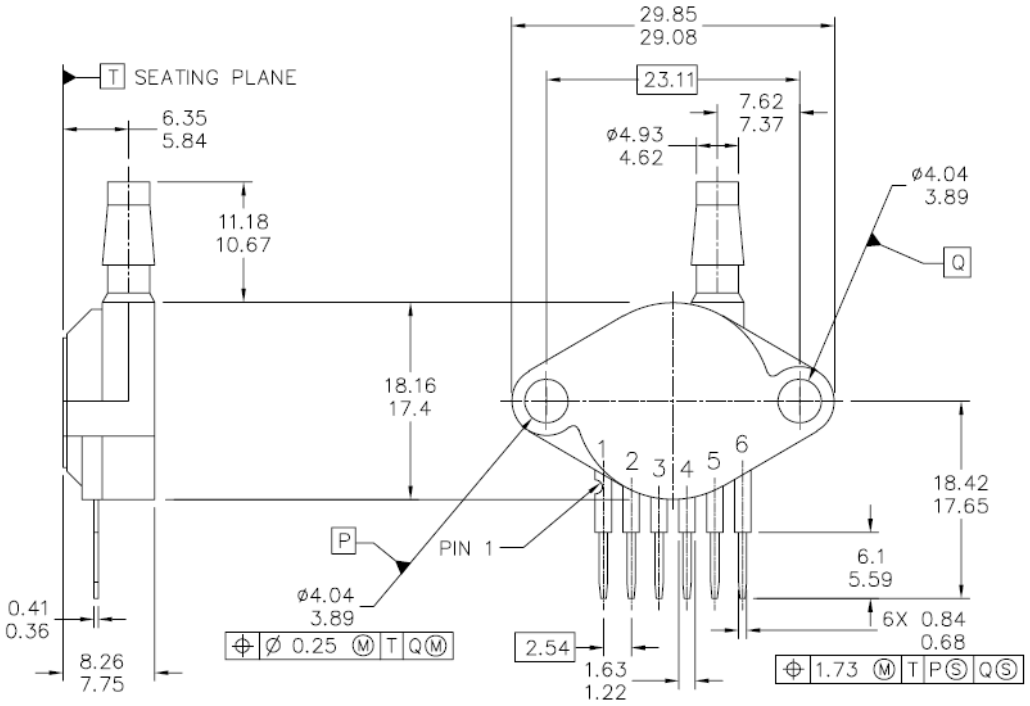
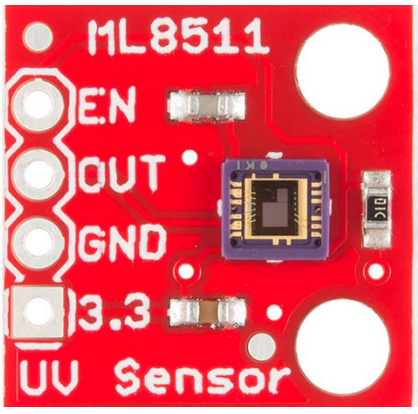
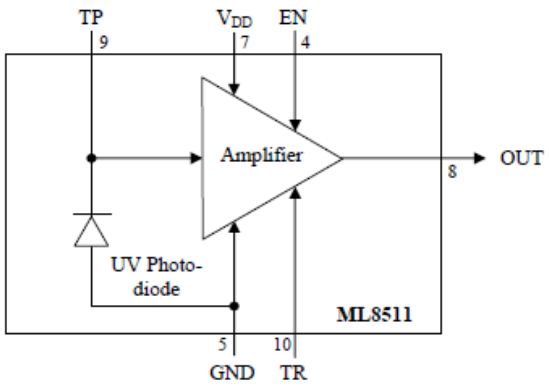


Fig. A.16 Dimensiones del Sensor de Presión MPX4250AP.

Sensor de Ultravioleta ML8511

El Sensor de Ultra Violeta ML8511, tiene como salida un pin de voltaje digital, que mide la intensidad de UV en mW/cm^2, internamente utiliza un amplificador operacional el cual convierte mediante un transductor de corriente a voltaje dependiendo de la intensidad de radiación ultravioleta. Se alimenta con una corriente de 0.1 uA, posee un rango de trabajo desde -20 °C hasta los 70 °C.





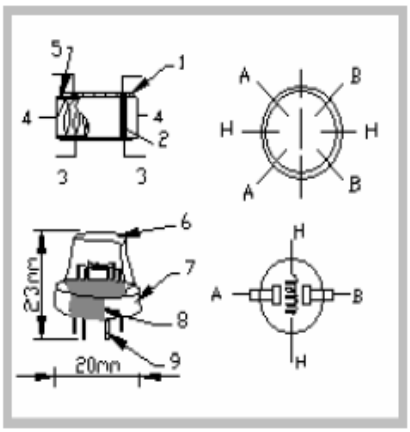
Pin	Symbol	I/O	Function
7	VDD	PW	Supply voltage. Decouple this pin to ground with 0.1 µF capacitor.
5	GND	PW	Ground
4	EN	I	Active high enable pin. (High: Active mode, Low: Standby mode)
8	OUT	O	Output (Low in power down or standby mode)
9	TP	I/O	Test pin. Do not connect.
10	TR	I/O	Internal reference voltage. Decouple this pin to ground with 1 nF capacitor.
1,2,3, 6,11,12	NC	-	No Connection. Do not connect.

Fig. A.17 Pines del Sensor Ultravioleta ML8511.

Sensor de Monóxido de Carbono

Este sensor detecta las concentraciones de CO en el aire y genera una salida en voltaje analógico. El sensor puede medir concentraciones de 10 a 10,000 ppm. Puede operar a temperaturas desde -10 °C hasta los 50°C, y consume menos de 150 mA, 5V.

Model No.			MQ-7
Sensor Type			Semiconductor
Standard Encapsulation			Plastic
Detection Gas			Carbon Monoxide
Concentration			10-10000ppm CO
Circuit	Loop Voltage	V_c	≤10V DC
	Heater Voltage	V_H	5.0V±0.2V ACorDC（High） 1.5V±0.1V ACorDC（Low）
	Heater Time	T_L	60±1S（High）90±1S（Low）
	Load Resistance	R_L	Adjustable
Character	Heater Resistance	R_H	31Ω±3Ω（Room Tem.）
	Heater consumption	P_H	≤350mW
	Sensing Resistance	R_s	2KΩ-20KΩ(in 100ppm CO)
	Sensitivity	S	Rs(in air)/Rs(100ppm CO)≥5
	Slope	α	≤0.6 (R_{300ppm}/R_{100ppm} CO)
Condition	Tem. Humidity		20℃±2℃；65%±5%RH
	Standard test circuit		Vc:5.0V±0.1V； V_H（High）：5.0V±0.1V； V_H（Low）：1.5V±0.1V
	Preheat time		Over 48 hours



	Parts	Materials
1	Gas sensing layer	SnO_2
2	Electrode	Au
3	Electrode line	Pt
4	Heater coil	Ni-Cr alloy
5	Tubular ceramic	Al_2O_3
6	Anti-explosion network	Stainless steel gauze (SUS316 100-mesh)
7	Clamp ring	Copper plating Ni
8	Resin base	Bakelite
9	Tube Pin	Copper plating Ni

Fig.1

Fig. A.18 Pines del Sensor de Monóxido de Carbono.

Sensor de Dióxido de Carbono MG-811

El sensor CO2 utiliza una tarjeta MG-811 que posee un circuito de acondicionamiento de señal mediante un amplificador operacional y un circuito de calor para calentar el sensor, posee salida analógica y digital. En la **Fig. A.19** se muestra los detalles de la tarjeta de adquisición MG-811.

MG-811 Specifications

SYMBOL	PARAMETER	VALUE	REMARKS
V_H	Heating Voltage	6.0±0.1V	AC or DC
R_H	Heating Resistor	~30.0 Ohm	At room temperature
I_H	Heating Current	~200mA	
P_H	Heating Power	~1200mW	
Tao	Operating Temperature	-20 – 50°C	
Tas	Storage Temperature	-20 – 70°C	
EMF	Output	100-600mV	400-10000ppm CO2

Pinout

PIN	DESCRIPTION	REMARKS
VCC	5V power supply for signal conditioning	<5.5V
VOUT	Analog voltage signal output	
BOOL	Comparator output	Open drain
HEAT	Heating power supply	6-24V 7.5-12V*
VSET	Heating voltage select	0-5V
GND	Common ground	Onboard heating circuit

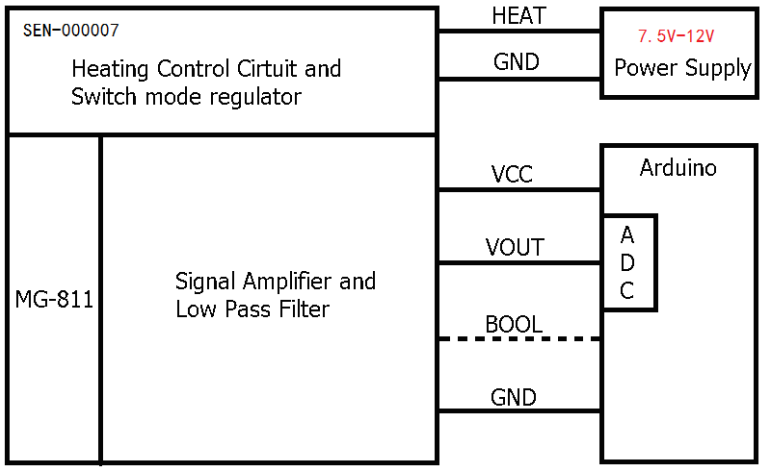
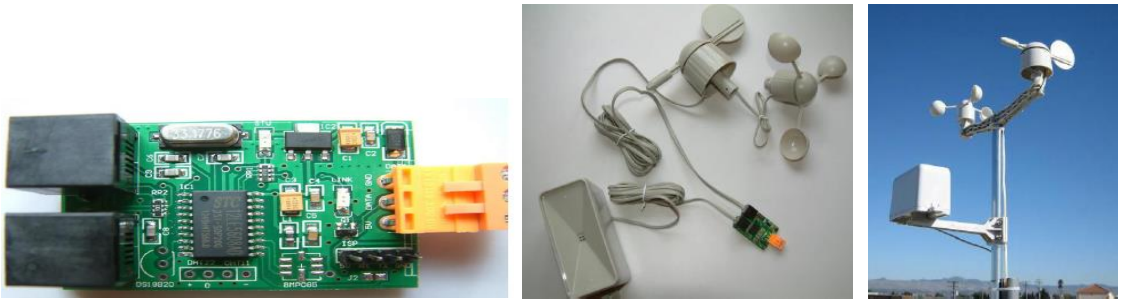
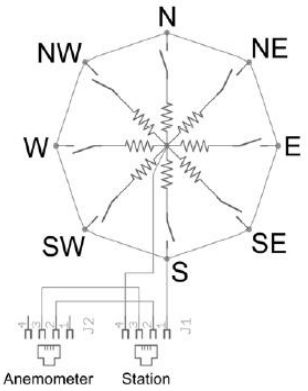


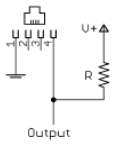
Figura A.19: Pines del Sensor de Dióxido de Carbono MG-811.

Sensor de Viento y Lluvia BH4TDV

El sensor BH4TDV, permite capturar la intensidad de lluvia, velocidad y posición de viento vía dos conectores RJ11. Opera con voltajes de entre 3.3 V hasta los 16 V, incorpora circuitería para el acondicionamiento de la señal.







Example wind vane interface circuit. Voltage readings for a 5 volt supply and a resistor value of 10k ohms are given in the table.

Direction (Degrees)	Resistance (Ohms)	Voltage (V=5v, R=10k)
0	33k	3.84v
22.5	6.57k	1.98v
45	8.2k	2.25v
67.5	891	0.41v
90	1k	0.45v
112.5	688	0.32v
135	2.2k	0.90v
157.5	1.41k	0.62v
180	3.9k	1.40v
202.5	3.14k	1.19v
225	16k	3.08v
247.5	14.12k	2.93v
270	120k	4.62v
292.5	42.12k	4.04v
315	64.9k	4.78v
337.5	21.88k	3.43v

Fig. A.20 Detalle del Sensor de Dirección de Viento y Lluvia.

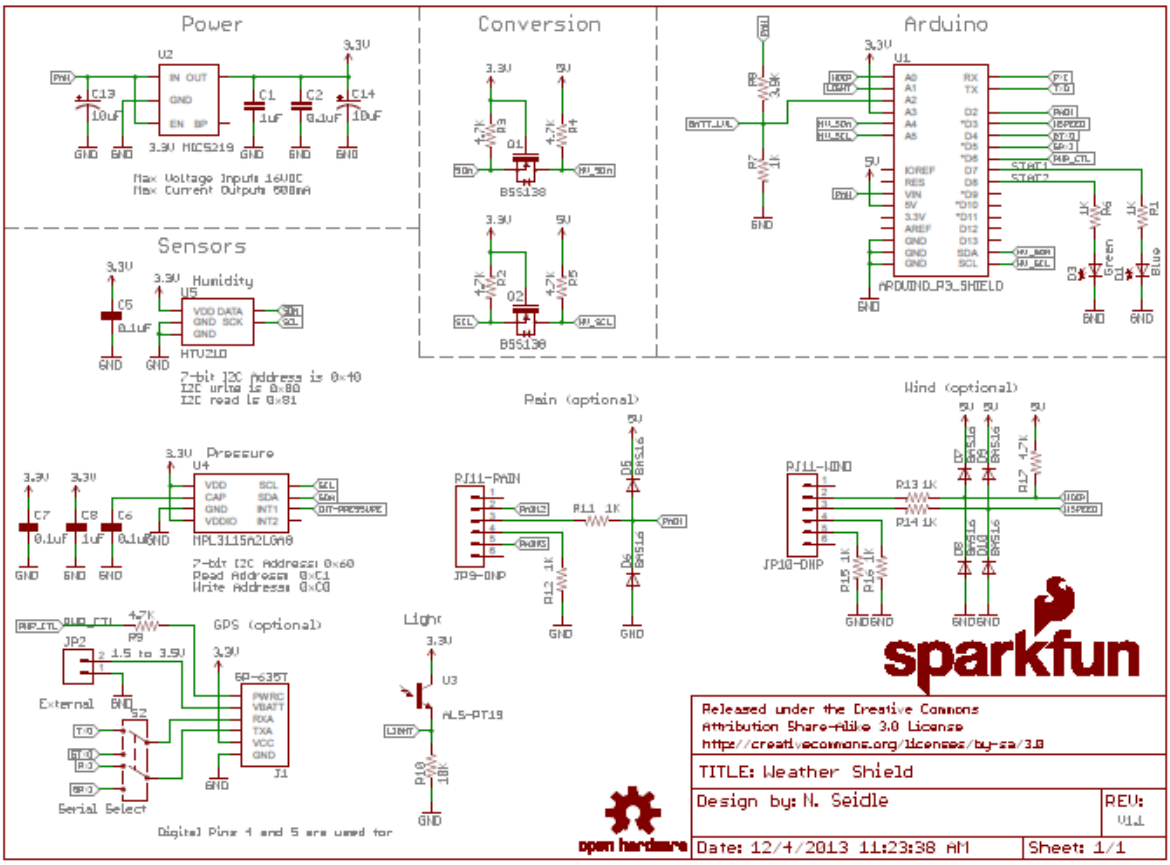


Fig. A.21 Pines del Sensor de Dirección y Velocidad del Viento e Intensidad de Lluvia.

Sensor de Cámara

El sensor RaspberryPi Camera, permite obtener fotos y video con una resolución de 5 Mega pixeles en formato nativo, con soporte para video de 1080p30, 720p60, y 640x480p60/90, dichas imágenes podrían captar los momentos de lluvia para que, por ejemplo, posteriormente pueda utilizarse procesamiento digital de imágenes para calcular la intensidad de lluvia mediante el diámetro de las gotas de lluvia.

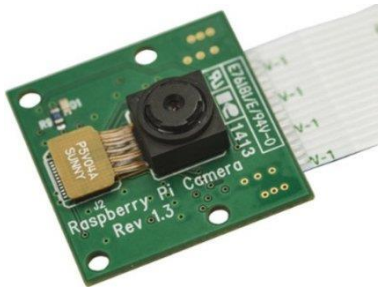


Fig. A.22 Pines del Sensor de Cámara de Video.

Sensor Bluetooth

El sensor Chip Bluetooth trabaja con un voltaje de 3.3 V, con una velocidad de transmisión de 1200, 2400, 4800, 9600, 19200, 38400, 57600 y 115200 baudios, por defecto viene configurado a 9600 baudios. Posee una corriente de trabajo de 8 mA, responde a comandos AT. Código de Pin 1234, Baud Rate: 9600, N, 8, 1. Permite enviar comandos a la Estación Meteorológica para tareas de calibración, operación y mantenimiento.

 **Función**	**Comando**	**Respuesta**
Test de Comunicación	AT	OK
Cambiar la Velocidad de Transmisión (ver siguiente tabla)	AT + BAUD**x**	OK**vvvv**
Cambiar Nombre del Dispositivo	AT + NAME**name**	OK**name**
Cambiar el Código PIN	AT + PIN**xxxx**	OKsetpin

Fig. A.23 Pines del Sensor Bluetooth.

A.4 Estándares y Normativas de una Estación Meteorológica

En esta sección se presentan los principios y cuestiones de la normalización internacional, tendencias y perspectivas de evolución de la Meteorología, en particular de las Estaciones Meteorológicas. Los sistemas de información ambiental (EIS por sus siglas en inglés), permiten llevar a cabo estudios que abarcan periodos temporales grandes y espacios geográficos extensos, así como usar datos para propósitos diferentes a los originales. La Organización Meteorológica Mundial (OMM) es un organismo especializado de las Naciones Unidas. Es el portavoz acerca del estado y el comportamiento de la atmósfera terrestre, su interacción con los océanos, el clima que produce y la distribución resultante de los recursos hídricos. La OMM cuenta con 191 Estados y Territorios Miembros (desde el 1 de enero de 2013). Su predecesora, la Organización Meteorológica Internacional (OMI), se fundó en 1873. La OMM se creó en 1950 y se convirtió en el organismo especializado de las Naciones Unidas para la meteorología (tiempo y clima), la hidrología operativa y las ciencias geofísicas conexas en 1951. Como el tiempo, el clima y el ciclo del agua no conocen fronteras nacionales, la cooperación internacional a escala mundial es esencial para el desarrollo de la meteorología y la hidrología operativa, así como para recoger los beneficios derivados de su aplicación. La OMM proporciona el marco en el que se desarrolla esta cooperación internacional. La OMM promueve la aplicación de la información meteorológica, climatológica, hidrológica y oceanográfica a las actividades humanas. La OMM ayuda a sus Miembros a prestar servicios meteorológicos eficaces y fiables que contribuyan a la seguridad de la vida y a la protección de los bienes, así como al bienestar y a la comodidad general de la población mundial. La OMM garantiza la prestación de servicios meteorológicos rentables y flexibles en todo el mundo que avalan la seguridad, regularidad y eficacia de las operaciones aeronáuticas. Es importante recalcar el aplicativo METEOTERM, que es una base de datos terminológica de la OMM, que contiene terminología especializada en español y otros idiomas, que incluye un vocabulario meteorológico internacional.

Las señales Meteorológicas que fueron utilizadas en la validación de los algoritmos, provienen de la base de datos del SENAMHI sede Huánuco, ello debido a que las señales meteorológicas tomadas en la laguna de Mancapozo ubicado en el Centro Poblado de Malconga, del distrito de Amarilis de la Región Huánuco, aún está en proceso de evaluación y se requiere una adquisición permanente de varias semanas, meses e inclusive años para implementar un sistema de pronóstico adecuado y con cierta exactitud, sin embargo las señales meteorológicas utilizadas provienen de la zona de estudio y permitió tener un alcance aproximado en las validaciones en la eventual

implementación comercial del prototipo propuesto. Por otra parte, en esta sección se van a construir y verificar los códigos fuente de cada una de las partes que componen los módulos descritos en el capítulo III. En la **Tabla N° A.1** se detalla la clasificación de las Estaciones Meteorológicas, establecido en base a la Organización Meteorológica Mundial (OMM), según su finalidad y el tipo de clasificación establecido a nivel internacional (ver [6], [7], [8]). Para mayor detalle, remitirse al Anexo C.4 y C.5, en donde se desarrollan los aspectos normativos y los estándares de las estaciones meteorológicas, así como los detalles de comunicación punto a punto vía radio enlace.

Tabla N° A.1 Tipos de Estaciones Meteorológicas.

Clasificación de las Estaciones Meteorológicas según la Organización Mundial de Meteorología (OMM)	
Según su Finalidad	**Clasificación**
Sinóptica	Climatológicas
	Agrícolas
	Especiales
	Aeronáuticas
	Satelitales
De acuerdo a la magnitud de las observaciones	Principales
	Ordinarias
	Auxiliares o adicionales
Por el nivel de observación	Superficie
	Altitud
Según el lugar de observación	Terrestre
	Aéreas
	Marítimas

Asimismo es importante tener en cuenta que la OMM ha publicado diversos documentos, tal como: Manual de los Sistemas de Información de la Organización Meteorológica Mundial, actualizado al año 2013 de 83 páginas, en donde se encuentran todos los detalles técnicos de la estandarización de las Estaciones Meteorológicas. A continuación, se lista un compendio de normas e instrumentos jurídicos en materia de fiscalización ambiental vigente en la normativa Peruana a través del Ministerio del Ambiente, a decir:

LEYES

- Ley Nº 28804 (21.07.2006): Ley que regula la Declaratoria de Emergencia Ambiental.
- Ley Nº 29325 (05.03.2009): Ley del Sistema Nacional de Evaluación y Fiscalización

Ambiental.

- Ley Nº 29662 (09.02.2011): Ley que prohíbe el asbesto anfíboles y regula el uso del asbesto crisólito.

DECRETOS LEGISLATIVOS

- Decreto Legislativo Nº 1013 (14.05.2008): Decreto Legislativo que Aprueba la Ley de Creación, Organización y Funciones del Ministerio del Ambiente.
- Decreto Legislativo Nº 1055 (27.06.2008): Decreto Legislativo que modifica la Ley Nº 28611 - Ley General del Ambiente.

DECRETO DE URGENCIA

- Decreto de Urgencia Nº 023-2008 (19.06.2008): Dictan medidas urgentes en materia económica y financiera a favor del Ministerio del Ambiente.

DECRETOS SUPREMOS

- Decreto Supremo Nº 059-2005-EM (08.12.2005): Aprueban Reglamento de Pasivos Ambientales de la Actividad Minera.
- Decreto Supremo Nº 009-2006-AG (24.02.2006): Reconocen derechos de posesión, uso y usufructo, ancestrales y tradicionales de pueblos originarios vinculados al aprovechamiento sostenible de la totora, los llachos y recursos naturales en los sectores Puno, Ramis y Lago Titicaca.
- Decreto Supremo Nº 043-2006-EM (28.07.2006): Establecen Disposiciones Generales para la Aplicación del Silencio Administrativo Negativo en los procedimientos administrativos tramitados ante la Dirección General de Asuntos Ambientales Energéticos.
- Decreto Supremo Nº 055-2006-AG (01.09.2006): Disponen la categorización de la Zona Reservada Los Pantanos de Villa.
- Decreto Supremo Nº 033-2007-PCM (05.04.2007): Aprueban el Procedimiento para la aprobación de los Estándares de Calidad Ambiental (ECA) y Límites Máximos Permisibles (LMP) de Contaminación Ambiental.
- Decreto Supremo Nº 020-2007-PRODUCE (28.10.2007): Establecen Plan Ambiental Complementario Pesquero (PACPE) en la Bahía El Ferrol.
- Decreto Supremo Nº 039-2007-MTC (13.11.2007): Aprueban Reglamento de la Ley Nº 29022 - Ley para la Expansión de Infraestructura en Telecomunicaciones.
- Decreto Supremo Nº 010-2008-PRODUCE (30.04.2008): Límites Máximos Permisibles (LMP) para la Industria de Harina y Aceite de Pescado y Normas

Complementarias.

- Decreto Supremo Nº 037-2008-PCM (14.05.2008): Establecen Límites Máximos Permisibles de Efluentes Líquidos para el Subsector Hidrocarburos.
- Decreto Supremo Nº 028-2008-EM (27.05.2008): Aprueban el Reglamento de Participación ciudadana en el Subsector Minero.
- Decreto Supremo Nº 007-2008-MINAM (06.12.2008): Aprueban Reglamento de Organización y Funciones del Ministerio del Ambiente.
- Decreto Supremo Nº 002-2009-MINAM (17.01.2009): Decreto Supremo que aprueba el Reglamento sobre Transparencia, Acceso a la Información Pública Ambiental y Participación y Consulta Ciudadana en Asuntos Ambientales.
- Decreto Supremo Nº 008-2009-MINAM (24.04.2009): Establecen disposiciones para la elaboración de los Planes Maestros de las Áreas Naturales Protegidas.
- Decreto Supremo Nº 012-2009-MINAM (23.05.2009): Aprueba la Política Nacional del Ambiente.
- Decreto Supremo Nº 018-2009-MINAM (08.09.2009): Aprueban Reglamento de Uso Turístico en Áreas Naturales Protegidas.
- Decreto Supremo Nº 078-2009-EM (08.11.2009): Implementan medidas de remediación ambiental a cargo del titular minero que haya realizado actividades y/o ejecutado proyectos relacionados con actividades mineras previstas en la Ley General de Minería.
- Decreto Supremo Nº 022-2009-MINAM (15.12.2009): Aprueban Reglamento de Organización y Funciones del Organismo de Evaluación y Fiscalización Ambiental – OEFA.
- Decreto Supremo Nº 003-2010-MINAM (17.03.2010): Aprueba Límites Máximos Permisibles para los efluentes de Plantas de Tratamiento de Aguas Residuales Domésticas o Municipales.
- Decreto Supremo Nº 007-2010-MINAM (13.07.2010): Aprueban el Texto Único de Procedimientos Administrativos - TUPA del Ministerio del Ambiente.
- Decreto Supremo Nº 014-2010-MINAM (07.10.2010): Aprueban los Límites Máximos Permisibles para las Emisiones Gaseosas y de Partículas de las Actividades del Sub Sector Hidrocarburos.
- Decreto Supremo Nº 025-2011-EF (16.02.2011): Aprueban monto por concepto de Dietas para los Miembros del Consejo Directivo del Organismo de Evaluación y Fiscalización Ambiental – OEFA.
- Decreto Supremo Nº 004-2011-EM (19.02.2011): Aprueban el Reglamento de la Ley

que regula los Pasivos Ambientales del Subsector Hidrocarburos.

- Decreto Supremo Nº 011-2011-MINAM (17.06.2011): Decreto Supremo que autoriza la constitución del Fideicomiso para la administración de recursos recaudados por concepto de multas impuestas por infracciones a normas ambientales.
- Decreto Supremo Nº 015-2011-MINAM (09.07.2011): Aprueban el Reglamento Interno del Tribunal de Solución de Controversias Ambientales.
- Decreto Supremo Nº 015-2012-VIVIENDA (14.09.2012): Aprueban Reglamento de Protección Ambiental para proyectos vinculados a las actividades de Vivienda, Urbanismo, Construcción y Saneamiento.
- Decreto Supremo Nº 008-2012-MINAM (14.11.2012): Aprueban Reglamento de la Ley que establece la moratoria al ingreso y producción de organismos vivos modificados al territorio nacional por un período de 10 años.
- Decreto Supremo Nº 018-2012-AG (14.11.2012): Aprueban Reglamento de Participación Ciudadana para la Evaluación, Aprobación y Seguimiento de Instrumentos de Gestión Ambiental del Sector Agrario.
- Decreto Supremo Nº 237-2012-EF (01.12.2012): Aprueban Escala Remunerativa del Organismo de Evaluación y Fiscalización Ambiental OEFA.
- Decreto Supremo Nº 271-2012-EF (21.12.2012): Aprueban monto por concepto de dietas para los vocales del Tribunal de Fiscalización Ambiental (TFA) del Organismo de Evaluación y Fiscalización Ambiental – OEFA.
- Decreto Supremo Nº 006-2013-MINAM (19.06.2013): Aprueban Disposiciones Complementarias para la aplicación de Estándar de Calidad Ambiental (ECA) de Aire
- Decreto Supremo Nº 008-2013-MINAM (22.08.2013): Decreto Supremo que aprueba disposiciones reglamentarias del artículo 20º-A de la Ley Nº 29325 - Ley del Sistema Nacional de Evaluación y Fiscalización Ambiental.
- Decreto Supremo Nº 009-2013-MINAM (04.09.2013): Aprueban Reglamento del numeral 149.1 del artículo 149º de la Ley Nº 28611 – Ley General del Ambiente.

RESOLUCIONES MINISTERIALES

- Resolución Ministerial Nº 490-2006-MEM-DM (15.10.2006): Encargan seguimiento, monitoreo y cumplimiento del Acuerdo suscrito entre los Apus de las Comunidades Indígenas del Río Corrientes, el Ministerio de Energía y Minas, el Ministerio de Salud, el Gobierno Regional de Loreto y la Empresa Pluspetrol Norte S.A.
- Resolución Ministerial Nº 251-2008-MINSA (08.04.2008): Crean Grupo de trabajo encargado de analizar y proponer las acciones para prevenir, mitigar y garantizar el tratamiento de efectos causados por la contaminación ambiental en la salud de las

personas.

- Resolución Ministerial Nº 304-2008-MEM-DM (26.06.2008): Aprueban Normas que regulan el Proceso de Participación Ciudadana en el Subsector Minero.
- Resolución Ministerial Nº 054-2008-MINAM (04.11.2008): Declaran la desactivación y extinción del CONAM, al haber concluido su fusión con el Ministerio del Ambiente.
- Resolución Ministerial Nº 571-2008-MEM-DM (16.12.2008): Aprueban Lineamientos para la Participación Ciudadana en las Actividades de Hidrocarburos.
- Resolución Ministerial Nº 044-2009-MINAM (19.02.2009): Aprueban Directiva "Procedimiento para la Oficialización de Eventos por el Ministerio del Ambiente".
- Resolución Ministerial Nº 104-2009-MINAM (23.05.2009): Aprueban Directiva "Procedimiento para la Evaluación y Autorización de Proyectos de Emisiones de Gases de Efecto Invernadero (GEI) y Captura de Carbono".
- Resolución Ministerial Nº 026-2010-MINAM (26.02.2010): Aprueban los "Lineamientos de Política para el Ordenamiento Territorial".
- Resolución Ministerial Nº 030-2011-MEM-DM (22.01.2011): Aprueban Términos de Referencia conforme a los cuales se elaborará el Plan de Implementación para el Cumplimiento de los Límites Máximos Permisibles (LMP) para la descarga de efluentes líquidos de Actividades Minero - Metalúrgicas, así como el procedimiento de evaluación de dicho plan.
- Resolución Ministerial Nº 141-2011-MINAM (30.06.2011): Ratifican lineamiento para la aplicación de los Límites Máximos Permisibles.
- Resolución Ministerial Nº 052-2012-MINAM (08.03.2012): Aprueban Directiva para la Concordancia entre el Sistema Nacional de Evaluación de Impacto Ambiental (SEIA) y el Sistema Nacional de Inversión Pública (SNIP).
- Resolución Ministerial Nº 150-2013-MINAM (24.05.2013): Aprueban los Lineamientos para el Proceso de Selección y Designación de los Laboratorios de Detección de Organismos Vivos Modificados.
- Resolución Ministerial Nº 191-2013-MINAM (04.07.2013): Aprueban Lista de Mercancías Restringidas y Lista de Mercancías Restringidas sujetas a Control y Muestreo en los Puntos de Ingreso, en el marco de la Ley Nº 29811 y su Reglamento.
- Resolución Ministerial Nº 247-2013-MINAM (28.08.2013): Aprueban Régimen Común de Fiscalización Ambiental.

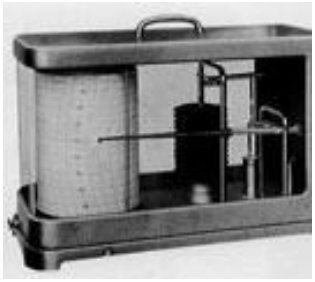
Características de los Sensores Meteorológicos

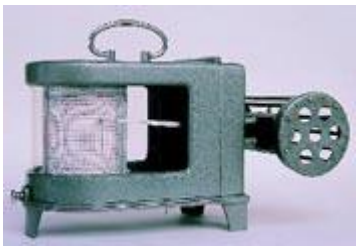
Como todo instrumental científico, un instrumento de uso meteorológico está destinado a evaluar los aspectos cuantitativos de un fenómeno natural. Para cumplir tal función, la Estación Meteorológica debe reunir los siguientes requisitos:

- Sensibilidad.
- Precisión.
- Sensibilidad y precisión constante a través del tiempo.
- Simplicidad de diseño y manejo.
- Facilidad de funcionamiento y manejo.
- Facilidad de funcionamiento y mantenimiento.
- Solidez de construcción.

Los instrumentos, una vez salidos de la fábrica, certificados y autorizados, deben sujetarse a ciertas normas de instalación, manejo, calibración y mantenimiento. A continuación, en la **Tabla N° A.2** se describe una lista de instrumentos meteorológicos más comunes utilizados en las estaciones meteorológicas (para mayor detalle ver referencias [1] al [40]):

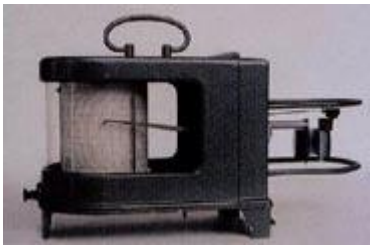
Tabla N° A.2 Instrumentos meteorológicos más comunes.

Instrumento	Descripción	Gráfico
Anemógrafo	Registra continuamente la dirección (grados) de la velocidad instantánea del viento (m/s), la distancia total (en km) recorrida por el viento en relación con el instrumento y las ráfagas (en m/s).	
Anemómetro	Mide la velocidad del viento (m/s) y, en algunos tipos, también la dirección (en grados).	
Barógrafo	Registra continuamente la presión atmosférica en milímetros de mercurio (mm Hg) o en milibares (mb). En el Sistema Internacional de Unidades, la unidad de presión es el hectopascal (hPa). 1 hPa = 1 mb.	

Instrumento	Descripción	Gráfico
Barómetro de Mercurio	Instrumento para medir la presión atmosférica, la cual se equilibra con el peso de una columna de mercurio. Las unidades son el milímetro de mercurio (mm Hg), el milibar (mb) o el hectopascal (hPa).	
Caseta o abrigo meteorológico	Pequeña casilla de paredes de madera, puerta y fondo de doble persiana que favorece la ventilación interior e impide que la radiación solar afecte a los instrumentos colocados en su interior. Deben de estar pintados de blanco.	
Evaporímetro	Aparato para medir la cantidad de agua que se evapora en la atmósfera durante un intervalo de tiempo dado. Se denomina también como atmómetro y es el término general para denominar cualquier aparato que sirva para medir la evaporación. Las unidades son el mililitro (ml) o el milímetro de agua evaporada.	
Heliofanógrafo o heliógrafo	Instrumento que registra la duración de la insolación o brillo solar, en horas y décimos.	
Higrógrafo	Aparato que registra la humedad relativa del aire (%).	

Instrumento	Descripción	Gráfico
Higrotermógrafo	Registra, simultáneamente, la temperatura (°C) y la humedad relativa del aire (%).	
Microbarógrafo	Igual que el barógrafo, pero registra variaciones de la presión mucho menores.	
Piranómetro	Mide la radiación solar global o difusa (cal.cm².mm).	
Pirhelió-metro/grafo	Instrumento para mediar y graficar la radiación solar directa.(cal.cm².mm)	
Pluviógrafo	Registra la cantidad de lluvia caída, en milímetros (mm).	
Pluviómetro	Mide la cantidad de lluvia caída, en milímetros (mm).	

Instrumento	Descripción	Gráfico
Psicrómetro	Mide la humedad relativa (%) de un modo indirecto.	
Satélite Meteorológico	Es un satélite diseñado exclusivamente para recepción y transmisión de información meteorológica. Los datos que proporciona son en su mayoría en tiempo real, especialmente imágenes. Existen dos clases de ellos, los geoestacionarios y los polar-sincrónicos.	
Satélite Meteorológico Geoestacionario	Se caracterizan por permanecer sobre un punto fijo con respecto a la superficie terrestre y una distancia aproximada de 36000 Km de altura. Las imágenes que proporcionan estos satélites tienen una frecuencia de 30 minutos y su resolución espacial va de 8 a 1 Km. De este tipo de satélites es el GOES 8, el cual cubre a toda Centroamérica. Ver Imagen de Satélite.	
Satélite Meteorológico Polar-Sincrónico	Estos satélites tienen órbitas de giro alrededor de la tierra con dirección casi paralela a los meridianos; es decir, recorren el planeta de polo a polo. Su órbita descendente es norte-sur en la mitad hemisférica iluminada por el sol; por el contrario, ascienden de sur a norte en la zona obscura. El	

Instrumento	Descripción	Gráfico
	tiempo aproximado en completar una vuelta es de 12 horas, por lo que completan dos ciclos en un día. Su altura aproximada es de 850 Km y su resolución espacial es mucho más fina que los geoestacionarios.	
Tanque Evaporimétrico	Mide la evaporación en milímetros (mm) de un recipiente o cubeta algo profunda y de bastante superficie en el cual se mide la evaporación por la disminución del nivel del agua.	
Termógrafo	Registra la temperatura del aire en grados Celsius (°C).	
Termómetros de Suelo	Indica la temperatura del suelo a diversas profundidades, en grados Celsius (°C).	
Termómetros de Máxima y Mínima	Indican las temperaturas máxima y mínima del aire (°C) ocurridas en el día.	

A.5 Detalle de las Técnicas de Pronóstico

Para un mejor detalle y análisis de los valores encontrados en base a los algoritmos de pronósticos de temperatura, se ha elaborado la **Tabla N° A.3** durante el periodo de enero a julio de 2014/2015, tomados de la estación meteorológica del SENAMHI sede Huánuco, y sus respectivos pronósticos para el mes de agosto de ese periodo de evaluación.

Tabla N° A.3 Comparación de los Algoritmos de Pronóstico de Temperatura.

MESES DEL AÑO 2014/2015	TEMPERATURA MÁXIMA (°C)	HUMEDAD RELATIVA MÁXIMA (%)	PRESIÓN MÁXIMA (miliBar)
ENERO	29.11111111	92	806.1978673
FEBRERO	28.05555556	91	806.2655951
MARZO	27.72222222	92	807.654015
ABRIL	28.94444444	91	808.5683403
MAYO	29.72222222	89	808.7715237
JUNIO	28.88888889	87	809.2456183
JULIO	27.83333333	89	810.0922158
PRONÓSTICO VÍA REGRESIÓN LINEAL SIMPLE			
JULIO	28.59325397	ERROR	-0.759920635
PRONÓSTICO VÍA REGRESIÓN LINEAL MÚLTIPLE			
JULIO	27.80746202	ERROR	0.025871315

Detalle del Algoritmo de Regresión Lineal Simple

El análisis de regresión lineal simple (RLS) permite estudiar una relación estocástica entre dos variables X e Y, donde los valores posibles de Y se pueden asociar con cualquier valor de X. A continuación se detalla el modelo poblacional de la regresión lineal simple:

$$Y_i = \beta_0 + \beta_1 X_i + U_i \qquad \textbf{(A.1)}$$

Dónde:

- β_0, β_1: Coeficientes de regresión a estimarse; β_0 es denominado intercepto y β_1 es denominado pendiente.
- Y_i: Variable respuesta, explicada, variable pronosticada para la i-ésima observación.
- X_i: Variable independiente, explicativa, predictora, regresora, etc.
- U_i: Variable aleatoria no observable que puede tomar cualquier valor, se le conoce como variable perturbadora o error estadístico. Esta variable representa a las demás variables no consideradas en el modelo, a los errores de muestreo y cualquier otro aspecto no especificado en el modelo.

El supuesto es que la variable aleatoria Y está formada por una parte predecible la cual

es función lineal de X y una parte no predecible que es el error aleatorio, este error aleatorio (U_i) incluye efectos de todos los otros factores no considerados en el modelo. Asimismo se debe tener en cuenta que la variable explicativa X debe ser considerada como fija, es decir, X es una variable matemática medida sin error.

A continuación se listan los supuestos asumidos en el modelo de regresión lineal simple:

Supuesto N° 1: En promedio el valor esperado de los errores U_i es Cero (0), es decir, hay errores por exceso y por defecto que en promedio se anulan.

$$E\{U_i|X_i\} = 0 \qquad \textbf{(A.2)}$$

Supuesto N° 2: El error en la i-ésima observación no depende del error cometido en la j-ésima observación, cuando esta suposición no es satisfecha se tiene un problema de autocorrelación.

$$\boldsymbol{Covarianza} \begin{cases} \boldsymbol{Cov}(U_i, U_j) = \boldsymbol{E}\big[(U_i - E(U_i))(U_j - E(U_j))\big] \\ \boldsymbol{E}(U_i U_j) = \boldsymbol{0}, \qquad \boldsymbol{i \neq j} \end{cases} \qquad \textbf{(A.3)}$$

Esto quiere decir que U_i y U_j no están correlacionados. También se conoce como la independencia de las observaciones.

Supuesto N° 3: La varianza de los errores para cada error, es un número constante; representa el supuesto de homo cedasticidad o igual dispersión, es decir, que las poblaciones tienen igual varianza. Esto es:

$$V(U_i|X_i) = E[U_i - E(U_i)]^2 = E[U_i]^2 \; = \; \sigma^2 \qquad \textbf{(A.4)}$$

En situaciones prácticas, lo que está al alcance del investigador de campo es una muestra de valores de Y correspondiente a las X's fijos, por consiguiente la tarea es la estimación de los parámetros $\widehat{\beta_0}$ y $\widehat{\beta_1}$ utilizando la información de la muestra recopilada.
El modelo de regresión lineal de la muestra se describe a continuación:

$$\widehat{Y_i} = \widehat{\beta_0} + \widehat{\beta_1} X_i + e_i \qquad \textbf{(A.5)}$$

Dónde:

- $\widehat{\beta_0}$: Término constante, es la ordenada en el origen o intercepto y se interpreta como el valor estimado o predicho de Y cuando X es 0.

- $\widehat{\beta_1}$: Pendiente, es el cambio pronosticado en Y cuando hay un cambio unitario en X.
- e_i: Término Residual.
- $\widehat{Y_i}$: Variable respuesta, variable de la i-ésima muestra de campo.
- X_i: Variable independiente, explicativa, predictora, regresora, etc.

Para la estimación de los parámetros de regresión $\widehat{\beta_0}$ y $\widehat{\beta_1}$ se emplea el método de los mínimos cuadrados ordinarios (MCO), que consiste en minimizar las sumas de los cuadrados residuales. Se sabe que:

$$Y_i = \widehat{Y_i} + e_i \Rightarrow e_i = Y_i - \widehat{Y_i} \qquad \textbf{(A.6)}$$

Esto requiere decir que el error de la muestra es la diferencia entre el valor observado y el valor estimado. Con el método de mínimos cuadrados ordinarios se desea minimizar la siguiente expresión:

$$Q = \sum_{i=1}^{n} {e_i}^2 = \sum_{i=1}^{n} \left(Y_i - \widehat{\beta_0} - \widehat{\beta_1} X_i\right)^2 \qquad \textbf{(A.7)}$$

Derivando la expresión respecto a $\widehat{\beta_0}$ y $\widehat{\beta_1}$ se obtiene:

$$\frac{\partial Q}{\partial \widehat{\beta_0}} = -2 \sum_{i=1}^{n} \left(Y_i - \widehat{\beta_0} - \widehat{\beta_1} X_i\right)$$
$$\frac{\partial Q}{\partial \widehat{\beta_1}} = -2 \sum_{i=1}^{n} \left(X_i Y_i - X_i \widehat{\beta_0} - \widehat{\beta_1} {X_i}^2\right) \qquad \textbf{(A.8)}$$

Igualando a cero las expresiones de la ecuación anterior, y luego lo ordenamos para obtener las siguientes ecuaciones normales:

$$\sum_{i=1}^{n} Y_i = n\widehat{\beta_0} + \widehat{\beta_1} \sum_{i=1}^{n} (X_i)$$
$$\sum_{i=1}^{n} X_i Y_i = \widehat{\beta_0} \sum_{i=1}^{n} (X_i) + \widehat{\beta_1} \sum_{i=1}^{n} \left({X_i}^2\right) \qquad \textbf{(A.9)}$$

Al despejar los valores de $\widehat{\beta_0}$ y $\widehat{\beta_1}$ de las expresiones de la ecuación anterior se tiene:

$$\widehat{\beta_1} = \frac{n\sum_{i=1}^{n}(X_iY_i) - \sum_{i=1}^{n}(X_i)\sum_{i=1}^{n}(Y_i)}{n\sum_{i=1}^{n}(X_i{}^2) - \left(\sum_{i=1}^{n}(X_i)\right)^2} \qquad \widehat{\beta_0} = \bar{y} - \widehat{\beta_1}\bar{x} \qquad \textbf{(A.10)}$$

Para verificar si los valores hallados son mínimos, se obtiene la segunda derivada:

$$\frac{\partial^2 Q}{\partial^2 \widehat{\beta_0}} = 2n > 0$$
$$\frac{\partial^2 Q}{\partial^2 \widehat{\beta_1}} = 2\sum_{i=1}^{n}(X_i{}^2) > 0 \qquad \textbf{(A.11)}$$

Como los valores son siempre positivos, entonces los valores de los estimadores hallados son mínimos. De acuerdo con el teorema de Gauss-Markov, se puede concluir que los estimadores mínimos cuadrados hallados son óptimos o de mínima varianza dentro de la clase de estimadores insesgados que son funciones lineales de las observaciones. A continuación se detallan las ecuaciones matemáticas más útiles respecto a los parámetros econométricos relevantes para el cálculo de la verificación del modelo de regresión lineal simple:

$$V(\widehat{\beta_0}) = \frac{\sigma_e{}^2}{n} + \bar{x}^2\left(\frac{\sigma_e{}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}\right) \qquad Desv.Estándar(\widehat{\beta_0}) = \sqrt{V(\widehat{\beta_0})}$$
$$V(\widehat{\beta_1}) = \frac{\sigma_e{}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \qquad Desv.Estándar(\widehat{\beta_1}) = \sqrt{V(\widehat{\beta_1})}$$
$$Donde: \sigma_e{}^2 = s_e{}^2 = \frac{\sum_{i=1}^{n}(e_i)^2}{n-2} = \frac{\sum_{i=1}^{n}(Y_i - \widehat{Y_i})^2}{n-2} = \frac{\sum_{i=1}^{n}Y_i{}^2 - \widehat{\beta_0}\sum_{i=1}^{n}Y_i - \widehat{\beta_1}\sum_{i=1}^{n}X_iY_i}{n-2} \qquad \textbf{(A.12)}$$

Los intervalos de confianza para los parámetros $\widehat{\beta_0}$ y $\widehat{\beta_1}$ vienen dados por las siguientes expresiones basadas en la función de distribución de T Student:

$$\beta_0 \in \langle \widehat{\beta_0} \mp t_{(1-\alpha/2,n-2)}s(\widehat{\beta_0})\rangle$$
$$\beta_1 \in \langle \widehat{\beta_1} \mp t_{(1-\alpha/2,n-2)}s(\widehat{\beta_1})\rangle \qquad \textbf{(A.13)}$$

Con la finalidad de saber que tan bien predice la variable estímulo a la variable respuesta, es importante analizar la variación de la variable Y. La variación total de los valores observados de Y alrededor de su media puede ser dividida en dos: una atribuible al modelo de regresión (variación explicable) y la otra a factores aleatorios (variación no explicable), tal como se muestra en la siguiente expresión:

$$\underset{\Downarrow}{\underset{Variación\ Total}{\sum_{i=1}^{n}(Y_i-\bar{Y})^2}} = \underset{\Downarrow}{\underset{Variación\ Explicada}{\sum_{i=1}^{n}(\hat{Y}_i-\bar{Y})^2}} + \underset{\Downarrow}{\underset{Variación\ No\ Explicada}{\sum_{i=1}^{n}(Y_i-\hat{Y})^2}} \quad \textbf{(A.14)}$$

Dónde:

- $SCT=\sum_{i=1}^{n}(Y_i-\bar{Y})^2$: **Suma de Cuadrados Total (SCT).** Expresa las desviaciones de las observaciones respecto al promedio total. Si SCT tiende al valor cero, se concluye que no existe variabilidad en la variable respuesta.
- $SCR=\sum_{i=1}^{n}(\hat{Y}_i-\bar{Y})^2$: **Suma de Cuadrados de la Regresión (SCR).** Expresa las desviaciones de los valores ajustados respecto al promedio de los valores de Y. Si el valor de SCR se aproxima al valor de SCT, se concluye que el modelo propuesto es adecuado.
- $SCE=\sum_{i=1}^{n}(Y_i-\hat{Y})^2$: **Suma de Cuadrados del Error (SCE).** Expresa las desviaciones de los valores observados respecto a los valores ajustados. Si SCE tiende a cero, entonces todas las observaciones caen en la línea de regresión, por consiguiente el modelo es adecuado.

Usualmente esta partición se representa en una tabla llamada Tabla de Análisis de Varianza, conocida también como Anova o Anva, la cual se muestra en la **Tabla N° A.4**:

Tabla N° A.4 Análisis de Varianza (Anova) para el Modelo de Regresión Simple.

Fuente de Variación	Grados de Libertad	Suma de Cuadrados	Cuadrado Medio	F_0
Debido a la Regresión	1	$SCR=\sum_{i=1}^{n}(\hat{Y}_i-\bar{Y})^2$	$CMR=\frac{SCR}{1}$	$F_0=\frac{CMR}{CME}$
Debido al Error	n-2	$SCE=\sum_{i=1}^{n}(Y_i-\hat{Y})^2$	$CME=\frac{SCE}{n-2}=s_e{}^2$	
Total	n-1	$SCT=\sum_{i=1}^{n}(Y_i-\bar{Y})^2$		

Para verificar la validez estadística del modelo de regresión lineal simple propuesto, se utiliza lo siguiente:

a) Coeficiente de determinación: R^2.

b) Coeficiente de Correlación Lineal Simple: r.

c) Pruebas de Significación: Pruebas T y F.

a) **Coeficiente de Determinación ($R^2 = \frac{SCR}{SCT}$)**: Indica en qué porcentaje la variable estímulo explica a la variable respuesta. Este coeficiente expresa la relación entre dos tipos de variación:

- V1=Variación de los valores de Y alrededor de la línea de regresión.
- V2=Variación de los valores de Y alrededor de su propia media.

Por consiguiente R^2 expresado en porcentaje, mide la variación total en Y explicada por el modelo de regresión. Por ser R^2 cociente entre dos sumas de cuadrados, luego de multiplicarlo por 100 %, el mínimo valor que puede tomar es 0 y el máximo valor que puede tomar es 100%.

$$R^2 \times 100\% = \frac{SCR}{SCT} \times 100\% = \frac{V1}{V2} \times 100\% \qquad \textbf{(A.15)}$$

b) **Coeficiente de Correlación Lineal Simple: (r):** Es una medida que indica el grado de asociación lineal entre dos variables; se obtiene de la siguiente expresión:

$$r = \frac{\sum_{i=1}^{n} X_i Y_i - \frac{(\sum_{i=1}^{n} X_i)(\sum_{i=1}^{n} Y_i)}{n}}{\sqrt{\sum_{i=1}^{n} X_i{}^2 - \frac{(\sum_{i=1}^{n} X_i)^2}{n}} \sqrt{\sum_{i=1}^{n} Y_i{}^2 - \frac{(\sum_{i=1}^{n} Y_i)^2}{n}}} \qquad \textbf{(A.16)}$$

En el caso de la regresión lineal simple, se cumple que $r = \pm\sqrt{R^2}$. En un modelo de regresión lineal simple, el signo del coeficiente de correlación corresponde al signo de la pendiente $\widehat{\beta_1}$. El rango de r es: $-1 \leq r \leq 1$.

Si el coeficiente de correlación es positivo y tiende a 1, se dice que hay una relación directa y significativa entre las variables, si el coeficiente de correlación es negativo y tiende a -1, se dice que hay una relación inversa y significativa entre las variables, si el coeficiente es cero no existe relación entre las variables.

c) Pruebas de Significación de las Variables (Pruebas T y F):

i. **Prueba T**: Las pruebas individuales o pruebas T son independientes para cada

parámetro del modelo de regresión lineal simple. El procedimiento se detalla a continuación:

i. Hipótesis:

$H_0: \beta_i = 0$ (la variable Xi no es significativa en el modelo)

$H_1: \beta_i \neq 0$ (la variable Xi si es significativa en el modelo)

ii. Especificación del nivel de significación o riesgo:

α y suposiciones

iii. Obtención de la estadística de prueba:

$$t_0 = \frac{\widehat{\beta}_i - \beta_i}{s(\widehat{\beta}_i)} = \frac{\widehat{\beta}_i}{s(\widehat{\beta}_i)}$$

Tal que Ho es verdadera, donde:

$\widehat{\beta}_i$: Estimador.

$s(\widehat{\beta}_i)$: Error estándar del estimador.

iv. Región crítica (RC) y regla de decisión (RD)

$$RC = \langle -\infty, t_{\left(\frac{\alpha}{2}, n-2\right)} \rangle \cup \langle t_{\left(1-\frac{\alpha}{2}, n-2\right)}, \infty \rangle$$

Rechazar Ho si $t_0 \subset RC$, es decir, si se cumple:

$$t_0 < t_{\left(\frac{\alpha}{2}, n-2\right)} \qquad \text{ó} \qquad t_0 > t_{\left(1-\frac{\alpha}{2}, n-2\right)}$$

También se puede calcular y utilizar el P_value definido como:

$$P\_value = 2 \times P\left(t_{(n-2)} > t_0\right).$$

v. Si t_0, valor de la estadística de prueba, pertenece a la región crítica, se rechaza la hipótesis nula; en caso contrario no se rechaza (H_0).

ii. **Prueba F**: Esta prueba nos permite determinar si el modelo lineal es apropiado o aceptable para explicar la relación entre las variables de estudio. El procedimiento se detalla a continuación:

i. Hipótesis por probar:

$H_0: \beta_i = 0$ (el modelo no es apropiado)

$H_1: \beta_i \neq 0$ (el modelo si es apropiado)

ii. Especificación del nivel de significación o riesgo:

α y suposiciones

iii. Obtención de la estadística de prueba:

$$F_0 = \frac{CMR}{CME} \sim F_{(1,n-2)}$$

Tal que Ho es verdadera.

iv. Región crítica (RC) y regla de decisión (RD)

$$RC = \langle F_{(1-\alpha,1,n-2)}, \infty \rangle \quad (Prueba\ de\ una\ Cola\ a\ la\ Derecha)$$

$$F_{Crítico} = F_{(1-\alpha,1,n-2)}$$

También se puede calcular y utilizar el P_value definido como:

$$P\_value = P\left(F_{(1,n-2)} > F_0\right).$$

v. Si $F_0 > F_{Crítico}$ se rechaza la hipótesis nula (H_0); en caso contrario no se rechaza (H_0). Si $P\_value < \alpha$, entonces se rechaza la hipótesis nula (H_0).

A.6 Referencias de Internet

La siguiente lista es una recopilación de dominios públicos los cuales fueron consultados durante el desarrollo de esta Tesis durante los primeros meses del año 2014/2015, cuyo contenido se adjunta en el CD elaborado por el autor, el cual servirá como aporte para futuras investigaciones en esta área.

http://www.raspberrypi.org/
Artículos y recursos gratuitos de Raspberry Pi.

http://arduino.cc/
Descripción y Teoría sobre el micro controlador Arduino.

http://www.lighttpd.net/
Software y manuales del servidor web Light TPD.

http://www.sqlite.org/
Software e Información sobre el motor de base de datos SQLite.

http://www.mathworks.com/
Información útil sobre procesamiento de algoritmos de pronóstico en Matlab.

http://www.ni.com/labview/esa/
Información útil sobre procesamiento de señales meteorológicas en Labview.

http://www.wmo.int/pages/index_es.html
Página web oficial de la Organización Meteorológica Mundial (OMM).

https://www.google.com.pe
Página web del buscador Google.

http://es.wikipedia.org/wiki/Wiki
Página web de la Enciclopedia Libre.

ANEXO B
DETALLE DE LOS LENGUAJES DE PROGRAMACIÓN

Debido a lo extenso que resulta documentar el detalle de la codificación de los algoritmos, solo se citará las partes más resaltantes, dejando toda la información de la tesis en un CD para mayor referencia.

B.1 Códigos de Programación

A continuación, se listan extractos de los códigos en C, matlab-R, PHP, Python y cron Linux que han sido elaborados para probar la factibilidad técnica del prototipo.

B.1.1 Codificación del Raspberry Pi en Phyton, SQLite, Apache

Para poder utilizar la base de datos SQLite de señales Meteorológicas provenientes de los sensores, se tuvo que desarrollar un driver en lenguaje Python para el compilador Arduino que permite pasar las señales digitalizadas hacia el Raspberry-Pi sobre el sistema operativo Linux para publicarlo en Web. Detalle de los comandos de instalación en entorno Linux de SQLite – Lighttpd – PHP sobre Raspberry-Pi:

- $ sudo apt-get install lighttpd
- $ sudo service lighttpd stop
- $ sudo apt-get install php5 php5-cgi php5-sqlite
- $ sudo lighttpd-enable-mod fastcgi fastcgi-php
- $ sudo service lighttpd start
- $ cd /var/www
- $ sqlite3 datos_Meteoro.db
- $ wget http://downloads.sourceforge.net/project/openopc/openopc/1.2.0/OpenOPC-1.2.0.source.tar.bz2
- $ tar -xjf OpenOPC-1.2.0.source.tar.bz2
- $ cd OpenOPC-1.2.0/src/

- $ sudo apt-get install pyro
- $ python opc.py

Instalación de las librerías para comunicación serial entre Arduino y Raspberry:

- sudo apt-get install python-serial

Nota: El tiempo aproximado para estos pasos es de aproximadamente 20 minutos. Cabe recalcar que todas las señales analizadas han sido obtenidas de los sensores y almacenados en la Base de Datos SQLite, el programa escribe_meteoro.py, fue compilado usando el Python sobre Linux, instalando previamente sus librerías. En concreto el código del programa es:

```
import sqlite3
import datetime
import time
import serial
# variables globales
dbname='/var/www/datos_Meteoro.db'
ctrl_T = "T"
# arduino = serial.Serial('/dev/ttyACM0',9600)
class Estacion(object):
        def __init__(self, timestamp, temperatura, presion, humedad, monoxido, dioxido, intensidadUV,velocidadViento,direccionViento,lluvia,temperaturaEstacion,presionEstacion,humedadEstacion):
                self.timestamp = timestamp
                self.temperatura = temperatura
                self.presion = presion
                self.humedad = humedad
                self.monoxido = monoxido
                self.dioxido = dioxido
                self.intensidadUV = intensidadUV
                self.velocidadViento = velocidadViento
                self.direccionViento = direccionViento
                self.lluvia = lluvia
                self.temperaturaEstacion = temperaturaEstacion
                self.presionEstacion = presionEstacion
                self.humedadEstacion = humedadEstacion

        def log_estacion(self):
                # almacena los datos de los sensores en la base de datos
                conn=sqlite3.connect(dbname)
                curs=conn.cursor()
                consulta = "insert into sensor_data (timestamp, temperatura, presion, humedad, monoxido, dioxido, intensidadUV, velocidadViento, direccionViento, lluvia, temperaturaEstacion, presionEstacion, humedadEstacion) values ("+self.timestamp+"',"+self.temperatura+"',"+self.presion+"',"+self.humedad+"',"+self.monoxido+"',"+self.dioxido+"',"+self.intensidadUV+"',"+self.velocidadViento+"',"+self.direccionViento+"',"+self.lluvia+"',"+self.temperaturaEstacion+"',"+self.presionEstacion+"',"+self.humedadEstacion+")"
```

```
                print( consulta )
                curs.execute(consulta)
                # guardar los cambios
                conn.commit()
                conn.close()
                print("insert exitoso")
        # muestra el contenido de los sensores
        def display_sensoresBD(self):
                conn=sqlite3.connect(dbname)
                curs=conn.cursor()
                for row in curs.execute("SELECT * FROM sensor_data"):
                        print( str(row[0])+"    "+str(row[1])+"     "+str(row[2])+"     "  +str(row[3])+"     "  +str(row[4])+"     "
+str(row[5])+"    " +str(row[6])+"    " +str(row[7])+"    " +str(row[8])+"    " +str(row[9])+"    " +str(row[10])+"    " +str(row[11])+"    "
+str(row[12])+"    " +str(row[13])+"    " +str(row[14]) )
                conn.close()
        # muestra el contenido de la base de datos
        def display_data(self):
                print( "Timestamp: " + self.timestamp)
                print( "Temperatura: " + self.temperatura)
                print( "Presion: " + self.presion)
                print( "Humedad: " + self.humedad)
                print( "Monoxido: " + self.monoxido)
                print( "Dioxido: " + self.dioxido)
                print( "Intensidad Ultravioleta: " + self.intensidadUV)
                print( "Velocidad de Viento: " + self.velocidadViento)
                print( "Direccion de Viento: " + self.direccionViento)
                print( "Caida de Lluvia: " + self.lluvia)
                print( "Temperatura de Estacion: " + self.temperaturaEstacion)
                print( "Presion de Estacion: " + self.presionEstacion)
                print( "Humedad de Estacion: " + self.humedadEstacion)
                # obtiene datos de los sensores desde el Arduino via USB Serial
        def get_sensores(self):
                arduino = serial.Serial('/dev/ttyACM0',9600)
                arduino.open()
                valor = ""
                comandoT="T"
                comandoP="P"
                comandoH="H"
                comandoM="M"
                comandoC="C"
                comandoU="U"
                comandoV="V"
                date=datetime.datetime.now()
                self.timestamp=date.strftime("%Y-%m-%d %H:%M:%S")

                arduino.write(comandoT)
                time.sleep(0.5)
                while arduino.inWaiting() > 0 :
                        self.temperatura = str(valor) + str(arduino.readline())
                #toma.temperatura = str(valor) + str(arduino.readline().strip())
```

```
                        #toma.temperatura = str(valor) + str(arduino.read(1))

                        arduino.write(comandoP)
                        time.sleep(0.5)
                        while arduino.inWaiting() > 0 :
                                self.presion = str(valor) + str(arduino.readline())

                        arduino.write(comandoH)
                        time.sleep(0.5)
                        while arduino.inWaiting() > 0 :
                                self.humedad = str(valor) + str(arduino.readline())
                        arduino.write(comandoM)
                        time.sleep(0.5)
                        while arduino.inWaiting() > 0 :
                                self.monoxido = str(valor) + str(arduino.readline())
                        arduino.write(comandoC)
                        time.sleep(0.5)
                        while arduino.inWaiting() > 0 :
                                self.dioxido = str(valor) + str(arduino.readline())
                        arduino.write(comandoU)
                        time.sleep(0.5)
                        while arduino.inWaiting() > 0 :
                                self.intensidadUV = str(valor) + str(arduino.readline())
                        arduino.write(comandoV)
                        time.sleep(0.5)
                        while arduino.inWaiting() > 0 :
                                valor = str(valor) + str(arduino.readline())
                        datos=valor.split(',')
                        self.direccionViento = datos[0]
                        self.velocidadViento = datos[1]
                        self.lluvia = datos[2]
                        self.temperaturaEstacion = datos[3]
                        self.humedadEstacion = datos[4]
                        self.presionEstacion = datos[5]
                        arduino.close()

# Funcion principal donde el programa inicia
def main():

        tomaDatos = Estacion("2015-10-07 07:00:19.120","",1.1,2.0,3.0,4.0,5.0,6.0,120.45,7.0,8.0,9.0,10.0)
        tomaDatos.get_sensores() #Recoje datos y actualiza variable Estacion
        tomaDatos.display_data() # Muestra los datos recolectados

        if tomaDatos.temperatura != "":
          tomaDatos.log_estacion() # Graba informacion en la Base de Datos
          tomaDatos.display_sensoresBD() # Muestra los datos grabados en la Base de Datos
        else:
                print( "no grabo!")
if __name__=="__main__":
   main()
```

B.1.2 Codificación del Arduino en C++

A continuación se muestran extractos de código en C++ para el Arduino que permiten enviar las señales digitalizadas de los sensores de Temperatura, Humedad y Presión hacia el Raspberry-Pi para que puedan ser almacenados en la base de datos y posteriormente publicados vía web-scada.

```
/* ========================================================================== */
/*   estacion.c                                                               */
/*   Este codigo integra la obtencion de diversos sensores conectados al Arduino */
/* ========================================================================== */
#include <OneWire.h>
#include <DallasTemperature.h>
// DQ esta conectado al pin 9 de Arduino para Temperatura
#define ONE_WIRE_BUS 9
// Configuramos para comunicar con otros dispositivos 1-Wire
OneWire oneWire(ONE_WIRE_BUS);
// Indicamos el pin asignado al sensor 1-Wire a DallasTemperature
DallasTemperature sensores(&oneWire);
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(11, 10); // RX | TX para Bluetooth
SoftwareSerial VSerial(5, 6); // RX | TX para Viento y Lluvia
/*
SE DEBE TENER EN CUENTA LA ASIGNACIÓN DE LOS PINES:
A0 : SENSOR DE MONOXIDO
A1 : SENSOR DE PRESION
A2 : SENSOR DE HUMEDAD
A3 : SENSOR DE CO2
A4 : SENSOR DE ULTRAVIOLETA

D9: SENSOR DE TEMPERATURA
D10: TX BTSerial -Blouetooth
D11: RX BTSerial
D5: RX VSerial - Viento y Lluvia
D6: TX VSerial
*/

float tempobj;
String mensaje = "";
char buffer[20];
long retardo;
long time;
char databuffer[35];
double temp;

void setup(void)
{
  pinMode(A0, INPUT); //Sensor Monoxido como entrada
  pinMode(A1, INPUT); //Sensor Presión como entrada
  pinMode(A2, INPUT); //Sensor Humedad como entrada
  pinMode(A3, INPUT); //Sensor CO2 como entrada
  pinMode(A4, INPUT); //Sensor Ultravioleta como entrada

  Serial.begin(9600); // Protocolo USB Serial RS232
  BTSerial.begin(9600); // Protocolo RS232 Bluetooth
  VSerial.begin(9600); //Protocolo RS232 Viento y Lluvia
  retardo=millis();
  sensores.begin(); //Protocolo 1W Temperatura
}
void loop(void)
{
 if (Serial.available()){
  char sensor = Serial.read();
   if (sensor == 'T'){
```

```
  Serial.print("");
  Serial.print(readTEMPERATURA());

} else if (sensor=='P'){
  Serial.print("");
  Serial.print(readPRESION(A1));
} else if (sensor=='H'){
  Serial.print("");
  Serial.print(readHUMEDAD(A2));
} else if (sensor=='M'){
  Serial.print("");
  Serial.print(readMONOXIDO(A0));
} else if (sensor=='C'){
    int percentage;
    float volts;
    Serial.print("");
    volts = readCO2(A3);
    //Serial.print(volts);
    //Serial.println( "  V   " );
    percentage=readCO2Percentage(volts);
    if (percentage == -1) {
          //Serial.print( "<400" );
          Serial.print( "400" );
       } else {
          Serial.print(percentage);
       }
    //Serial.print( "ppm" );
    //Serial.print( "       Tiempo:" );
    //Serial.print(millis());
    //Serial.print("\n");

} else if (sensor=='U'){
    float uvLevel = readULTRAVIOLETA(A4);
    float outputVoltage = 5.0 * uvLevel/1024;
    float uvIntensity = mapfloat(outputVoltage, 0.99, 2.9, 0.0, 15.0);
    //Serial.print(" UV Salida Analogica: ");
    //Serial.print(uvLevel);
    //Serial.print(" Salida de Voltaje: ");
    //Serial.print(outputVoltage);
    //Serial.print(" UV Intensidad: ");
    Serial.print(uvIntensity);
    //Serial.print(" mW/cm^2");

} else if (sensor=='V'){
  getBuffer(); //Inicio!
  Serial.print("");
  //Serial.print("Direccion de Viento: ");
  Serial.print(WindDirection());
  Serial.print(",");
  //Serial.print("Promedio de la Velocidad de Viento (Un Minuto): ");
  //Serial.print(WindSpeedAverage());
  //Serial.println("m/s  ");
  //Serial.print("Max Velocidad del Viento (Cinco Minutos): ");
  Serial.print(WindSpeedMax());
  Serial.print(",");
  //Serial.println("m/s");
  //Serial.print("Caida de Lluvia (Una Hora): ");
  //Serial.print(RainfallOneHour());
  //Serial.println("mm  ");
  //Serial.print("Caida de Lluvia (24 Horas): ");
  Serial.print(RainfallOneDay());
  Serial.print(",");
  //Serial.println("mm");
  //Serial.print("Temperatura: ");
  Serial.print(Temperature());
  Serial.print(",");
```

```
    //Serial.println("C  ");
    //Serial.print("Humedad: ");
    Serial.print(Humidity());
    Serial.print(",");
    //Serial.println("%  ");
    //Serial.print("Presion Barometrica: ");
    Serial.print(BarPressure());
    //Serial.println("hPa");
    Serial.println("");
  }

 }

if (BTSerial.available()) {
   while(BTSerial.available())
   {
     char inchar = (char)BTSerial.read();
     mensaje += inchar;
   }

   mensaje.toCharArray(buffer, sizeof(buffer));

   if ((buffer[0]=='P')||(buffer[0]=='T')||(buffer[0]=='H')||(buffer[0]=='M')||(buffer[0]=='C')||(buffer[0]=='U')||(buffer[0]=='V')||(buffer[0]=='A'))
   {
     if (buffer[0]=='A')
     {
       BTSerial.print("A");
     }
     if (buffer[0]=='P')
     {
       BTSerial.print("");
       BTSerial.println(readPRESION(A1));
     }
     if (buffer[0]=='T')
     {
       BTSerial.print("");
       BTSerial.println(readTEMPERATURA());
       //Para test enviamos la Temperatura al USB-Serial
       Serial.print(readTEMPERATURA());
     }
     if (buffer[0]=='H')
     {
       BTSerial.print("");
       BTSerial.println(readHUMEDAD(A2));
     }
     if (buffer[0]=='M')
     {
       BTSerial.print("");
       BTSerial.println(readMONOXIDO(A0));
     }
     if (buffer[0]=='C')
     {
       int percentage;
       float volts;
       BTSerial.print("");
       volts = readCO2(A3);
       BTSerial.print(volts);
       BTSerial.println( "  V   " );
       percentage=readCO2Percentage(volts);
       if (percentage == -1) {
             BTSerial.print( "<400" );
          } else {
             BTSerial.print(percentage);
          }
          BTSerial.print( "ppm" );
          BTSerial.print( "       Tiempo:" );
```

```
          BTSerial.print(millis());
          BTSerial.print("\n");

    }
    if (buffer[0]=='U')
    {
      float uvLevel = readULTRAVIOLETA(A4);
      float outputVoltage = 5.0 * uvLevel/1024;
      float uvIntensity = mapfloat(outputVoltage, 0.99, 2.9, 0.0, 15.0);
      BTSerial.print(" UV Salida Analogica: ");
      BTSerial.print(uvLevel);
      BTSerial.print(" Salida de Voltaje: ");
      BTSerial.print(outputVoltage);
      BTSerial.print(" UV Intensidad: ");
      BTSerial.print(uvIntensity);
      BTSerial.print(" mW/cm^2");
    }
    if (buffer[0]=='V')
    {
        getBuffer(); //Inicio!
        BTSerial.print("Direccion de Viento: ");
        BTSerial.print(WindDirection());
        BTSerial.println("  ");
        BTSerial.print("Promedio de la Velocidad de Viento (Un Minuto): ");
        BTSerial.print(WindSpeedAverage());
        BTSerial.println("m/s  ");
        BTSerial.print("Max Velocidad del Viento (Cinco Minutos): ");
        BTSerial.print(WindSpeedMax());
        BTSerial.println("m/s");
        BTSerial.print("Caida de Lluvia (Una Hora): ");
        BTSerial.print(RainfallOneHour());
        BTSerial.println("mm  ");
        BTSerial.print("Caida de Lluvia (24 Horas): ");
        BTSerial.print(RainfallOneDay());
        BTSerial.println("mm");
        BTSerial.print("Temperatura: ");
        BTSerial.print(Temperature());
        BTSerial.println("C  ");
        BTSerial.print("Humedad: ");
        BTSerial.print(Humidity());
        BTSerial.println("%  ");
        BTSerial.print("Presion Barometrica: ");
        BTSerial.print(BarPressure());
        BTSerial.println("hPa");
        BTSerial.println("");
        BTSerial.println("");      }
   }
   else
   {
      tempobj=atof(buffer);
   }

   mensaje="";

  }

}

float readTEMPERATURA(){
  float val=0; // variable donde guardaremos la temperatura leida del sensor
    sensores.requestTemperatures(); //Enviamos el comando para obtener la temperatura
    val =  sensores.getTempCByIndex(0); // Almacenamos la temperatura en la variable val
  return val;
}
```

```
float readFiltroTEMPERATURA(){
  float sample = 0;   //reset the sample value
  for(char i = 0; i <= 64; i++){
    sensores.requestTemperatures(); //Enviamos el comando para obtener la temperatura
    sample += sensores.getTempCByIndex(0); // Almacenamos temperatura de 64 muestras en la variable
   }
   sample = sample / 64;  //Obtenemos el promedio de 64 muestras
  return sample;
 }

float readHUMEDAD(int pin){
   int pressureValue = analogRead(pin);
   float pressure=((pressureValue-0.844))/0.031;
   return pressure;
}

float readMONOXIDO(int pin){
   int monoxidoValue = analogRead(pin);
   float monoxido=(monoxidoValue);
   return monoxido;
}

float readPRESION(int pin){
   int pressureValue = analogRead(pin);
   float pressure=((pressureValue/1024.0)+0.095)/0.000009;
   return pressure;
}

float readCO2(int pin){ // pin=3
    /************************Software Related Macros************************************/
    #define        READ_SAMPLE_INTERVAL        (50)    //define how many samples you are going to take in normal operation
    #define        READ_SAMPLE_TIMES           (5)     //define the time interval(in milisecond) between each samples in
                                                       //normal operation
    /************************Application Related Macros*********************************/
    //These two values differ from sensor to sensor. user should derermine this value.
    #define        ZERO_POINT_VOLTAGE          (0.324)
    //define the output of the sensor in volts when the concentration of CO2 is 400PPM
    #define        REACTION_VOLTGAE            (0.020)
    //define the voltage drop of the sensor when move the sensor from air into 1000ppm CO2

   float volts;
   int i;
   for (i=0;i<READ_SAMPLE_TIMES;i++) {
      volts += analogRead(pin);
      delay(READ_SAMPLE_INTERVAL);
   }
   volts = (volts/READ_SAMPLE_TIMES) *5/1024 ;
   return volts;
}

/*****************************  readCO2Percentage ***********************************
Input:   volts   - SEN-CO2 output measured in volts
         pcurve  - pointer to the curve of the target gas
Output:  ppm of the target gas
Remarks: By using the slope and a point of the line. The x(logarithmic value of ppm)
         of the line could be derived if y(MG-811 output) is provided. As it is a
         logarithmic coordinate, power of 10 is used to convert the result to non-logarithmic
         value.
*********************************************************************************/
int readCO2Percentage(float volts){
    /************************Hardware Related Macros************************************/
    #define        DC_GAIN                     (8.5)   //define the DC gain of amplifier
    /************************Application Related Macros*********************************/
    //These two values differ from sensor to sensor. user should derermine this value.
    #define        ZERO_POINT_VOLTAGE          (0.324)
```

```
    //define the output of the sensor in volts when the concentration of CO2 is 400PPM
    #define         REACTION_VOLTGAE            (0.020)
    //define the voltage drop of the sensor when move the sensor from air into 1000ppm CO2
    /*****************************Globals***********************************************/
    float           CO2Curve[3]  =  {2.602,ZERO_POINT_VOLTAGE,(REACTION_VOLTGAE/(2.602-3))};
                                    //two points are taken from the curve.
                                    //with these two points, a line is formed which is
                                    //"approximately equivalent" to the original curve.
                                    //data format:{ x, y, slope}; point1: (lg400, 0.324), point2: (lg4000, 0.280)
                                    //slope = ( reaction voltage ) / (log400 –log1000)
  float *pcurve=CO2Curve;

  if ((volts/DC_GAIN )>=ZERO_POINT_VOLTAGE) {
      return -1;
    } else {
      return pow(10, ((volts/DC_GAIN)-pcurve[1])/pcurve[2]+pcurve[0]);
    }

}

float readULTRAVIOLETA(int pin){
    byte numberOfReadings = 8;
    unsigned int runningValue = 0;

    for(int x = 0 ; x < numberOfReadings ; x++)
      runningValue += analogRead(pin);
    runningValue /= numberOfReadings;
    float ultravioleta=(runningValue);
    return ultravioleta;
}

float mapfloat(float x, float in_min, float in_max, float out_min, float out_max)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}

float readVIENTO_LLUVIA(int pin){
    int pressureValue = analogRead(pin);
    float pressure=(pressureValue);
    return pressure;
}

void getBuffer() //Obtenemos el estado de los datos ambientales Viento-Lluvia
{
  int index;
  for (index = 0;index < 35;index ++)
  {
    if(VSerial.available())
    {
      databuffer[index] = VSerial.read();
      if (databuffer[0] != 'c')
      {
        index = -1;
      }
    }
    else
    {
      index --;
    }
  }
}

int transCharToInt(char *_buffer,int _start,int _stop) //char a int
{
  int _index;
  int result = 0;
```

```
  int num = _stop - _start + 1;
  int _temp[num];
  for (_index = _start;_index <= _stop;_index ++)
  {
    _temp[_index - _start] = _buffer[_index] - '0';
    result = 10*result + _temp[_index - _start];
  }
  return result;
}

int WindDirection() //Direccion del Viento
{
  return transCharToInt(databuffer,1,3);
}

float WindSpeedAverage() //Velocidad del Aire (1 minuto)
{
  temp = 0.44704 * transCharToInt(databuffer,5,7);
  return temp;
}

float WindSpeedMax() //Máxima Velocidad del Aire (5 minutos)
{
  temp = 0.44704 * transCharToInt(databuffer,9,11);
  return temp;
}

float Temperature() //Temperatura ("C")
{
  temp = (transCharToInt(databuffer,13,15) - 32.00) * 5.00 / 9.00;
  return temp;
}

float RainfallOneHour() //Caída de Lluvia (1 hora)
{
  temp = transCharToInt(databuffer,17,19) * 25.40 * 0.01;
  return temp;
}

float RainfallOneDay() //Caída de Lluvia (24 horas)
{
  temp = transCharToInt(databuffer,21,23) * 25.40 * 0.01;
  return temp;
}

int Humidity() //Humedad
{
  return transCharToInt(databuffer,25,26);
}

float BarPressure() //Presión Barometrica
{
  temp = transCharToInt(databuffer,28,32);
  return temp / 10.00;
}
```

B.1.3 Codificación del Algoritmo de Predicción en Matlab y R

Se utilizaron dos variantes del algoritmo de predicción basado en la regresión lineal: simple y múltiple. Para comprobar el algoritmo se utilizaron los datos proporcionados por el SENAMHI provenientes de la estación meteorológica de Huánuco correspondiente al periodo del año 2015.

```
%  Regresión Lineal Múltiple en Lenguaje Matlab
Y=[29.11; 28.05; 27.72; 28.94; 29.72; 28.88;27.83];
X=[1 1 806.19; 1 2 806.26; 1 3 807.65; 1 4 808.56; 1 5 808.77; 1 6 809.24; 1 7 810.09];
% Cálculo de la Matriz B = INV(X' * X) * X' *Y
B=inv(X'*X)*X'*Y
%  Y_est= bo+b1*X_1,i+b2*X_2,j+b3*X_3,k
%  Y_est = X * B
%  B = INV(X' * X) * X' *Y
% X_1i          Y                    X_2j        X_3k
%Mes             Temperatura                  Presión    Humedad Relativa

//Regresion Lineal Simple en Lenguaje R
out <- lm(y ~ x)
summary(out)
plot(x, y)
abline(out)
```

B.2 Estructura del Modelado de la Base de Datos SQLite

Para poder utilizar la aplicación en el servidor web, se utilizó la tecnología PHP, en donde se programó la lógica interna para procesar los datos meteorológicos almacenados en la base de datos SQLite. La página principal index.php hace una llamada al archivo grafica.php, que contiene la funcionalidad de acceso a la base de datos. A continuación, se detalla el fragmento de código SQL de la base de datos SQLite de la estación meteorológica:

```
CREATE TABLE `sensor_data` (
`id` INTEGER PRIMARY KEY AUTOINCREMENT,
`timestamp` DATETIME DEFAULT 'current_timestamp',
`temperatura` REAL NOT NULL,
`presion` REAL,
`humedad` REAL,
`monoxido` REAL,
`dioxido` REAL,
`intensidadUV` REAL,
`velocidadViento` REAL,
`direccionViento` REAL,
`lluvia` REAL,
`temperaturaEstacion` REAL,
`presionEstacion` REAL,
`humedadEstacion` REAL
);


CREATE TABLE wind_ranges (
lbound INT,
hbound INT,
[desc] TEXT,
nr INT
);
```

Detalle del fragmento de código fuente de la función PHP grafica.php:

```
<?php


        include("PHPGraphLib/phpgraphlib.php");
        date_default_timezone_set('America/lima');

        $grafica    = $_GET['grafico'];
        $fecha1     = $_GET['fecha1'];
```

```
	$fecha2	= $_GET['fecha2'];
	$opcion	= $_GET['opcion'];
	$datos		= $_GET['datos'];

$dbDb		= "datos_Meteoro.db";	// Nombre de la base de datos
$dbTabla	= "sensor_data";	// Nombre de la tabla



	$bd = new SQLite3($dbDb);

	if ($opcion=='turnos')
	{
		$query="select timestamp,".$grafica." from ".$dbTabla." where strftime('%d/%m/%Y', timestamp) BETWEEN '".$fecha1."' AND '".$fecha2."' limit ".$datos;
	}
	else if($opcion=='dias')
	{
		$query="select strftime('%Y-%m-%d', timestamp),Avg(".$grafica.") from ".$dbTabla." where strftime('%d/%m/%Y', timestamp) BETWEEN '".$fecha1."' AND '".$fecha2."' group by strftime('%Y-%m-%d', timestamp) limit ".$datos;
	}
	else if($opcion=='semana')
	{
		$query="select strftime('%Y-%m-%W', timestamp),Avg(".$grafica.") from ".$dbTabla." where strftime('%d/%m/%Y', timestamp) BETWEEN '".$fecha1."' AND '".$fecha2."' group by strftime('%Y-%m-%W', timestamp) limit ".$datos;
	}
	else if($opcion=='meses')
	{
		$query="select strftime('%Y-%m', timestamp),Avg(".$grafica.") from ".$dbTabla." where strftime('%d/%m/%Y', timestamp) BETWEEN '".$fecha1."' AND '".$fecha2."' group by strftime('%Y-%m', timestamp) limit ".$datos;
	}
	else if($opcion=='anios')
	{
		$query="select strftime('%Y', timestamp),Avg(".$grafica.") from ".$dbTabla." where strftime('%d/%m/%Y', timestamp) BETWEEN '".$fecha1."' AND '".$fecha2."' group by strftime('%Y', timestamp) limit ".$datos;
	}

	// Revisamos la selección del usuario
	if ($grafica=='direccionViento')
	{
		$query="select d.direccionViento, r.desc as descripcion, r.nr, ((COUNT(d.direccionViento) * 100.00 / (select COUNT(*) from sensor_data where strftime('%d/%m/%Y', timestamp) BETWEEN '".$fecha1."' AND '".$fecha2."' ))) AS porcentaje FROM sensor_data AS d INNER JOIN wind_ranges AS r ON r.lbound <= d.velocidadViento AND d.velocidadViento < r.hbound where strftime('%d/%m/%Y', timestamp) BETWEEN  '".$fecha1."' AND '".$fecha2."' GROUP BY d.direccionViento, r.desc ";
	}

	$results = $bd->query($query);

	$dataArray=array();

	if ($grafica=='direccionViento') {

		while ($row = $results->fetchArray())
		{

			//$angulo=$row[0];

			switch($row[0]){
				case ($row[0] >= 348.75 or $row[0] < 11.25):
					$angulo="N";
					break;
				case ($row[0] >= 11.25 and $row[0] < 33.75):
					$angulo="NNE";
					break;
				case ($row[0] >= 33.75 and $row[0] < 56.25):
					$angulo="NE";
					break;
				case ($row[0] >= 56.25 and $row[0] < 78.75):
					$angulo="ENE";
					break;
				case ($row[0] >= 78.75 and $row[0] < 101.25):
					$angulo="E";
```

```
                                        break;
                                case ($row[0] >= 101.25 and $row[0] < 123.75):
                                        $angulo="ESE";
                                        break;
                                case ($row[0] >= 123.75 and $row[0] < 146.25):
                                        $angulo="SE";
                                        break;
                                case ($row[0] >= 146.25 and $row[0] < 168.75):
                                        $angulo="SSE";
                                        break;
                                case ($row[0] >= 168.75 and $row[0] < 191.25):
                                        $angulo="S";
                                        break;
                                case ($row[0] >= 191.25 and $row[0] < 213.75):
                                        $angulo="SSW";
                                        break;
                                case ($row[0] >= 213.75 and $row[0] < 236.25):
                                        $angulo="SW";
                                        break;
                                case ($row[0] >= 236.25 and $row[0] < 258.75):
                                        $angulo="WSW";
                                        break;
                                case ($row[0] >= 258.75 and $row[0] < 281.25):
                                        $angulo="W";
                                        break;
                                case ($row[0] >= 281.25 and $row[0] < 303.75):
                                        $angulo="WNW";
                                        break;
                                case ($row[0] >= 303.75 and $row[0] < 326.25):
                                        $angulo="NW";
                                        break;
                                case ($row[0] >= 326.25 and $row[0] < 348.75):
                                        $angulo="NNW";
                                        break;
                        }

                        $porcentaje=$row[3];

                        $bucketNum=$row[2];

                        if ($bucketNum=='0') {
                                if (array_key_exists($angulo, $dataArray)) {
                                        foreach($dataArray as $key => $field){
                                                if($key == $angulo){
                                                        $dataArray[$key]                        =
$porcentaje+$dataArray[$key];
                                                }
                                        }
                                } else {
                                        $dataArray[$angulo]=$porcentaje;
                                }
                        }
                        else if ($bucketNum=='1') {
                                if (array_key_exists($angulo, $dataArray)) {
                                        foreach($dataArray as $key => $field){
                                                if($key == $angulo){
                                                        $dataArray[$key]                        =
$porcentaje+$dataArray[$key];
                                                }
                                        }
                                } else {
                                        $dataArray[$angulo]=[0,$porcentaje];
                                }
                        }
                        else if ($bucketNum=='2') {
                                if (array_key_exists($angulo, $dataArray)) {
                                        foreach($dataArray as $key => $field){
                                                if($key == $angulo){
                                                        $dataArray[$key]                        =
$porcentaje+$dataArray[$key];
                                                }
                                        }
                                } else {
                                        $dataArray[$angulo]=[0,0,$porcentaje];
```

```
                                        }
                                }
                                else if ($bucketNum=='3') {
                                        if (array_key_exists($angulo, $dataArray)) {
                                                foreach($dataArray as $key => $field){
                                                        if($key == $angulo){
                                                                $dataArray[$key]                        =
$porcentaje+$dataArray[$key];
                                                        }
                                                }
                                        } else {
                                                $dataArray[$angulo]=[0,0,0,$porcentaje];
                                        }
                                }
                                else if ($bucketNum=='4') {
                                        if (array_key_exists($angulo, $dataArray)) {
                                                foreach($dataArray as $key => $field){
                                                        if($key == $angulo){
                                                                $dataArray[$key]                        =
$porcentaje+$dataArray[$key];
                                                        }
                                                }
                                        } else {
                                                $dataArray[$angulo]=[0,0,0,0,$porcentaje];
                                        }
                                }
                                else if ($bucketNum=='5') {
                                        if (array_key_exists($angulo, $dataArray)) {
                                                foreach($dataArray as $key => $field){
                                                        if($key == $angulo){
                                                                $dataArray[$key]                        =
$porcentaje+$dataArray[$key];
                                                        }
                                                }
                                        } else {
                                                $dataArray[$angulo]=[0,0,0,0,0,$porcentaje];
                                        }
                                }
                                else if ($bucketNum=='6') {
                                        if (array_key_exists($angulo, $dataArray)) {
                                                foreach($dataArray as $key => $field){
                                                        if($key == $angulo){
                                                                $dataArray[$key]                        =
$porcentaje+$dataArray[$key];
                                                        }
                                                }
                                        } else {
                                                $dataArray[$angulo]=[0,$porcentaje];
                                        }
                                        $dataArray[$angulo]=[0,0,0,0,0,0,$porcentaje];
                                }
                                else if ($bucketNum=='7') {
                                        if (array_key_exists($angulo, $dataArray)) {
                                                foreach($dataArray as $key => $field){
                                                        if($key == $angulo){
                                                                $dataArray[$key]                        =
$porcentaje+$dataArray[$key];
                                                        }
                                                }
                                        } else {
                                                $dataArray[$angulo]=[0,0,0,0,0,0,0,$porcentaje];
                                        }
                                }
                                //print_r($dataArray);

                        }
                }
                else {

                        while ($row = $results->fetchArray())
                        {
                                $dat1=$row[0];

                                $sensor=$row[1];
```

```
                        $dataArray[$dat1]=$sensor;
                }
        }
        $bd->close();


        if ($grafica=='direccionViento') {
        //Gráfico del Velocímetro
        require_once('jpgraph/src/jpgraph.php');
        require_once ('jpgraph/src/jpgraph_windrose.php');
        //---------------------------------------------------------
        // First create a new windrose graph with a title
        //---------------------------------------------------------
        $graph = new WindroseGraph (640,480);
        //---------------------------------------------------------
        // Setup title
        //---------------------------------------------------------
        $graph->title-> Set('Grafica de Direccion y Velocidad de Viento');
        $graph->title-> SetFont( FF_VERDANA, FS_BOLD,12);
        $graph->title-> SetColor( 'navy');

        //---------------------------------------------------------------
        // Create the windrose plot. (Each graph may have multiple plots)
        // The default plot will have 16 compass axis.
        //---------------------------------------------------------------
        $wp = new WindrosePlot ($dataArray);
        $wp->SetRadialGridStyle('solid');
        $wp->SetType(WINDROSE_TYPE16);
        // Let the library determine a good angle
        $wp->scale->SetAngle('auto');
        $graph->Add( $wp);
        //---------------------------------------------------------------
        // Send the graph to the browser
        //---------------------------------------------------------------
        $graph->Stroke();

        } else  {

                $graph = new PHPGraphLib(1200,500);
                $graph->addData($dataArray);
                $graph->setupXAxis(40, 'black');
                $graph->setTitle($grafica);
                $graph->setBars(false);
                $graph->setLine(true);
                $graph->setDataPoints(true);
                $graph->setDataPointColor('maroon');
                $graph->setDataValues(true);
                $graph->setDataValueColor('maroon');
                $graph->setGoalLineColor('red');
                $graph->createGraph();

        }
?>
```

A continuación se detalla el fragmento de código fuente de index.php:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
 <head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
   <title>ESTACION METEOROLOGICA DE LA LAGUNA DE MANCAPOZO</title>
   <link href="layout.css" rel="stylesheet" type="text/css">
   <script language="javascript" type="text/javascript" src="http://localhost:9080/estacion/jquery.js"></script>
   <script language="javascript" type="text/javascript" src="http://localhost:9080/estacion/jquery.flot.js"></script>
        <script language="javascript" type="text/javascript" src="http://localhost:9080/estacion/jquery.flot.time.js"></script>

        <script src="src/jquery-1.8.3.min.js" type="text/javascript" charset="utf-8"></script>
        <script src="src/jquery.maskedinput.js" type="text/javascript"></script>
        <script src="src/jquery-ui.min.js"></script>
        <link type="text/css" rel="stylesheet" href="src/jquery-ui.css" />
```

```
<script type="text/javascript">
   $(function() {$("#date1").datepicker({ dateFormat:"dd/mm/yy"}).mask("99/99/9999");});
   $(function() {$("#date2").datepicker({ dateFormat:"dd/mm/yy"}).mask("99/99/9999");});
</script>

      <script type="text/javascript">
        $(function ()
                                    {

        $('#tiporeporte').change(function()
                                    {

                                    var fecha1;
                                    var fecha2;
                                    var opcion;

                                    fecha1=document.getElementById('date1').value;
                                    fecha2=document.getElementById('date2').value;
                                    opcion = $('input:radio[name="radio"]:checked').val();
                                    datos=document.getElementById('datos').value;

                                    image = document.getElementById('imagen1');

                                    document.getElementById('UNO').style.display='block';
                                    document.getElementById('TODOS').style.display='none';

          switch($(this).val()) {
               case '1':
                                                                        image = document.getElementById('imagen0');

           image.src="grafica.php?grafico=temperatura&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;
                                                                        break;
               case '2':
                                                                        image = document.getElementById('imagen0');

           image.src="grafica.php?grafico=presion&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;
                   break;
               case '3':
                                                                        image = document.getElementById('imagen0');

           image.src="grafica.php?grafico=humedad&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;
                   break;
               case '4':
                                                                        image = document.getElementById('imagen0');

           image.src="grafica.php?grafico=monoxido&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;
                   break;
               case '5':
                                                                        image = document.getElementById('imagen0');

           image.src="grafica.php?grafico=dioxido&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;
                   break;
               case '6':
                                                                        image = document.getElementById('imagen0');

           image.src="grafica.php?grafico=intensidadUV&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;
                   break;
               case '7':
                                                                        image = document.getElementById('imagen0');

           image.src="grafica.php?grafico=velocidadViento&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;
                   break;
               case '8':
                                                                        image = document.getElementById('imagen0');

           image.src="grafica.php?grafico=direccionViento&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;
                   break;
               case '9':
                                                                        image = document.getElementById('imagen0');

           image.src="grafica.php?grafico=lluvia&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;
                   break;
               case '10':
                                                                        image = document.getElementById('imagen0');

           image.src="grafica.php?grafico=temperaturaEstacion&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;
                   break;
               case '11':
                                                                        image = document.getElementById('imagen0');

           image.src="grafica.php?grafico=presionEstacion&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;
```

```
                break;
              case '12':
                                                                image = document.getElementById('imagen0');

            image.src="grafica.php?grafico=humedadEstacion&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;
                break;
              case '13':

                                                                document.getElementById('UNO').style.display='none';
                                                                document.getElementById('TODOS').style.display='block';

                                                                image = document.getElementById('imagen1');

            image.src="grafica2.php?grafico=temperatura&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;

                                                                image = document.getElementById('imagen2');

            image.src="grafica2.php?grafico=presion&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;



                                                                image = document.getElementById('imagen3');

            image.src="grafica2.php?grafico=humedad&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;

                                                                image = document.getElementById('imagen4');

            image.src="grafica2.php?grafico=monoxido&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;

                                                                image = document.getElementById('imagen5');

            image.src="grafica2.php?grafico=dioxido&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;

                                                                image = document.getElementById('imagen6');

            image.src="grafica2.php?grafico=intensidadUV&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;

                                                                image = document.getElementById('imagen7');

            image.src="grafica2.php?grafico=velocidadViento&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;

                                                                image = document.getElementById('imagen8');

            image.src="grafica2.php?grafico=direccionViento&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;

                                                                image = document.getElementById('imagen9');

            image.src="grafica2.php?grafico=lluvia&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;

                                                                image = document.getElementById('imagen10');

            image.src="grafica2.php?grafico=temperaturaEstacion&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;

                                                                image = document.getElementById('imagen11');

            image.src="grafica2.php?grafico=presionEstacion&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;

                                                                image = document.getElementById('imagen12');

            image.src="grafica2.php?grafico=humedadEstacion&fecha1="+fecha1+"&fecha2="+fecha2+"&opcion="+opcion+"&datos="+datos;



                break;

              default:
                //  default code block
              }
            });

          });

      </script>
</head>
  <body>
  <h1> </h1>
  <table width="578" border="0" align="center">
    <tr>
      <td width="383" align="center" class="post-title">ESTACION METEOROLOGICA DE LA LAGUNA DE MANCAPOZO</td>
    </tr>
    <tr>
      <td align="center"><table width="313" border="0" align="center">
```

```
    <tr>
      <td height="153" colspan="2"><form id="form1" name="form1" method="post" action="">
        <table width="465" border="0">
          <tr>
            <td colspan="2" align="center">RANGO DE FECHA</td>
          </tr>
          <tr>
            <td colspan="2" align="center"><input id="date1" value="1000" type="text" tabindex="1" name="f1" />
              a
            <input id="date2" value="1231" type="text" tabindex="1" name="f2" /></td>
          </tr>
          <tr>
            <td width="92"><input type="radio" name="radio" id="turnos" value="turnos" />
              <label for="turnos">Turnos</label></td>
            <td width="118"><input name="radio" type="radio" id="meses" value="meses" checked="CHECKED" />
            <label for="meses">Meses</label></td>
          </tr>
          <tr>
            <td><input type="radio" name="radio" id="dias" value="dias" />
              <label for="dias">Dias</label></td>
            <td><input type="radio" name="radio" id="anios" value="anios" />
              <label for="anios">Años</label></td>
          </tr>
          <tr>
            <td height="21"><input type="radio" name="radio" id="semana" value="semana" />
              <label for="semana">Semanas</label></td>
            <td> </td>
          </tr>
          <tr>
            <td height="21" colspan="2" align="center"> </td>
          </tr>
        </table>
      </form></td>
    </tr>
  </table></td>
</tr>
<tr>
  <td align="center">TIPO GRAFICA
    <select name="tiporeporte" id="tiporeporte">
      <option value="N">Seleccione una opcion</option>
      <option value="1">TEMPERATURA</option>
      <option value="2">PRESION</option>
      <option value="3">HUMEDAD</option>
      <option value="4">MONOXIDO</option>
      <option value="5">DIOXIDO</option>
      <option value="6">INTENSIDAD ULTRAVIOLETA</option>
      <option value="7">VELOCIDAD_VIENTO</option>
      <option value="8">DIRECCION_VIENTO</option>
      <option value="9">INTENSIDAD_LLUVIA</option>
      <option value="10">T_ESTACION</option>
      <option value="11">P_ESTACION</option>
      <option value="12">H_ESTACION</option>
      <option value="13">TODOS</option>
  </select></td>
</tr>
<tr>
  <td align="center"><form name="form2" method="post" action="">
    <label for="datos">Cantidad de datos </label>
    <input name="datos" type="text" id="datos" value="50" size="10">
  </form></td>
</tr>

          <tr >
                    <td >

          <table  border="0" align="center" id="UNO">
                      <tr >
                        <td ><div id="grafica0" align='center'><img src="" id="imagen0"/></div></td>
              </tr>

</table>

          <table  border="0" align="center" id="TODOS">
                      <tr >
```

```
                        <td ><div id="grafica1" align='center'><img src="" id="imagen1"/></div></td>
        <td ><div id="grafica2" align='center'><img src="" id="imagen2"/></div></td>
                </tr>
                        <tr >
                        <td ><div id="grafica3" align='center'><img src="" id="imagen3"/></div></td>
        <td ><div id="grafica4" align='center'><img src="" id="imagen4"/></div></td>
                </tr>
                        <tr >
                        <td ><div id="grafica5" align='center'><img src="" id="imagen5"/></div></td>
        <td ><div id="grafica6" align='center'><img src="" id="imagen6"/></div></td>
                </tr>
                        <tr >
                        <td ><div id="grafica7" align='center'><img src="" id="imagen7"/></div></td>
        <td ><div id="grafica8" align='center'><img src="" id="imagen8"/></div></td>
                </tr>
                        <tr >
                        <td ><div id="grafica9" align='center'><img src="" id="imagen9"/></div></td>
        <td ><div id="grafica10" align='center'><img src="" id="imagen10"/></div></td>
                </tr>
                        <tr >
                        <td ><div id="grafica11" align='center'><img src="" id="imagen11"/></div></td>
        <td ><div id="grafica12" align='center'><img src="" id="imagen12"/></div></td>
                </tr>
  </table>
     </td>
            </tr>

   </table>

</body>
  </html>
```

BIBLIOGRAFÍA

[1] Andrew K. Dennis: "*Raspberry Pi Home Automation with Arduino".* Packt Publishing Birmingham - Mumbai, 2013.

[2] Jonathan Oxer and Hugh Blemings: "*Arduino Cool Projects for Open Source Hardware"*. Apress - Springer, 2009.

[3] Peter Membrey and David Hows: *"Learn Raspberry Pi with Linux"*. Apress, 2013.

[4] Ruth Suehle and Tom Callaway: "*Raspberry Pi Hacks: Tips & Tools for Making Things with the Inexpensive Linux Computer"*. O'Reilly, 2014.

[5] Jose Luis Fuentes y Yague: "*Iniciación a la Meteorología Agrícola"*. Mundi-Prensa, Madrid España, 1996.

[6] José Francisco Villalpando Ibarra y José Ariel Ruiz Corral: "*Observaciones agro meteorológicas y su uso en la agricultura"*. Uteha Noriega Editores, 1993.

[7] Jorge Suárez y José Alberto Di Candia: "*Meteorología: 3ra Edición"*. Arbo Editores, 1953.

[8] Leopoldo Hernández Robredo: "*Meteorología, Física y Climatología Agrícolas"*. Salvat Editores, 1952.

[9] Aland Bravo: *"Diseño e Implementación de una Biblioteca Digital Distribuida basada en Web Services"*, Tesis de Maestría en Telemática por la Universidad Nacional de Ingeniería UNI, 2013.

[10] Mackenzie L. Davis y Susan J. Masten: "*Ingeniería y Ciencias Ambientales*", McGraw Hill Interamericana, 2005.

[11] T. N. Krishnamurti, H. S. Bedi, V. M. Hardiker, L. Ramaswamy: "*An Introduction to Global Spectral Modelling*". Editorial Springer, 2006.

[12] Anastasios A. Tsonis y James B. Elsner: "*Nonlinear Dynamics in Geosciences*". Editorial Springer, 2007.

[13] Kevin Hamilton, Wataru Ohfuchi: "*High Resolution Numerical Modelling of the Atmosphere and Ocean*". Editorial Springer, 2008.

[14] Xuhui Lee, William Massman y Beverly Law: "*Handbook of Micrometeorology*". Editorial Springer, 2005.

[15] Stefan Bronnimann, Jurg Luterbacher, Tracy Ewen, Henry F. Díaz, Richard S. Stolarski, Urs Neu: "*Climate Variability and Extremes during the Past 100 years*". Editorial Springer 2008.

[16] K. Mohanakumar: "*Stratosphere Troposphere Interactions: An Introduction*". Editorial Springer 2008.

[17] Pukh Raj Rakhecha y Vijai P. Singh: "*Applied Hydrometeorology*". Editorial Springer 2009.

[18] Joseph L. Awange y Erik W. Grafarend: *"Solving Algebraic Computational Problems in Geodesy and Geoinformatics: The Answer to Modern Challenges*". Editorial Springer 2005.

[19] A. Marshak y A. B. Davis: "*3D Radiative Transfer in Cloudy Atmospheres*". Editorial Springer 2005.

[20] Brigitte Markner-Jager: "*Technical English for Geosciences: A Text/Work Book*". Editorial Springer 2008.

[21] Seon K. Park y Liang Xu: "*Data Assimilation for Atmospheric, Oceanic and Hydrologic Applications*". Editorial Springer 2009.

[22] Silas Michaelides: "*Precipitation: Advances in Measurement, Estimation and Prediction*". Editorial Springer 2008.

[23] Soroosh Sorooshian, Kuo-lin Hsu, Erika Coppola, Barbara Tomassetti, Marco Verdecchia and Guido Visconti: "*Hydrological Modelling and the Water Cycle: Coupling the Atmospheric and Hydrological Models*". Springer 2009.

[24] Guido Kanschat, Erik Meinkohn, Rolf Rannacher y Rainer Wehrse: "*Numerical Methods in Multidimensional Radiative Transfer*". Editorial Springer 2009.

[25] Alexander Baklanov, C. S. B. Grimmond, Alexander Mahura, Maria Athanassiadou: "*Meteorological and Air Quality Models for Urban Areas*". Editorial Springer 2009.

[26] Thomas Stocker: "*Introduction to Climate Modelling: Advances in Geophysical and Environmental Mechanics and Mathematics*". Editorial Springer 2011.

[27] Lewis F. Richardson:"*Weather Prediction by Numerical Process*". Cambridge University Press, Londres 1922.

[28] P. Raymond Zealley: "*An Introduction to Forecasting Weather*". Cambridge University Press, Londres 1922.

[29] Joseph L. Awange, Erik W. Grafarend, Bela Palancz, Piroska Zaletnyik: "*Algebraic Geodesy and Geoinformatics*". Editorial Springer 2010.

[30] Jurgen P. Kropp, Hans Joachim Schellnhuber: "*In Extremis: Disruptive Events and Trends in Climate and Hidrology*". Editorial Springer 2011.

[31] S. D. Attri, L. S. Rathore, M. V. K. Sivakumar, S. K. Dash: "*Challenges and Opportunities in Agro meteorology*". Editorial Springer 2011.

[32] R. Nagarasan: "*Drought Assessment*". Editorial Springer 2009.

[33] M. Gebremichael, F. Hossain: "*Satellite Rainfall Applications for Surface Hydrology*". Editorial Springer 2010.

[34] Kevin Sene: "*Hydro-meteorology: Forecasting and Applications*". Editorial Springer 2010.

[35] Antonio Navarra, Valeria Simoncini: "*A Guide to Empirical Orthogonal Functions for Climate Data Analysis*". Editorial Springer 2010.

[36] Mihalis Lazaridis: "*First Principles of Meteorology and Air Pollution*". Editorial Springer 2011.

[37] Xiaofan Li, Shouting Gao: "*Precipitation Modeling and Quantitative Analysis*". Editorial Springer 2012.

[38] Paul C. Stark, Louise M. Ryan, James L. McDonald, Harriet A. Burge: "*Using meteorologic data to predict daily ragweed pollen levels*". Editorial Elseviere. International Journal of Aerobiology 13th edition, 1997.

[39] Yansen Wang, Giap Huynh, Chatt Williamson: "*Integration of Google Maps / Earth with microscale meteorology models and data visualization*". Editorial Elseviere Computers and Geosciences Journal 2013.

[40] Yu Haipeng, Huang Jianping, Li Weijing, and Feng Guolin: "*Development of the Analogue-Dynamical Method for Error Correction of Numerical Forecasts*". Springer Editor, and Chinese Meteorological Society, 90th Anniversary Special Collection, Vol 28 N° 5, 2014.

[41] David Bailey, Edwin Wright: "Practical SCADA for Industry". Editorial Elsevier 2003.

[42] Wolfgang Mahnke Stefan-Helmut Leitner Matthias Damm: "*OPC Unified Architecture*". Editorial Springer, 2009.