Estación meteorológica de bajo costo sobre plataforma Arduino.

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Roberto Rosales Fernández

Tutores: M. Sc. Yadira Ramírez Rodríguez

Ing. Julio Alberto Leyva Durán

La Habana, 2023

“Cuba necesita de los hombres de pensamientos, sobre todo de los hombres de pensamientos claro, no solo hombres que hayan acumulado conocimientos; hombres que pongan sus conocimientos del lado del bien, del lado de la justicia, del lado de la patria, porque vivimos en momentos en que el papel del pensamiento es excepcional, porque solo el pensamiento puede guiar a los pueblos en los instantes de grandes empresas como esta que llevando adelante nuestro pueblo”

(Fidel, 1960)

DEDICATORIA.

La presente Tesis la dedico primeramente a mis padres que con su apoyo incondicional han logrado formar en mi vida la perseverancia, amor y respecto a cada cosa que se realiza en el transcurso de mi vida y gracias a cada detalle de la forma de ser de ellos puedo culminar un peldaño importante en mi área académica. También a mi hermano y a mi esposa pues son parte de mi motivación a seguir adelante y ayudarme siempre en todo lo que necesito.

AGRADECIMIENTOS

Especialmente a mis padres por haber sabido guiarme en todo el transcurso de mi vida y continuamente me han ofrecido lo mejor.

Agradezco a mi familia, no solo por todo el trabajo que han asumido a mi lado, sino también por el apoyo emocional y la seguridad que me han dado durante todo este proceso.

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los 7 días del mes de noviembre del año 2023.

Roberto Rosales Fernández

_____

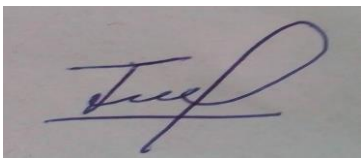
Firma del Autor

M.Sc. Yadira Ramírez Rodríguez

_____

Firma del Tutor

Ing. Julio Alberto Leyva Durán



_____

Firma del Tutor

DATOS DE CONTACTO

Yadira Ramírez Rodríguez graduada de ingeniería en Ciencias informáticas en el 2007.Master en Calidad de Software y profesora auxiliar con más de quince años de experiencia en la asignatura de Ingeniería de Software. Jefa del Departamento Docente de Informática de la Facultad 4 de la Universidad de las Ciencias Informáticas Líneas de Investigación: Ingeniería de Sistemas y Calidad de Software. Correo electrónico: yramirez@uci.cu

Julio Alberto Leyva Durán graduado de ingeniería en Ciencias informáticas en el 2008. Profesor Asistente con más de catorce años de experiencia en la asignatura de Técnicas de Programación en la Universidad de las Ciencias Informáticas. Especialista Superior en la Empresa XETID. Líneas de Investigación: Automática Aplicada. Correo electrónico: jaleyva@uci.cu

RESUMEN

El proyecto expone la creación de una estación meteorológica que muestra en una pantalla de cristal líquido los siguientes parámetros ambientales: temperatura, humedad, velocidad y dirección del viento. El diseño de la solución se basa en la plataforma Arduino; la misma transmite la información de las variables meteorológicas a una aplicación localizada en una computadora personal; por medio de una radio línea. La aplicación de escritorio recolecta y visualiza mediante gráficos los valores de lectura y los almacena en una base de datos local cada un minuto.

El proceso de desarrollo fue guiado por la metodología XP. Como resultado se obtuvo un prototipo de estación meteorológica portátil basada en Arduino con un sistema de información mediante una aplicación de escritorio desarrollada en el lenguaje de programación Java y como gestor de base de datos PostgreSQL.

Palabras clave:

Arduino, estación meteorológica, software libre, hardware libre

REFERENCIAS BIBLIOGRÁFICAS

Introducción 1

CAPÍTULO I: Fundamentación Teórica 5

1.1. Fundamentos teóricos asociados al objetivo de estudio 5

1.2 Estación meteorológica automatizada 6
Figura 1 Estación de meteorológica automatizada (CAMPBELL SCIENTIFIC SPAIN, 2023). 7
1.2.1 Importancia de las estaciones meteorológicas 7
1.2.2. Tipos de estaciones meteorológicas automatizadas 8

1.2. Estaciones de meteorológica de bajo costo 9
1.2.1 Estación meteorológica basado en Raspberry pi 9
Figura 2 Raspberry pi 4. (Raspberry Pi , 2023) 10
1.2.3 Estación meteorológica con la plataforma Arduino 10
Figura 3 Arduino Uno (Smith, 2011). 11
Tabla 1 Características de la placa Arduino Uno Fuente: elaboración propia. 12
1.2.4 Comparación de las principales plataformas para el desarrollo de la estación meteorológica de bajo costo. 12
Tabla 2. Comparación entre hardware y software de las plataformas arduino y raspberry pi Fuente: Elaboración propia 13

1.3 Comunicación serie 14

1.4 Sensores en las estaciones de meteorología 15
Para el diseño de la estación de meteorológica se escogieron los siguientes sensores: 16
Temperatura y humedad relativa del aire EE181. 16
Figura 4 Temperatura y humedad relativa del aire EE181 (Campbell scientifico, 2023). 17
Conjunto de sensores meteorológicos p/n 80422: 17
Figura 5 Sensor de velocidad y dirección del viento. (Arduino, 2023) 17

1.5 Diseño del prototipo del hardware 18

1.6. Metodología de desarrollo 18
1.6.1. Metodología de desarrollo de software Programación Extrema. 20

1.7 Herramientas y tecnologías seleccionadas 22

1.8 Lenguaje de programación 23

1.9. Entorno Integrado de Desarrollo (IDE) 24
1.9 Tipos de sistemas gestores de bases de datos 25
1.9.1 Selección del sistema gestor de base de datos 26

1.10 Conclusiones parciales 26

CAPÍTULO II: Análisis y diseño de la solución propuesta 27

2.1 Descripción de propuesta de solución..... 27

2.2 Requisitos no funcionales..... 28

2.3 Requerimientos funcionales 29
Para la estación meteorológica..... 29
Para la aplicación 29

2.4 Descripción de las historias de usuarios..... 29
Tabla 4 HU1:Medir los datos de sensores. 30
Tabla 5 HU2 Mostrar los parámetros atmosféricos en conjunto con la fecha y hora. 31
Tabla 6 HU3 Permitir la comunicación serial de datos. 31
Tabla 7 HU4 Activar la comunicación con el Arduino. 32
Tabla 8 HU5 Almacenar los datos. 32
Tabla 9 HU6 Representar los datos..... 33

2.5 Plan de iteraciones 33
- Iteración 1: 33
- Iteración 2 34
- Iteración 3: 34
- Iteración 4: 34

Tabla 10 Plan de iteraciones 34

2.6 Plan de entregas 35

Tabla 7 Plan de entrega..... 35

2.7 Diseño arquitectónico 35
2.7.1. Patrón arquitectónico..... 36
Figura 9 Capas de la estación de meteorológica (Fuente: elaboración propia) 37
Figura 10 Aplicación de MVC a la aplicación Fuente: elaboración propia. 37

2.8 Patrones de diseño..... 38
2.8.1 Patrones GRASP..... 39
Figura 11 Bajo acoplamiento Fuente elaboración propia. 39
Figura 12 Patrón Creador Fuente: elaboración propia. 40
Figura 13 Representación del patrón experto de la clase Conexión Fuente: elaboración propia..... 41
2.7.2 Patrones GoF 41
- Comportamiento:..... 41

Figura 14 Ejemplo del patrón de comportamiento aplicado en un método asíncrono elaboración propia. 41

- **Estructurales: 42**

- **Composición..... 42**

2.8 Tarjetas CRC 42
Tabla11 Target CRC de la clase DAOReporteImpl..... 42

Tabla 12 Target CRC de la clase Conexión 42
Tabla 13 Tarjeta CRC de la clase VistaPrincipal 42
Tabla 14 Tarjeta CRC de la clase Controladora 43
Tabla 15 Tarjeta CRC de la clase InterDAOReporteImpl 43

2.9 Conclusiones del capítulo 43

CAPÍTULO III: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA 44

3.1 Fase de implementación 44
3.1.1 Tareas de ingeniería para la Iteración 44

Tabla 16 Tarea 1 Desarrollar un software que permita la Adquisición de los datos. 45

Tabla 17 Tarea 2 Representar los valores en una pantalla de cristal líquido con I2C. 45
3.2 Pruebas software 47

3.2.1 Pruebas de aceptación 49

3.6 Conclusiones parciales 54

Conclusiones 55

RECOMENDACIONES 56

ANEXOs 57

Figura 16 Vista en 3D del diseño de "shield" para el huso del Arduino como parte de la estación de meteorologica Fuente: elaboración propia. 57

Imagen 12 Esquema eléctrico de la tarjeta extensora para Arduino (Fuente: Elaboración propia). 58

Bibliografía 59

ÍNDICE DE TABLAS

TABLA 1 CARACTERÍSTICAS DE LA PLACA ARDUINO UNO FUENTE: ELABORACIÓN PROPIA.12
TABLA 2. COMPARACIÓN ENTRE HARDWARE Y SOFTWARE DE LAS PLATAFORMAS ARDUINO Y RASPBERRY PI FUENTE: ELABORACIÓN PROPIA..... 13
TABLA 3 COMPARACIÓN ENTRE LAS METODOLOGÍAS ÁGILES Y TRADICIONALES FUENTE :ELABORACIÓN PROPIA. 20
TABLA 4 HU1:MEDIR LOS DATOS DE SENSORES. 30
TABLA 5 HU2 MOSTRAR LOS PARÁMETROS ATMOSFÉRICOS EN CONJUNTO CON LA FECHA Y HORA. 31
TABLA 6 PLAN DE ITERACIONES 34
TABLA 7 PLAN DE ENTREGA. 35
TABLA 8 TARGET CRC DE LA CLASE DAOREPORTEIMPL. 42
TABLA 9 TARGET CRC DE LA CLASE CONEXION 42
TABLA 10 TARJETA CRC DE LA CLASE VISTAPRINCIPAL 42
TABLA 11 TARJETA CRC DE LA CLASE CONTROLADORA 43
TABLA 12 TARJETA CRC DE LA CLASE INTERDAOREPORTEIMPL 43
TABLA 13 TAREA 1 DESARROLLAR UN SOFTWARE QUE PERMITA LA ADQUISICIÓN DE LOS DATOS.: 45
TABLA 14 TAREA 2 REPRESENTAR LOS VALORES EN UN PANTALLA DE CRISTAL LÍQUIDO CON I2C 45
TABLA 15 .TAREA 3 CONFIGURAR LA CONEXIÓN SERIAL DEL ARDUINO. 46
TABLA 16 TAREA 4 VALIDAR LOS DATOS DEL PUERTO SERIAL 46
TABLA 17 VALIDACIÓN Y almacenamiento DE LOS DATOS 47
TABLA 18 REPRESENTACIÓN DE LOS DATOS 47
TABLA 19 CÓDIGO DE LA PRUEBA HU1,2,3 50
TABLA 20 CÓDIGO DE LA PRUEBA HU4 50
TABLA 21 CÓDIGO DE LA PRUEBA HU5 51
TABLA 22 CÓDIGO DE LA PRUEBA HU6 52

ÍNDICE DE FIGURAS

FIGURA 1 ESTACIÓN DE METEOROLÓGICA AUTOMATIZADA (CAMPBELL SCIENTIFIC SPAIN, 2023). 7
FIGURA 2 RASPBERRY PI 4 (RASPBERRY PI , 2023) 10
FIGURA 3 ARDUINO UNO (SMITH, 2011). 11
FIGURA 4 TEMPERATURA Y HUMEDAD RELATIVA DEL AIRE EE181 (CAMPBELL SCIENTIFICO, 2023). 17
FIGURA 5 SENSOR DE VELOCIDAD Y DIRECCIÓN DEL VIENTO (ARDUINO, 2023)..... 17
FIGURA 6 VISTA DEL PCB DE LA TARJETA COMPLEMENTARIA DEL ARDUINO FUENTE: ELABORACIÓN PROPIA. 18
FIGURA 7 ESQUEMA DE LA PROPUESTA DE SOLUCIÓN (ELABORACIÓN PROPIA). 28
FIGURA 8 DIAGRAMA CLIENTE SERVIDOR DE LA APLICACIÓN FUENTE: ELABORACIÓN PROPIA..... 36
FIGURA 9 CAPAS DE LA ESTACIÓN DE METEOROLÓGICA (FUENTE: ELABORACIÓN PROPIA) 37
FIGURA 10 APLICACIÓN DE MVC A LA APLICACIÓN FUENTE: ELABORACIÓN PROPIA. 37
FIGURA 11 BAJO ACOPLAMIENTO FUENTE ELABORACIÓN PROPIA. 39
FIGURA 12 PATRÓN CREADOR FUENTE: ELABORACIÓN PROPIA. 40
FIGURA 13 REPRESENTACIÓN DEL PATRÓN EXPERTO DE LA CLASE CONEXIÓN FUENTE: ELABORACIÓN PROPIA. 41
FIGURA 14 EJEMPLO DEL PATRÓN DE COMPORTAMIENTO APLICADO EN UN MÉTODO ASÍNCRONO ELABORACIÓN PROPIA..... 41

INTRODUCCIÓN

Desde sus inicios el hombre se ha preocupado por conocer los diferentes factores meteorológicos debido a que las apariciones de anomalías futuras en las condiciones del tiempo atmosférico influyen en el resultado de sus actividades. Para ellos el ser humano ha realizado un estudio estadístico sobre el clima con el objetivo de estar preparado ante la ocurrencia de un fenómeno desfavorable; mitigando así los efectos negativos del mismo.

En la actualidad la información meteorológica se encuentra más accesible, producto al desarrollo de la tecnología y las comunicaciones que permiten la interconexión de la información de las estaciones meteorológicas de forma global. La Organización Meteorológica Mundial (OMM) creada por Organización de las Naciones Unidas asegura la cooperación entre los servicios meteorológicos internacionales.

En Cuba el Instituto de Meteorología (INSMET) es el encargado de suministrar la información meteorológica oportuna sobre el estado y comportamiento futuro de la atmósfera. Esta información está dirigida a velar por la seguridad de la vida humana y a reducir las pérdidas de bienes materiales ante desastres naturales, contribuyendo directamente al bienestar de la comunidad y al desarrollo sostenible. Para la gestión del pronóstico cuenta con un total de 43 estaciones meteorológicas distribuidas por todo el país que proveen la información del pronóstico del estado futuro del tiempo.

Para realizar el pronóstico del tiempo es preciso obtener continuamente la información sobre las condiciones existentes en la superficie y la atmósfera para posterior procesamiento en un modelo predicción. Como resultado se simula la atmosfera, permitiendo la predicción del posible estado de la misma dentro de un breve lapso de tiempo, pero el comportamiento de los parámetros atmosféricos son complejos y se encuentran en contante fluctuación; por tanto, a un pronóstico proyectado en un futuro más lejano podrían aparecer fenómenos que no fueron contemplados y esos pequeños cambios que se producen en la atmosfera pueden tener grandes efectos y alterar toda la predicción. Por lo que los pronósticos pierden confiabilidad con la proyección en el tiempo. Los mismo son más precisos en áreas más pequeñas debido que las estaciones meteorológicas pueden registrar mayor nivel de detalle, y en ocasiones en lugares donde existan microclimas es necesario registrar esos valores para dar un pronóstico más confiable.

Como consecuencia será una necesidad conocer con mayor exactitud los parámetros meteorológicos en lugares donde las estaciones de meteorología se encuentren distantes y sea necesario conocerlos con exactitud los valores de tiempo atmosférico; con el objetivo de tomar decisiones precisas y efectivas que permita ser previsores ante la ocurrencia de un fenómeno natural desfavorable. A partir

de la situación problemática anteriormente expuesta, se plantea como **problema de la investigación**: ¿Cómo desarrollar una estación meteorológica de bajo costo que provee un reporte de las variables meteorológicas de forma remota?, teniendo como **objeto de estudio**: Las estaciones meteorológicas de bajo costo. Enmarcado en el **campo de acción**: La adquisición de variables meteorológicas mediante sensores en la plataforma Arduino.
Se propone como **objetivo general**: Desarrollar una estación meteorológica de bajo costo mediante la plataforma Arduino.

Para dar cumplimiento al objetivo propuesto se proponen las siguientes **tareas investigativas:**

1. Elaborar el marco teórico de la investigación mediante el estudio de los referentes teóricos sobre el desarrollo de una estación meteorológica de bajo costo con elementos de la plataforma Arduino.
2. Realizar la identificación de los requisitos, análisis y diseño de la estación meteorológica propuesta.
3. Validar los resultados de la investigación aplicando técnicas, métricas y pruebas de software.

Métodos de investigación

Los métodos de investigación son una importante herramienta para la búsqueda y el perfeccionamiento del conocimiento acerca de la realidad. Para dar solución al problema planteado se definen los métodos científicos utilizados en la investigación, clasificados en teóricos y empíricos. A continuación, se describen los métodos teóricos empleados en la presente investigación:

Analítico-sintético: se emplearon en el proceso de análisis documental y revisión bibliográfica, con el objetivo de extraer las ideas esenciales que permitirán fundamentar desde el punto de vista teórico la investigación, así como la propuesta que se realiza.

Modelación: se emplearon para realizar un diseño simplificado de la realidad a través de los diagramas de clases y de componentes y modelación de la base de datos.

Los métodos empíricos: el método empírico es un modelo de investigación que pretende obtener conocimiento a partir de la observación de la realidad. Por ende, está basado en la experiencia. En este modelo, la observación de la realidad es el punto de partida para formular hipótesis, las cuales deben ser sometidas a prueba mediante la experimentación (Ing. Alina Karla Quesada Somano, 2020). A continuación, describimos los métodos empíricos empleados en la presente investigación:

Observación: se empleó en el control del objeto de estudio y análisis del sistema, con el propósito de conocer como los especialistas y técnicos desarrollan los procesos.

Entrevista: se utilizó para, a través de la comunicación verbal, permitió identificar las deficiencias existentes, fue útil a la hora de definir los distintos requisitos para elaborar la solución propuesta, y obtener la información necesaria para el desarrollo de la investigación.

El presente trabajo está dividido en los siguientes tres capítulos que recogen todo lo abordado en la investigación.

Capítulo 1. Fundamentación Teórica: Fundamentación teórica, capítulo encargado de definir los elementos teóricos necesarios para el desarrollo de la investigación y los principales conceptos que se emplearán durante todo el trabajo. Se realiza un análisis de las soluciones similares y se selecciona la metodología de desarrollo de software y las herramientas y tecnologías a utilizar.

Capítulo 2. Análisis y diseño de la solución propuesta: En este capítulo se realizará una breve descripción de la solución propuesta, sus requisitos funcionales y no funcionales. Asimismo, se representarán los artefactos generados a través del proceso de desarrollo del sistema.

Capítulo 3. Validación de la solución propuesta: Implementación y pruebas, capítulo que contiene los estándares de codificación utilizados en la implementación, así como las diferentes pruebas a las que será sometida la aplicación una vez terminada para verificar su correcto funcionamiento.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se precisan un conjunto de elementos para conformar el marco teórico relacionado con el desarrollo de estaciones meteorológica de bajo costo. Dentro de los cuales se analizan las formas de adquisición de variables meteorológicas mediante sensores con la plataforma Arduino y Raspberry pi y tomando como referencias diferentes fuentes de información. Además, se identifican, analizan y se seleccionan las tecnologías y herramientas existentes para la conformación del prototipo de Estación Meteorológica de bajo costo.

1.1. Fundamentos teóricos asociados al objetivo de estudio

Para lograr una comprensión total del presente trabajo, a continuación, se muestran los conceptos relevantes asociados al desarrollo de una estación meteorológica, brindando un soporte teórico y conceptual sobre los siguientes términos:

- **Meteorología:** es la ciencia encargada del estudio de la atmósfera, de sus propiedades y de los fenómenos que en ella tienen lugar, los llamados meteoros. El estudio de la atmósfera se basa en el conocimiento de una serie de magnitudes, o variables meteorológicas, como la temperatura, la presión atmosférica o la humedad, las cuales varían tanto en el espacio como en el tiempo. (Rosa Maria Rodriguez, 2004).
- **Temperatura del aire:** La temperatura es la medida del contenido de calor de un cuerpo o del medio ambiente. En la gran mayoría de estaciones, es medida en grados centígrados(°C). Mediante la medida de esta, se puede medir la influencia de la temperatura en la velocidad en que se desarrolla de los cultivos e insectos y de esa manera predecir la aparición de etapas fenológicas de cultivos y estadios biológicos de insectos (G. Medina García, 2008).
- **Humedad ambiental:** La humedad ambiental es la medida que nos dice la cantidad de vapor agua presente en el aire. Se representa en porcentaje o grado de humedad…También es una variable climática importante presente en el desarrollo de pronósticos de posibles enfermedades de cultivos (G. Medina García, 2008)**.**
- **Velocidad del viento:** Se describe por dos características: velocidad y dirección. El instrumento para medir a este se llama anemómetro y se mide en metros por segundo(m/s) o kilómetros por hora(km/h). Permite determinar la evapotranspiración en cultivos y además es importante por su efecto en la erosión del suelo y daño a los cultivos, así como en la propagación de pesticidas (G. Medina García, 2008)**.**

- **El tiempo atmosférico o meteorológico** es el estado de la atmósfera en un momento y lugar determinado; definido por diversas variables meteorológicas como la temperatura, la presión, el viento, la radiación solar, la humedad y la precipitación (Gutiérrez, 2008).
- **La atmósfera**: es la capa gaseosa que envuelve la Tierra, y que se adhiere a ella gracias a la acción de la gravedad. se extiende desde el suelo hasta los 80-100 kilómetros de altura (Rosa Maria Rodriguez, 2004).
- **Microclimas:** Clima local de características distintas a las de la zona en que se encuentra. (Gutiérrez, 2008).
- **Estaciones meteorológicas.** estaciones meteorológicas son instalaciones dedicadas a medir y registrar regularmente diversas variables meteorológicas. Estos datos se utilizan tanto para la proveer información del clima, la elaboración de predicciones meteorológicas a partir de modelos matemáticos, como para estudios climáticos. Normalmente una estación meteorológica incluye una serie de instrumentos que permiten medir diferentes variables climáticas. Puede incluir, por ejemplo, termómetros para medir temperaturas, barómetros para medir presión atmosférica y pluviómetros para medir la cantidad de precipitación, entre otros posibles diferentes instrumentos. En la actualidad, la mayor parte de las estaciones meteorológicas se hallan automatizadas, de forma que puedan registrar la información durante todo el tiempo y solo se requiera de un mantenimiento ocasional (Melendez, 2022) .

1.2 Estación meteorológica automatizada

Una estación meteorológica automatizada es un sistema autónomo y automático formado por un conjunto de sensores de medición de variables meteorológicas y equipos electrónicos montados sobre una estructura de soporte y conectados a un sistema de adquisición de datos con capacidades de almacenamiento, cálculos y transmisión de información a oficinas centrales (Melendez, 2022).

La utilización de las estaciones meteorológicas automatizadas es el registro datos de forma continua y permitir realizar mediciones en intervalos de tiempo menores que tomando los datos manualmente, además de ser totalmente autónomo al no necesitar intervención humana (Melendez, 2022).

En la Figura 1 que se muestra a continuación se puede observar un ejemplo de estación de meteorológica.

Figura 1 Estación de meteorológica automatizada (CAMPBELL SCIENTIFIC SPAIN, 2023).

1.2.1 Importancia de las estaciones meteorológicas

Desde el inicio el hombre ha debido crear estrategias para adaptarse a las condiciones climáticas. Sin embargo, el desarrollo económico y la urgente necesidad de optimizar los recursos productivos, entre los que se incluye el clima, la obligación de estudiar y entender mejor su comportamiento por medio de la observación sistemática y permanente de variables que lo integran, y gracias a la implementación y uso de estaciones meteorológicas manuales y automáticas (Maldonado I., 2018).

La información meteorológica resulta de gran utilidad en distintas disciplinas como la agronomía y la hidrología, entre otras. La observación de variables y fenómenos meteorológicos se lleva a cabo en Estaciones Meteorológicas Convencionales (EMC) asistidas por un observador capacitado.

En los últimos años, el uso de Estaciones Meteorológicas Automáticas (EMA) ha experimentado un incremento significativo. La OMM las define como "las estaciones en las cuales las observaciones son realizadas y transmitidas automáticamente"(GATTINONI, 2019).

Como complemento se puede decir que la estación meteorológica es una estructura que brinda datos del clima específicamente en el lugar donde se implementa. La importancia de estos datos es vital, dado que los microclimas generados en cada lugar por montes, montañas, sierras, lagos y lagunas, no necesariamente se ven reflejados en los informes del clima para zonas en general. La estación meteorológica proporciona información relevante con la que se podrá tomar decisiones y acciones de forma más certera en diferentes áreas del conocimiento (MONTALVO LEZAMA, 2014).

1.2.2. Tipos de estaciones meteorológicas automatizadas

Existen varios tipos de estaciones meteorológicas que se pueden instalar de acuerdo a un determinado propósito o funcionalidad que se desea utilizar con la base de datos recolectada, por lo cual los tipos de estaciones se pueden clasificar en dos grupos, los cuales son (Melendez, 2022):

- Estaciones meteorológicas para "Investigación y pronóstico".
- Estaciones meteorológicas para "Operación crítica del clima".

Las estaciones meteorológicas de tipo investigación y pronóstico: Los parámetros meteorológicos medidos y registrados en su base de datos, son utilizados para el desarrollo de modelos y pronósticos, lo cual ayudan en determinadas aplicaciones predecir el comportamiento del clima que pudiera afectar una actividad determinada o para la investigación específica de ciertos estudios del clima. Se pueden conocer las siguientes estaciones de este tipo (Melendez, 2022):

- Estación pluviométrica.
- Estación climatológica principal.
- Estación climatológica ordinaria.
- Estación Meteorológica sinóptica.
- Estación Agro-meteorología.

Estaciones meteorológicas operación crítica del clima: son aquellas donde su base de datos medida y registrada de parámetros son utilizados para operar ante condiciones climáticas que pudieran ocasionar situaciones adversas en una zona ante una cierta actividad. Se pueden conocer las siguientes (Melendez, 2022):

- Estación Meteorológica AWOS (Sistema Automático de Observación Meteorológica, *AWOS* por sus siglas en inglés).
- Estación meteorológica para energías renovables.

- Estación meteorológica para calidad de aire.
- Estación meteorológica para seguridad en carreteras.

1.2. Estaciones de meteorológica de bajo costo

Las estaciones meteorológicas y sensores profesionales suelen tener una estructura compleja para lograr una alta precisión por lo que el precio de estos productos suele ser sumamente elevado. Como consecuencia el monitorio de estos sistemas meteorológico una vez implementado conlleva a un costo adicional; siendo no factible del punto de vista económico.

Siendo necesario la búsqueda de opciones que permita tener las mismas prestaciones a un costo más accesibles; dentro las opciones actuales para el desarrollo de una estación de meteorología se encuentran el uso de plataformas hardware y software libre entre las que se encuentran (Bareño, 2011):

- El empleo de computadores de placa única (SBC, Single Board Computers, por sus siglas en inglés), como el conocido Raspberry pi.
- El uso de elementos de la plataforma Arduino.

Posibilitando el desarrollo la una estación meteorológica con menos costos monetarios y con un diseño modular que nos permitan adaptación y mejora del producto.

1.2.1 Estación meteorológica basado en Raspberry pi

El Raspberry Pi es lo que se conoce como ordenador de una sola placa, que es exactamente lo que su nombre indica: como un ordenador de sobremesa, un portátil o un Smartphone, pero construido sobre una única placa de circuito impreso. Como la mayoría de los ordenadores de una sola placa, Raspberry Pi es pequeño más o menos del tamaño de una tarjeta de crédito, pero eso no impide que sea potente: un Raspberry Pi puede hacer cualquier cosa que haga un ordenador más grande y de mayor consumo, aunque puede que no lo haga tan rápido (Halfacree, 2020).

Es un pequeño computador que corre con un sistema operativo Linux capaz de permitirle a las personas de todas las edades explorar la computación. Es capaz de hacer la mayoría de las tareas típicas de un computador de escritorio, desde navegar en internet, reproducir videos en alta resolución, manipular documentos de ofimática, hasta reproducir juegos (MCI ELECTRONICS, 2023).

El raspberry pi es un dispositivo electrónico e informático que también se puede describir como una minicomputadora de bajo costo y tamaño compacto; tiene la capacidad de conectar un monitor o televisor mediante sus conexiones HDMI, y se puede administrar con un teclado y un mouse a través de sus conexiones USB. Al insertar una tarjeta de memoria, es posible instalar un sistema operativo ba-

sado en Linux que requiera pocos recursos informáticos. De esta manera, raspberry Pi puede realizar tareas básicas de una computadora; cuenta con un puerto Wi-Fi y una conexión Ethernet por tanto se puede acceder internet *(Solis, 23)* .

Como principales desventajas del Raspberry pi como parte del desarrollo de la estación de meteorológica serían las siguientes:

- Raspberry pi dispone de un sistema operativo completo que para correr la aplicación en cuestión necesita que arrancar mismo.
- Al ser un miniordenador su consumo energético en ocasiones no es muy factible y su tamaño tampoco.
- El costo de adquisición suele ser excesivo cuando se trata de proyectos sencillos existiendo otras opciones de sistema embebido como es Arduino.

En la Figura 2 se puede observar una Raspberry pi.

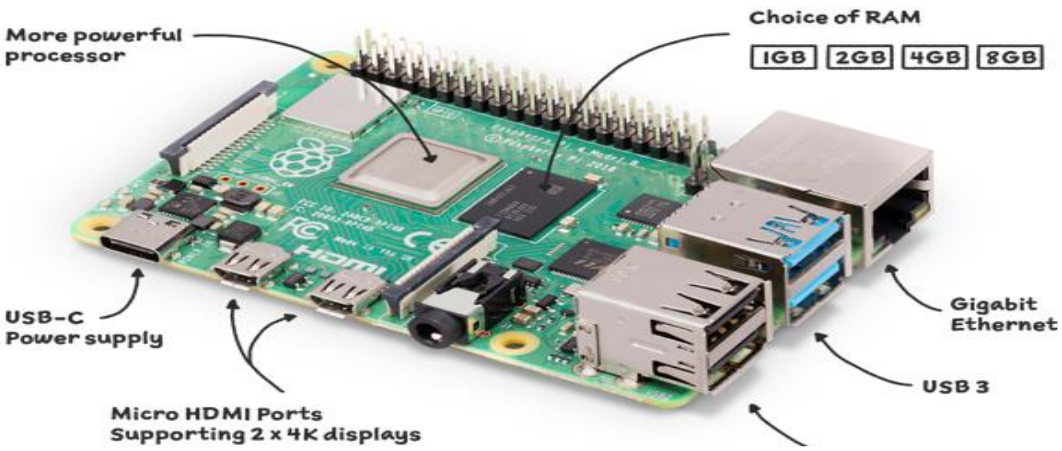


Figura 2 Raspberry pi 4. (Raspberry Pi , 2023)

1.2.3 Estación meteorológica con la plataforma Arduino

Arduino es una plataforma de hardware libre, ampliamente conocida porque facilita el diseño de proyectos electrónicos, se caracteriza por su sencillez y bajo costo; además de su versatilidad para ser programada, contando con un software de desarrollo cuyo lenguaje de programación está basada en C++ (Monk, 2014.).

Arduino se enfoca en acercar y facilitar el uso de la electrónica y programación de sistemas embebidos en proyectos multidisciplinarios. Toda la plataforma, tanto para sus componentes de hardware como de software, son liberados con licencia de código abierto que permite libertad de acceso a ellos. El hardware consiste en una placa de circuito impreso con un microcontrolador, puertos digitales y analógicos de entrada/salida, los cuales pueden conectarse a placas de expansión, que amplían las características de funcionamiento de la placa Arduino. Asimismo, posee un puerto de conexión USB desde donde se puede alimentar la placa y establecer comunicación con el computador. La primera placa Arduino fue introducida en 2005, ofreciendo un bajo costo y facilidad de uso para novatos y profesionales; buscaba desarrollar proyectos interactivos con su entorno mediante el uso de actuadores y sensores (MCL electronics, 2022).

Arduino cuenta con un catálogo muy amplio de placas con microprocesadores y shields que simplifican la conexión de los circuitos necesarios para el desarrollo de los artefactos que el usuario requiera crear (Dixys, y otros, 2018).

Para el desarrollo de una estación meteorológica de bajo costo con Arduino se puede disponer las siguientes ventajas:

- Las placas de Arduino es barato comparadas con otras plataformas de microcontroladores.
- El mismo está especialmente pensada para realizar diferentes proyectos de electrónica. Donde permite la conexión de varios componentes y sensores a los conectores de la placa. Esto facilita la realización de tareas sencillas que no requieran de una programación compleja.
- Es más rápido para el desempeño de funciones básica.
- Su tamaño reducido lo hace atractivo para lugares de poco espacio.

En la Figura 3 se muestras el hardware del Arduino uno:

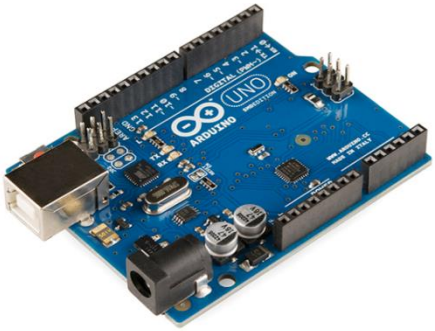


Figura 3 Arduino Uno (Smith, 2011).

Arduino uno es la placa ideal para iniciar en el desarrollo de hardware debido a que es la placa más ampliamente documentada en la familia Arduino (Tabla 1).

Tabla 1 Características de la placa Arduino Uno Fuente: elaboración propia.

Microcontrolador	Atmega328
Voltaje de operación	5V
Voltaje de entrada (recomendado)	7V- 12V
Voltaje de entrada (límite)	6V- 20V
Pines para entrada – salida digital	14 (6 pueden usarse como salida de PWM)
Pines de entrada analógica	Pines de entrada analógica 6
Corriente continua por pin IO	40 mA
Corriente continua en el pin	3.3V 50 mA
Memoria flash	32 KB (0.5 KB ocupados por el bootloader)
SRAM	2 KB
EEPROM	1 KB
Frecuencia de reloj	16 MH

1.2.4 Comparación de las principales plataformas para el desarrollo de la estación meteorológica de bajo costo.

La principal diferencia entre una Arduino UNO y una Raspberry pi es que Raspberry es una micro-computadora requiere un sistema operativo basado en el núcleo de Linux y funciona como el computador de bajos recursos computacionales, mientras que la Arduino UNO es una tarjeta programable que requiere una computadora para programarla con un software específico llamado Arduino IDE, no posee un sistema operativo basad o en Linux, pero si es capaz de manejar tareas concurrente mediante sistemas de tiempo real (RTOS, Sistemas Operativo de tiempo real por sus siglas en inglés).

A continuación, una comparación entre hardware y software de dichas plataformas.

Tabla 2. Comparación entre hardware y software de las plataformas arduino y raspberry pi Fuente: Elaboración propia

	Arduino	Raspberry Pi 4
Procesador	Atmega328 Microcontrolador a 16Mhz	Procesador Broadcom BCM2711, cuatro núcleos Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz.
Tamaño	7.6 x 1.9 x 6.4 cm	8.6cm x 5.4cm x 1.7cm
Memoria	0.002MB	512MB
Velocidad de reloj	16 MHz	700 MHz
On Board Network	Ninguna	10/100 wired Ethernet RJ45
Multitarea	No	Si
Voltaje de entrada	7 a 12 V	5 V
Memoria Flash	32KB	Tarjeta SD (2 a 16G)
Puertos USB	Uno	Dos
Sistema operativo	RTOS	Distribuciones de Linux
Entorno de desarrollo integrado (IDE)	Arduino	Scratch, IDLE, cualquiera con soporte Linux
Cantidad de pines digitales	14	24 pines de entrada y salida digital a 3.3V
Cantidad de pines Analógico	**6**	**0**
Precio en dólares	$32	$96

Como se observa la tarjeta de Arduino se hace más viable debido a que el precio en el mercado internacional de la tarjeta de Raspberry pi es el triple de su valor. Arduino es más flexible para el trabajo con sensores analógicos producto a que su placa tiene incorporado pines analógicos; esta flexibilidad le permite trabajar con casi cualquier tipo de sensor, mientras que la Raspberry pi para la lectura de los sensores analógicos requiere la asistencia de un hardware adicional.

El IDE Arduino es simple de usar en comparación con el entorno de desarrollo que emplea la Raspberry pi.

1.3 Comunicación serie

La comunicación serie, es el proceso de envío de datos de un bit por vez, secuencialmente, sobre un canal de comunicación. La comunicación serial es un protocolo muy común entre dispositivos que se incluye de manera estándar en prácticamente cualquier computadora (Weis, 2020).

El puerto serial envía y recibe bytes de información un bit a la vez. Siendo más lento que la comunicación en paralelo, que permite la transmisión de un byte completo por vez, este método de comunicación es más sencillo y puede alcanzar mayores distancias.
Para realizar la comunicación se utilizan 3 líneas de transmisión:

- Tierra (o referencia).
- Transmitir (Tx).
- Recibir (**Rx**).

Debido a que la transmisión es asincrónica, es posible enviar datos por una línea, mientras se reciben datos por otra.
La ventaja de la comunicación serie es que necesita un número más pequeño de líneas de transmisión que una comunicación paralela que transmita la misma información. Esta última necesita tantas líneas de transmisión como la cantidad de bits que componen la información, mientras que la primera se puede llevar a cabo con una sola línea de transmisión.

Ejemplos de comunicación serial:

- **RS-232**: Norma eléctrica que especifica las características de la comunicación serie de baja velocidad.
- **RS-485/422**: Norma eléctrica que especifica las características de la comunicación serie de baja velocidad, implementado en las comunicaciones industriales para lograr una mayor reducción del ruido en la transmisión.

Ambas normas eléctricas especifican una serie de parámetros que deben ser definidos de igual forma en los dispositivos que se van a comunicar. es el usuario quien tiene que decidirlo y configurar ambas partes. Dichos parámetros son:

- **Velocidad de transmisión (baud rate):** Indica el número de bits por segundo que se transfieren, y se mide en baudios (bauds). Las velocidades de transmisión más comunes para las líneas telefónicas son de 14400, 28800, y 33600.

- **Bits de datos:** Se refiere a la cantidad de bits en la transmisión. Cuando la computadora envía un paquete de información, el tamaño de ese paquete no necesariamente será de 8bits. Las cantidades más comunes de bits por paquete son 5, 7 y 8 bits. El número de bits que se envía depende del tipo de información que se transfiere.

- **Bits de parada:** Usado para indicar el fin de la comunicación de un solo paquete. Los valores típicos son 1, 1.5 o 2 bits. Dan un margen de tolerancia para la diferencia en los relojes. Mientras más bits de parada se usen, mayor será la tolerancia a la sincronía de los relojes, sin embargo, la transmisión será más lenta.

- **Bit de Paridad:** Con este bit se pueden descubrir errores en la transmisión. Se puede dar paridad par o impar. En la paridad par, por ejemplo, la palabra de datos a transmitir se completa con el bit de paridad de manera que el número de bits 1 enviados es par (Weis, 2020).

1.4 Sensores en las estaciones de meteorología

La selección de los sensores es una tarea que define la calidad, así como el tipo de estación meteorológica que deseamos construir. Los sensores más utilizados en una estación de meteorológica automática son: temperatura del aire (ºC), humedad relativa del aire (%HR), radiación Solar (w/m2), precipitación (mm), velocidad del viento (m/s), dirección del viento (grados) (F., 2018).

Un sensor se define a menudo como un dispositivo que recibe y responde a un signo o el estímulo (Fraden, 2010).

Cuando se diseñan sistemas de adquisición de datos con computadora, hay aspectos a cerca de los sensores que es necesario tener en cuenta:

- La naturaleza de la señal que el sensor genera: voltaje, rango de amplitud, respuesta en frecuencia, precisión necesaria, determinan el tipo de acondicionamiento de señal, convertidor A/D y cualquier otro hardware a utilizar.
- La influencia de las señales de ruido, así como los efectos de carga del hardware de adquisición de datos sobre el sensor.
- La calibración del sensor con respecto a la variable física. Si la respuesta del sensor a los cambios de la variable física es lineal o no. Una calibración mal hecha va a producir mediciones erróneas.
- La interdependencia entre los distintos componentes del sistema de adquisición de datos, por ejemplo, un sensor muy bueno, con un pobre convertidor A/D no sirve de casi nada. La precisión del sensor, esto es la capacidad de medir el mismo valor repetidas veces en idénticas condiciones.
- El tiempo de respuesta del sensor, es decir, el tiempo requerido para responder a un cambio brusco de la variable que está siendo censada.
- El coeficiente de temperatura del sensor, el cual viene dado por el cambio que se produce en la respuesta del sensor debido al cambio en la temperatura a la cual se encuentra, por ejemplo: el aumento en las corrientes de fuga y el voltaje offset de un amplificador, el aumento de la corriente en la oscuridad de un fotodiodo.
- La histéresis de un sensor, la cual se define como la dependencia de la salida del sensor de la respuesta anterior. Esta es muy común en sistemas magnéticos y mecánicos.

Existen varias formas de clasificar los sensores, por ejemplo, se pueden clasificar por el principio físico de funcionamiento (inductivo, capacitivo, termoeléctrico o resistivo), por la variable física medida (temperatura, presión, posición por la capacidad de generar energía (activos) o de necesitar de un circuito de excitación (pasivos) (Mayné, 2003).

Para el diseño de la estación de meteorológica se escogieron los siguientes sensores:

Temperatura y humedad relativa del aire EE181.

Es un sensor robusto y preciso para medir la temperatura y humedad relativa del aire, ideal para aplicaciones desatendidas a largo plazo. Dispone de una protección especial en el elemento sensor de la humedad relativa que aumenta la vida del elemento sensor protegiéndolo de la suciedad, polvo, sal y otros contaminantes. La medida de la temperatura del aire la realiza una pt1000 en el rango de -40°C a +60°C. Para óptimos resultados, el EE181 debería recalibrarse anualmente. (Campbell scientific,

2023). Como accesorio lleva un protector para radiación con ventilación natural modelo URS1, se puede observar en la figura 4.

Figura 4 Temperatura y humedad relativa del aire EE181 (Campbell scientifico, 2023).

Conjunto de sensores meteorológicos **p/n 80422:**

Incluye un anemómetro y veleta y un pluviómetro. Incluye un hardware de montaje y conexiones RJ11 a los sensores. El anemómetro usa un interruptor de láminas, así que se puede usar una simple detección de frecuencias para medir la velocidad del viento. La veleta usa un potenciómetro para detectar la dirección del viento (MCI electronics, 2023).

Figura 5 Sensor de velocidad y dirección del viento. (Arduino, 2023)

1.5 Diseño del prototipo del hardware

Una vez elegidos la plataforma Arduino y los sensores para la estación meteorológica, se diseñó la placa del circuito impreso El plano en el cual se muestran todos los componentes significativos de un circuito y sus conexiones se puede observar en Anexo 2 A continuación, se muestra la vista 2D del PCB para la estación de meteorológica.

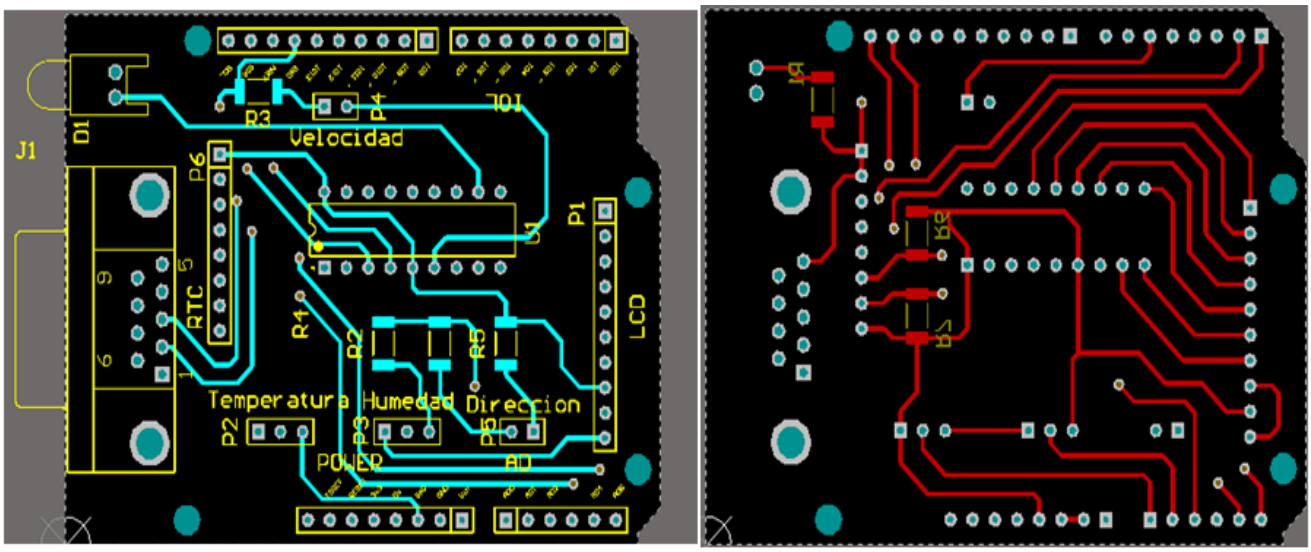


Figura 6 Vista del PCB de la tarjeta complementaria del Arduino Fuente: elaboración propia.

1.6. Metodología de desarrollo

La metodología de desarrollo es un marco de trabajo que parte de una posición teórica y conlleva a una selección de técnicas concretas o métodos acerca del procedimiento para el cumplimiento de los objetivos. Es el conjunto de métodos que se utilizan en una determinada actividad con el fin de formalizarla y optimizarla. Determina los pasos a seguir y que hacer para realizarlos y finalizar una tarea. Estas se dividen en dos enfoques o ramas, las cuales son conocidas como metodologías ágiles y metodologías tradicionales. Las tradicionales centran su atención en llevar una documentación exhaustiva de todo el proyecto, la planificación y control del mismo, en especificaciones precisas de requisitos y modelado y en cumplir con un plan de trabajo, definido todo esto en la fase inicial del desarrollo del proyecto. Por otro lado, las ágiles son convenientes para guiar proyectos de escaso volumen comercial que demanden una rápida implementación (Maida, 2015).Tiene como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo, haciendo énfasis en la calidad y menor tiempo de construcción del software.

Para elegir una metodología de desarrollo de software se debe tener en cuenta dos factores fundamentales: el tipo de proyecto que se desea desarrollar y el tiempo que se dispone para desarrollar el mismo. En la actualidad no se puede afirmar que existe una metodología que funcione de manera universal, son más bien concebidas como marcos metodológicos que deben ajustarse a cada organización y tipo de proyecto a desarrollar, por lo que existen dos grandes enfoques de metodologías: las tradicionales y las ágiles. En la siguiente tabla se presenta una comparación entre los dos tipos de metodologías.

Tabla 3 Comparación entre las metodologías ágiles y tradicionales Fuente :elaboración propia.

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

Teniendo en cuenta que es un proyecto pequeño, el tamaño del equipo de un solo integrante y son necesarias las prácticas de ingeriría para poder documentar todo el proceso de desarrollo de la solución, se determina como metodología de desarrollo se software XP (Extreme Programming). Entre las ventajas de esta metodología es la disminución del tiempo de desarrollo sin que se afecte la calidad del producto. Además, genera poca cantidad de artefactos y se encuentra enfocada principalmente al código

1.6.1. Metodología de desarrollo de software Programación Extrema.

La programación extrema (XP) es una metodología ágil de gestión de proyectos que se centra en la velocidad y la simplicidad con ciclos de desarrollo cortos. Los objetivos de XP son muy simples: la satisfacción del cliente. Esta metodología trata de dar al cliente el software que él necesita y cuando

lo necesita. Por tanto, debemos responder muy rápido a las necesidades del cliente, incluso cuando los cambios sean al final de ciclo de la programación.
El segundo objetivo es potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software (Solís, 2003).

Características de la Metodología XP

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. XP es una metodología ágil para pequeños y medianos equipos, desarrollando software cuando los requerimientos son ambiguos o rápidamente cambiantes. Su ciclo de vida se basa en seis fases: Exploración, Planificación, Iteración, Producción, Mantenimiento y Muerte. Además, consta de cuatro fases fundamentales: Planeación, Diseño, Desarrollo y Pruebas.

- **Planeación:** La planeación es la etapa inicial de todo proyecto en XP. En este punto se comienza a interactuar con el cliente y el resto del grupo de desarrollo para descubrir los requerimientos del sistema. En este punto se identifican el número y tamaño de las iteraciones al igual que se plantean ajustes necesarios a la metodología según las características del proyecto.
- **Diseño:** En XP solo se diseñan aquellas historias de usuario que el cliente ha seleccionado para la iteración actual por dos motivos: por un lado, se considera que no es posible tener un diseño completo del sistema y sin errores desde el principio. El segundo motivo es que, dada la naturaleza cambiante del proyecto, el hacer un diseño muy extenso en las fases iniciales el proyecto para luego modificarlo, se considera un desperdicio de tiempo.
- **Desarrollo:** El desarrollo es un proceso que se realiza en forma paralela con el diseño y la cual está sujeta a varias observaciones por parte de XP consideradas controversiales por algunos expertos tales como la rotación de los programadores o la programación en parejas.
- **Pruebas:** Cuando se tienen bien implementadas las pruebas no habrá temor de modificar el código del otro programador en el sentido que, si se daña alguna sección, las pruebas mostrarán el error y permitirán encontrarlo. Uno de los elementos que podría obstaculizar que

un programador cambie una sección de código funcional es precisamente hacer que este deje de funcionar. Si se tiene un grupo de pruebas que garantice su buen funcionamiento, este temor se mitiga en gran medida.

1.7 Herramientas y tecnologías seleccionadas

A continuación, se muestra el conjunto de herramientas y tecnologías que fueron seleccionadas para el desarrollo de la solución propuesta

Lenguaje de Modelado UML 2.1.

Es un lenguaje para especificar, construir, visualizar y documentar los artefactos de un sistema de *software* orientado a objetos. Para realizar los modelos del sistema propuesto se hará uso del Lenguaje Unificado de Modelado 2.1 (*UML*, por sus siglas en inglés), porque tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos, hasta la implementación y configuración con los diagramas de despliegue (Larman, 2003).

Herramienta CASE:

Las herramientas de Ingeniería de Software Asistida por Ordenador (CASE Ingeniería de software asistida por computadora, Computer Aided Software Engineering) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas ayudan al proceso de desarrollo del software en tareas como: realizar un diseño del proyecto, cálculo de costos, implementación de parte del código con el diseño dado, compilación automática, documentación o detección de errores entre otras (Slavkova, 1993).

Visual Paradigma for UML 5.0.

Constituye una herramienta CASE profesional, que utiliza UML (Lenguaje Unificado de Modelado por sus siglas en inglés *Unified Modeling Language*)[1] como lenguaje de modelado y que soporta el ciclo de vida completo del desarrollo de software, además de tener la ventaja de ser multiplataforma (Gaines, y otros, 2017).

[1] UML *(Unified Modeling Language)*: Lenguaje Unificado de Modelado que propicia un conjunto de ayudas para el desarrollo de programas informáticos.

Esta herramienta propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (Pressman, 2002). La UCI cuenta con licencia para su uso, por lo que se contribuye al cumplimiento de las políticas de migración hacia software libre que sigue el país. A continuación, se listan algunas de sus características:

- Ofrece amplias características de modelado de casos de uso incluyendo la función completa de UML, diagrama de casos de uso y editor de flujo de eventos.
- Permite diseñar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Proporciona abundantes tutoriales del Lenguaje de Modelado, demostraciones interactivas de UML y proyectos UML.
- Se integra con herramientas Java, como son: Eclipse/IBM, NetBeans IDE, entre otras.
- Es apoyado por un conjunto de lenguajes tanto en la generación del código como en la Ingeniería Inversa (Gaines, y otros, 2017).

Una vez analizados los elementos expuestos anteriormente y que la UCI posee una licencia académica para su uso se decide por parte del equipo de arquitectura del proyecto utilizar Visual Paradigm en su versión 8.0, teniendo en cuenta que esta herramienta ofrece funcionalidades para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.

1.8 Lenguaje de programación

Java: es una plataforma informática de lenguaje de programación creada por Sun Microsystems en 1995. Ha evolucionado desde sus humildes comienzos hasta impulsar una gran parte del mundo digital actual, ya que es una plataforma fiable en la que se crean muchos servicios y aplicaciones. Los nuevos e innovadores productos y servicios digitales diseñados para el futuro también siguen basándose en Java (ORACLE, 2023).

Características:

Se constituye como un lenguaje orientado a objetos, su intención es permitir que los desarrolladores de aplicaciones escriban el programa una sola vez y lo ejecuten en cualquier dispositivo. Para comprender qué es Java es necesario definir las características que lo diferencian de otros lenguajes de programación.

- Es simple Java ofrece la funcionalidad de un lenguaje potente, derivado de C y C++, pero sin las características menos usadas y más confusas de estos, haciéndolo más sencillo.
- Orientado a objetos El enfoque orientado a objetos (OO) es uno de los estilos de programación más populares. Permite diseñar el software de forma que los distintos tipos de datos que se usen estén unidos a sus operaciones.
- Es distribuido Java proporciona una gran biblioteca estándar y herramientas para que los programas puedan ser distribuidos.
- Independiente a la plataforma: Esto significa que programas escritos en el lenguaje Java pueden ejecutarse en cualquier tipo de hardware, lo que lo hace portable.
- Recolector de basura: Cuando no hay referencias localizadas a un objeto, el recolector de basura de Java borra dicho objeto, liberando así la memoria que ocupaba. Esto previene posibles fugas de memoria.
- Es seguro y sólido: Proporcionando una plataforma segura para desarrollar y ejecutar aplicaciones que, administra automáticamente la memoria, provee canales de comunicación segura protegiendo la privacidad de los datos y, al tener una sintaxis rigurosa evita que se quiebre el código, es decir, no permite la corrupción del mismo.
- Es multi hilo: Java logra llevar a cabo varias tareas simultáneamente dentro del mismo programa. Esto permite mejorar el rendimiento y la velocidad de ejecución (ORACLE, 2023).

1.9. Entorno Integrado de Desarrollo (IDE)

Un entorno de desarrollo integrado, es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, PHP, Python, Java, C#, Delphi, Visual Basic. Algunos ejemplos de IDE son los siguientes: PhpStorm, Eclipse, Qtcreator, NetBeans, Visual C++, Arduino. (Gomes, 2022)

A continuación, se hace una breve descripción de los IDE que serán usados para el desarrollo del sistema a implementar.

- **Arduino-2.1.0**: Es un entorno integrado de desarrollo para el software de código abierto, este se ejecuta en Windows, Mac OS X y Linux. El entorno está escrito en Java y está basado en Processing y otro software de código abierto, este se puede usar con cualquier placa Arduino. Los programas de Arduino están compuestos por un solo fichero con extensión “**ino**”, aunque

es posible organizarlo en varios ficheros. La principal característica del (IDE) y del lenguaje de programación es su sencillez y facilidad de uso (Arduino, 2023).

- **NetBeans** es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. Inicialmente desarrollado por Sun pero se ha integrado como proyecto en la fundación Apache (Fernandez, 2023).

1.9 Tipos de sistemas gestores de bases de datos

Como elemento importante de esta investigación, se realizará un estudio y caracterización de algunos sistemas gestores de bases de datos para posteriormente seleccionar el más indicado a utilizar en la propuesta de solución, dentro de los que se encuentran los siguientes (marin, 2019).

- **MySQL**: es un sistema de gestión de base de datos, multihilo y multiusuario seguramente el más usado en aplicaciones creadas como *software* libre. Por un lado, se ofrece bajo la GNU GPL[2], pero, empresas que quieran incorporarlo en productos privativos pueden comprar a la empresa una licencia que les permita ese uso.Entre las ventajas que ofrecen se encuentra:
  - ✓ Velocidad al realizar las operaciones.
  - ✓ Bajo costo en requerimientos para la elaboración de bases de datos.
  - ✓ Facilidad de configuración e instalación.
- **PostgreSQL**: es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD. Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y apoyada por organizaciones comerciales. La comunidad PostgreSQL se denominada el *PGDG* (PostgreSQL Global Development Group). Sus principales características son:
  - ✓ Alta concurrencia: mediante un sistema denominado *MVCC* (Acceso concurrente multi versión, por sus siglas en inglés).
  - ✓ Amplia variedad de tipos nativos: provee nativamente varios soportes.
  - ✓ Ahorros considerables de costos de operación.
  - ✓ Estabilidad y confiabilidad.

[2] Licencia Pública General es la licencia de derecho de autor más ampliamente usada en el mundo del software libre y código abierto, y garantiza a los usuarios finales (personas, organizaciones, compañías) la libertad de usar, estudiar, compartir (copiar) y modificar el software.

1.9.1 Selección del sistema gestor de base de datos

Después de haber analizado las características de los diferentes sistemas gestores de bases de datos debido a que es necesario almacenar la información de todas las personas autorizadas en las diferentes áreas o locales en el sistema que se va a desarrollar, se decidió usar en PostgreSQL en su versión 9.6.8 pues tiene un alto rendimiento probado, fiabilidad, alta velocidad para consultas y facilidad de uso con el servidor apache.

PgAdmin 4

PgAdmin es una herramienta indispensable para gestionar y administrar PostgreSQL, la base de datos de código abierto más avanzada del mundo. Por lo tanto, pgAdmin es la herramienta para gestionar nuestras bases de datos espaciales PostGIS.El software tiene la apariencia de una aplicación de escritorio sea cual sea el entorno de tiempo de ejecución (escritorio o web), y mejora enormemente respecto a pgAdmin III con elementos de interfaz de usuario actualizados, opciones de despliegue multiusuario / web, paneles y un diseño más moderno (Morales, 2020).

1.10 Conclusiones parciales

En el desarrollo del capítulo se obtuvo una mejor visión acerca del problema planteado y se dieron cumplimiento a las primeras tareas de investigación llegando a las siguientes conclusiones:

- ✓ El análisis de los conceptos asociados a la solución permitió un acercamiento a los temas relacionados con el desarrollo de una estación de meteorología
- ✓ La selección de las herramientas y tecnologías libres a utilizar como: Arduino 2.1.0, los lenguajes de programación Arduino y Java definen las bases para el diseño e implementación satisfactoria de la estación meteorológica a desarrollar.
- ✓ El proyecto tiene las características de tener un tiempo de desarrollo corto y la necesidad de realizar entregas constantes en ciclos cortos, por lo que es necesario en uso de una metodología ágil. Luego de este análisis, se decide utilizar la metodología XP, permitiendo esto cierta flexibilidad a lo largo del desarrollo de la aplicación y una interacción con el usuario.

CAPÍTULO II: ANÁLISIS Y DISEÑO DE LA SOLUCIÓN PROPUESTA

En este capítulo se describirá la propuesta de solución mediante el análisis y diseño del sistema, tomando como referencia los conceptos y términos definidos en el primer capítulo, y regido por la metodología de desarrollo de software XP. Se enunciarán los requisitos funcionales y no funcionales y la arquitectura que será seleccionada, así como los artefactos generados en la etapa correspondiente al modelado del sistema.

2.1 Descripción de propuesta de solución

Después de haber realizado un análisis de los principales conceptos y herramientas que serán utilizados durante la presente investigación, se puede observar la propuesta de solución en la figura 7 la misma se define de la siguiente manera.

Mediante una placa Arduino se obtendrán los valores equivalentes a parámetros ambientales obtenidos en los correspondientes sensores de temperatura, humedad, velocidad y dirección del viento; dichas magnitudes serán representados en una pantalla de cristal líquido de 20*4 en conjunto con valores correspondiente a la fecha y hora en que fueron registrados, producto al dato entregado por un reloj de tiempo real. Arduino realizara un paquete de datos con la elaboración de un reporte que envía a través de una comunicación serial RS232 hacia un modem conversor de serial a radio frecuencia, transmitiéndose los datos hacia un cliente remoto.

El cliente remoto tiene el mismo modem, pero con la función de convertir la comunicación inalámbrica en una comunicación serial RS232 y transmitirlo a una computadora personal con la aplicación de escritorio elaborada en el lenguaje de programación java para la representación y almacenamiento de los parámetros meteorológicos con ayuda del gestor de base de datos **PostgreSQL**.

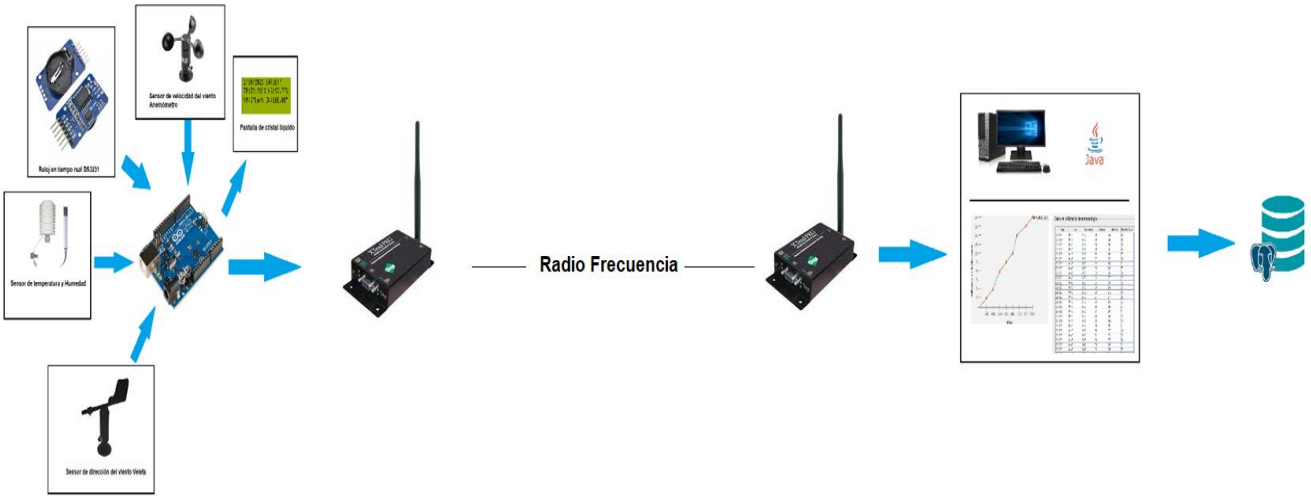


Figura 7 Esquema de la propuesta de solución (Elaboración Propia).

2.2 Requisitos no funcionales

Los Requerimientos No Funcionales (RNF), como su nombre sugiere, son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa definen las restricciones del sistema (Sommerville, 2005). Los requisitos no funcionales de la aplicación a desarrollar son:

Requisitos no funcionales de Usabilidad

RNF 01. La aplicación debe mostrar la información legible para el usuario.

RNF 02. La aplicación debe permitir el acceso a más de un dispositivo.

Requisitos no funcionales de Apariencia o interfaz externa

RNF 03. La aplicación debe visualizar los gráficos en 2D.

Requisitos no funcionales de Software

RNF 04. La aplicación debe ser independiente de la plataforma.

Requisitos no funcionales de Restricciones del diseño y la implementación

RNF 05. La aplicación debe emplear NetBeans IDE 8.2 como editor de código.

RNF 06. La aplicación debe emplear java como lenguaje de programación.

RNF 07. La aplicación de la estación meteorológica debe emplear Arduino IDE como editor de código

en su versión 1.8.19.

RNF 08. La aplicación debe emplear Visual Paradigm en su versión 8.0 como herramienta CASE.

2.3 Requerimientos funcionales

Los requisitos básicamente son aquellas funcionalidades que deben cumplir los sistemas basados en las necesidades del cliente. A continuación, se describen los requisitos asociados a la propuesta de solución. Que se dividen en dos una para la estación meteorológica y otro para aplicación que va almacenar y mostrar los datos recibidos. Como bien se define la metodología (XP), una de las formas de encapsular los requisitos funcionales son las historias de usuario(HU). A continuación, se presentan las HU desarrolladas. Según el cliente.

Para la estación meteorológica:

- ✓ HU1: Medir los datos de los sensores de humedad, temperatura, velocidad y dirección del viento en un tiempo determinado**.**
- ✓ HU2: Mostrar los parámetros atmosféricos en conjunto con la fecha y hora.
- ✓ HU3: Permitir la comunicación serial de datos.

Para la aplicación:

- ✓ HU4: Activar la comunicación con el Arduino.
- ✓ HU5: Almacenar los datos.
- ✓ HU6: Representar los datos

2.4 Descripción de las historias de usuarios

Las historias de usuarios permiten encapsular los documentos de especificación funcional. Estas HU son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar (María José Pérez Pérez, 2018). Las principales características de las historias de usuario son según independientes unas de otras, negociables, valoradas por los clientes o usuarios, estimables, pequeñas y verificables.

Para la elaboración de las historias de usuario se realizaron diferentes entrevistas con los miembros a miembros del MINAG[3], ANAP[4] como posibles clientes. Si bien el cliente no fue quien escribió perso-

[3] Acrónimo El Ministerio de Agricultura de la República de Cuba

[4] Asociación Nacional de Agricultores Pequeños es una organización de carácter social que representa los intereses del campesinado cubano y que vela porque se cumplan sus derechos

nalmente las HU, fue quien esbozó su contenido, asignó la prioridad y tuteló la redacción de cada una de ellas, por lo que su redacción fue clara y breve.
El cliente asignó la prioridad de cada una teniendo en cuenta la necesidad de su pronta implantación y su criticidad en el flujo de eventos, mientras el equipo de desarrollo revisó la prioridad, analizó la dependencia entre ellas y asignó el costo de cada una, este último se traduce en las semanas para su desarrollo. Se sugiere dividir las HU en más pequeñas si estas se demoran más tiempo en desarrollarse de lo planificado.

Las HU se representan mediante tablas, que contienen las siguientes secciones:

- ✓ Código: las siglas de HU más un número consecutivo, este permite identificar la historia.
- ✓ Nombre: nombre que la identifica.
- ✓ Referencia: es el conjunto de códigos de las diferentes HU de las cuales depende la que actualmente se encuentra en desarrollo.
- ✓ Prioridad: esta característica es dada por el cliente con los valores: alta, media o baja en dependencia de la importancia y orden en que desean que sean implementadas.
- ✓ Iteración Asignada: número de la iteración en la que se desarrollará la HU.
- ✓ Puntos Estimados: tiempo estimado en semanas que se le asignará.
- ✓ Descripción: breve descripción del proceso que define la historia.

Se elaboraron 6 HU, y se tomaron para los puntos estimados, la unidad como una semana de trabajo. Se confeccionaron todas con prioridad alta.

Tabla 4 HU1:Medir los datos de sensores.

Historia de Usuario	
Código: HU1	Nombre: Medir los datos de los sensores
Referencia: Ninguna	Prioridad: Alta
Iteración asignada: 1	Puntos Estimados: 1 semana

Descripción: El Arduino realizara la lectura los niveles de voltaje en sus pines de entrada en correspondencia con la conexión de los sensores. Realizando la conversión de voltaje a magnitudes física según las características del sensor correspondientes.
Observaciones: ninguna.
Prototipo de interfaz de usuario: ninguna.

Tabla 5 HU2 Mostrar los parámetros atmosféricos en conjunto con la fecha y hora.

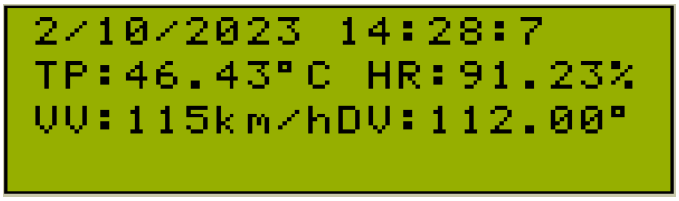
Historia de Usuario	
Código: HU2	Nombre: Mostrar los parámetros atmosféricos en conjunto con la fecha y hora.
Referencia: Depende de la HU1	Prioridad: Alta
Iteración asignada: 1	Puntos Estimados: 1 semana
Descripción: Para la representación de datos se empleará una pantalla de cristal líquido(LCD) y se representaran los datos en el siguiente orden fecha, hora, temperatura, humedad, velocidad.	
Observaciones:	
Prototipo de interfaz de usuario: 	

Tabla 6 HU3 Permitir la comunicación serial de datos.

Historia de Usuario	
Código: HU3	Nombre: Permitir la comunicación serial de datos.
Referencia: depende de la HU1	Prioridad: Alta
Iteración asignada: 1	Puntos Estimados: 1 semana
Descripción: se configura la comunicación serial con la Interfaz RS232 para la trasmisión de los valore de las magnitudes físicas adquirida por el Arduino donde los datos serán transmitidos en una estructura de arreglo de una dimensión de 6 elemento y la asignación de	

cada elemento seria la siguiente: A[0] =fecha, A[1] =hora A[2] =temperatura A[3]=Humedad, A[4] =velocidad, A[0] = dirección del viento.
Prototipo de interfaz de usuario: ninguna

Tabla 7 HU4 Activar la comunicación con el Arduino.

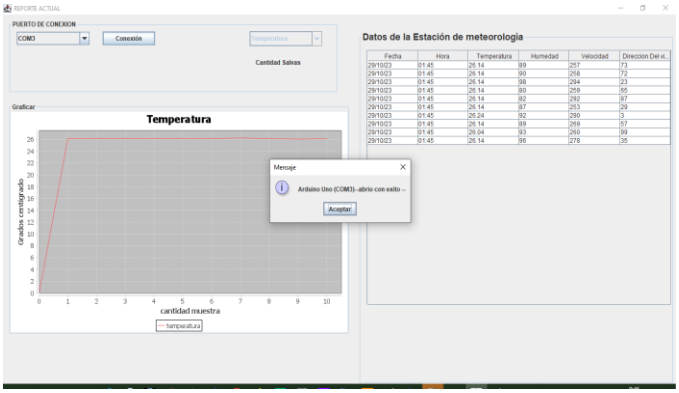
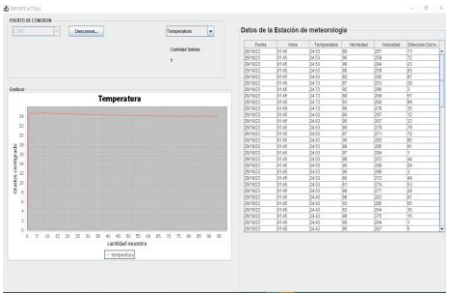
Historia de Usuario	
Código: HU4	Nombre: Activar la comunicación con el Arduino.
Referencia: 3	Prioridad: Alta
Iteración asignada: 2	Puntos Estimados: 1 semana
Descripción: El sistema debe permitir la comunicación y la sincronización de los datos recibidos. Una vez seleccionado el puerto COM.	
Prototipo de interfaz de usuario: 	

Tabla 8 HU5 Almacenar los datos.

Historia de Usuario	
Código: HU5	Almacenar los datos .
Referencia: depende de la HU4	Prioridad: alta
Iteración asignada: 3	Puntos Estimados: 1 semana
Descripción: Debe permitir que los datos recibidos de almacenen en una base de dato de forma automática cada un minuto	
Prototipo de interfaz de usuario: ninguna	

Tabla 9 HU6 Representar los datos

Historia de Usuario	
Código: HU6	Nombre : Representar los datos
Referencia: depende de la HU4	Prioridad: Alta
Iteración asignada: 4	Puntos Estimados: 1 semana
Descripción: Debe representar los datos recibidos en una tabla según parámetros y representarlos gráficamente por parámetros independientes	
Prototipo de interfaz de usuario: 	

2.5 Plan de iteraciones

Después de identificar y redactar cada una de las HU, se debe elaborar el plan de iteraciones. Cada HU se convierte en tareas específicas de programación, de la misma forma para cada HU se establecen las pruebas de aceptación. Las pruebas fallidas en el ciclo anterior son analizadas para evaluar su corrección para prever que no vuelvan a ocurrir.

El plan fue dividido en 4 iteraciones, para las que se la estación de meteorología. A continuación, se describen cada una de las iteraciones anteriormente mencionadas:

- ✓ **Iteración 1:** para esta iteración se desarrollan las HU1,2,3 correspondiente a la medición y visualización de los datos con su fecha y hora de adquisición; y con la configuración de la comunicación serial. Donde se hace la selección de los sensores y los dispositivos de hardware necesarios para la creación del prototipo de estación meteorológica; dando la funcionabilidad con la programación del software. Para la implementación de las HU2 y HU3 se apoyan en la implementación de la HU1.Se realizan las pruebas de aceptación haciendo una entrega funcional al finalizar la Iteración.

- ✓ **Iteración 2**: para esta iteración se desarrolla la HU4 que corresponde con Activar la comunicación con el Arduino donde el sistema debe permitir la comunicación y la sincronización de los datos recibidos, una vez seleccionado el puerto COMM.

  Se ejecutar las pruebas de aceptación y realizan pruebas de integración para verificar que no han ocurrido afectaciones en las historias anteriores. Al finalizar la iteración se realiza una entrega funcional.
- ✓ **Iteración 3:** para esta iteración se desarrolla el HU5 que corresponde con almacenar los datos; para ello va depender de la ejecución del caso de HU4 una vez leído el dato en el puerto serial procede con la validación del mismo y el posterior almacenamiento en un periodo de tiempo de un minuto. En esta iteración al igual que en la anterior, se realizan las pruebas de aceptación e integración y al finalizar la iteración se debe obtener un sistema funcional que almacena los valores de las magnitudes físicas del clima.
- ✓ **Iteración 4:** en esta interacción se desarrolla la HU6 que corresponde con la representación del dato leído; utilizando una tabla para representar todas las variables meteorológicas y una gráfica de puntos para mostrar los parámetros según el tiempo de lectura, la implementación de esta HU va depender de la ejecución de la HU4.

  Se realizaron las pruebas de aceptación e integración y al finalizar la iteración se realiza la entrega final de la propuesta de solución.

En cada iteración, para aproximar el tiempo de ejecución se tomó como medida, cada semana contaba de 5 días (lunes, martes, miércoles, jueves y viernes) en los que se trabajaban 8 horas. En la siguiente tabla, se muestra el plan de iteraciones y se incorpora el tiempo estimado para cada una de las iteraciones y las HU que se van a desarrollar.

Tabla 10 Plan de iteraciones

Iteración	Historia de usuario	Puntos de estimación
1	Medir los datos de los sensores de humedad, temperatura, velocidad y dirección del viento en un tiempo determinado	3 semanas
	Mostrar los parámetros atmosféricos en conjunto con la fecha y hora	
	Permitir la comunicación serial de datos	
2	Activar la comunicación con el Arduino	1 semana

3	Almacenar los datos	1 semana
4	Representar los datos	1 semana

2.6 Plan de entregas

El plan de entregas establece que HU son agrupadas para conformar una entrega, y el orden de las mismas. Este plan es el resultado de una reunión entre todos los actores del proyecto (cliente, desarrolladores y gerentes). El cronograma se realiza sobre la base de las estimaciones de tiempos de desarrollo realizadas por los desarrolladores.

A partir del plan de iteraciones analizado en el acápite anterior, y en correspondencia con el mismo se realiza el plan de entregas, que se muestra en la siguiente tabla. En él, se proponen tres versiones funcionales y una última entrega de iteraciones de la propuesta de solución.

Tabla 6 Plan de entrega

Historias de usuario	Primera Iteración 2/10/2023	Segunda Iteración 23/10/2023	Tercera Iteración 30/10/2023	Cuarta Iteración 6/11/2023
HU1,2,3	V1.0			
HU4		V1.1		
HU5			V1.2	
HU5				V1.3

2.7 Diseño arquitectónico

Durante el desarrollo de software es necesario realizar la selección de patrones que permitan obtener una buena estructura y organización del código de la aplicación, pues de esta manera, se garantiza la eficiencia en el funcionamiento del sistema. Estos patrones no son más que una guía para realizar alguna acción. Especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes. Comprenden el diseño de más alto nivel de la estructura del sistema, expresan un esquema organizativo y estructural para sistemas de software, lo que posibilita un mejor entendimiento del problema que se le quiera dar solución *(Wilson, 2020)*.

La arquitectura del software alude a la estructura global del software y a las formas en que la estructura proporciona la integridad conceptual de un sistema. En su forma más simple, la arquitectura es la

estructura jerárquica de los componentes del programa, la manera en que los componentes interactúan y la estructura de datos que van a utilizar los componentes. Sin embargo, en un sentido más amplio, los componentes se pueden generalizar para representar los elementos principales del sistema y sus interacciones *(Pressman, 2005)*.

2.7.1. Patrón arquitectónico

En el desarrollo de la estación de meteorológica de manera general se empleó la Arquitectura Cliente Servidor el cual está compuesto por dos componentes, el proveedor y el consumidor. El proveedor es un servidor que brinda una serie de servicios o recursos los cuales son consumido por el Cliente. En este caso la estación meteorológica es el servidor que proporciona los datos a la aplicación. La aplicación solo se encarga de representar y guardar los datos enviados por el Arduino; comportándose como cliente. En la figura 8, se muestra el patrón arquitectónico Cliente Servidor, que se emplea en la aplicación de manera general.

Figura 8 Diagrama cliente servidor de la aplicación Fuente: elaboración propia.

Dentro del servidor Arduino se encuentra la aplicación encargada de la cuantificación de los parámetros meteorológicos; la cual cuenta con un patrón arquitectónico de dos capas (ver ilustración 7): la capa del firmware, la cual posee dos componentes lógicos (Configuración API y la comunicación serial RS232), que tiene como responsabilidad la adquisición de los datos y él envió de los mismo al cliente mediante la interacción con el hardware.

La capa Hardware posibilita las interfaces físicas de los periféricos del Arduino, los cuales tienen la tarea de la lectura de los valores de los sensores y del reloj de tiempo real para sincronizar los datos del firmware y la memoria EEPROM para persistir la configuración en el Arduino.

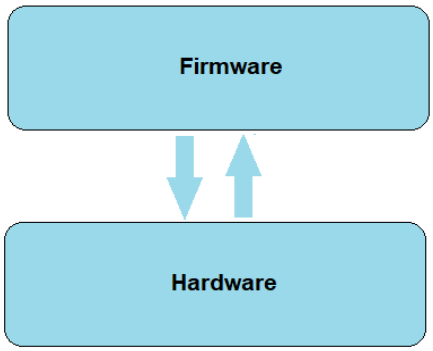


Figura 9 Capas de la estación de meteorológica (Fuente: elaboración propia)

Para el diseño de la aplicación en java se utilizó el patrón arquitectónico modelo vista controlador (MVC) el cual se refleja en la siguiente ilustración:

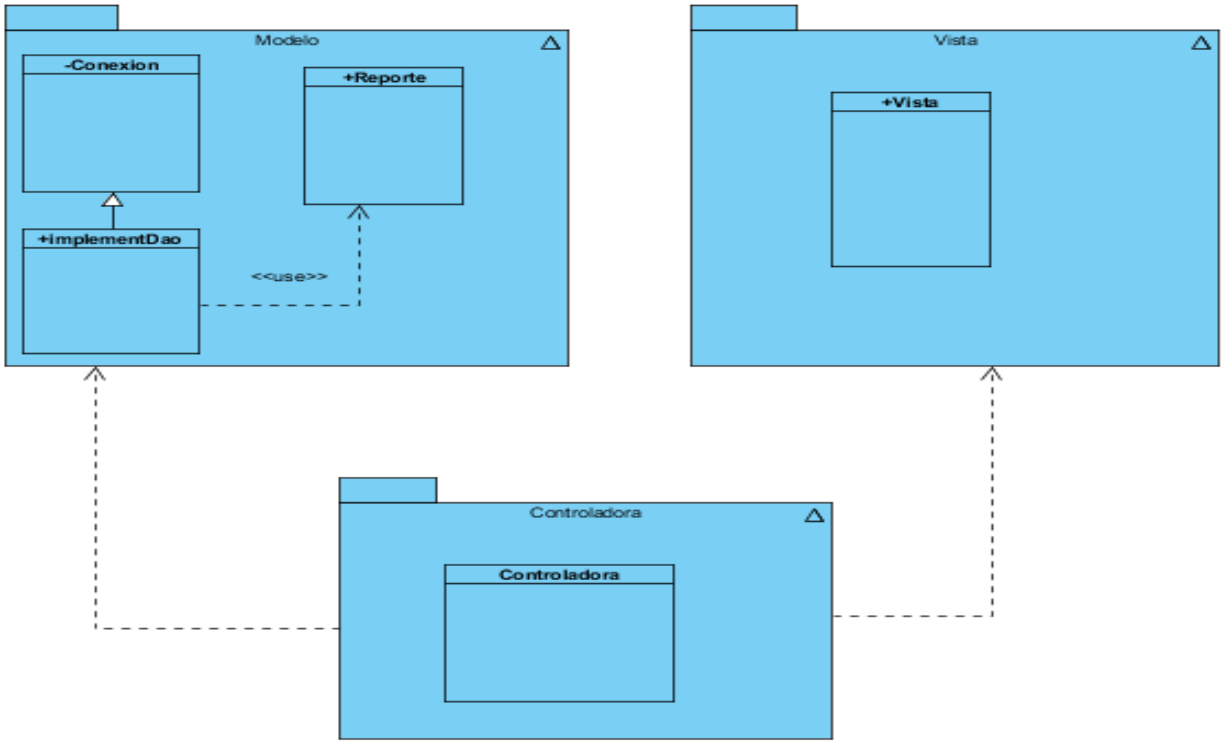


Figura 10 Aplicación de MVC a la aplicación Fuente: elaboración propia.

En el diagrama presentado se muestran los paquetes de para el diseño de la aplicación la misma está compuesta por tres paquetes.

- ✓ La vista define cómo se deben mostrar los datos de la aplicación. Esta capa se encarga de interactuar con el usuario en este caso es la clase **VistaPrincipal** la encargada de realizar la interfaz visual.
- ✓ El controlador contiene una lógica que actualiza el modelo y/o vista en respuesta a las entradas de los datos a la de la aplicación y se implementa las reglas del dominio de la aplicación.
- ✓ Modelo: El modelo es el conjunto de clases que representa la información en esta aplicación la podemos ver con el uso del patrón objeto de acceso a datos (DAO, Data Access Object por sus siglas en inglés) es la encargada de las operaciones de persistencia contra la tabla Reporte de la base de datos.

2.8 Patrones de diseño

Un patrón de diseño se caracteriza como una regla de tres partes que expresa una relación entre cierto contexto, un problema y una solución. Para el diseño de software, el contexto permite al lector entender el ambiente en el que reside el problema y qué solución sería apropiada en dicho ambiente. El patrón de diseño incorpora el conocimiento de diseño pragmático, ganando con dificultad, en una forma que permite que otros lo reutilicen “un millón de veces sin elaborarla dos veces de la misma forma”. Un patrón de diseño evita “reinventar la rueda” o, peor aún, inventar una “nueva rueda” que sea un poco menos redonda, demasiado pequeña para el uso que se pretende y muy angosta para el terreno en el que rodará. Si se usan con eficacia, los patrones de diseño invariablemente harán del lector un buen diseñador de software (Roger S. Pressman, 2015).

Los patrones de diseño son un conjunto de estrategias, o buenas prácticas, que pueden facilitar el trabajo en muchas situaciones a la hora de realizar una aplicación orientada a objetos. Además, son los encargados de identificar clases, instancia, roles, colaboraciones y la distribución de responsabilidades (Erich Gamma, 2010).

Estos ofrecen las siguientes ventajas:

- Proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifica y describe formas de solucionar problemas que ocurren de forma frecuente en el desarrollo. Están basados en la recopilación del conocimiento de los expertos en desarrollo del software.
- Es una experiencia real, probada y funcional.
- Facilitan la localización de objetos que forman el sistema, la determinación de la gradualidad adecuada, el aprendizaje y la comunicación entre programadores.
- Especifican interfaces para las clases e implementaciones al menos parciales.

2.8.1 Patrones GRASP

Los Patrones Generales de Software para Asignación de Responsabilidades (GRASP[5]), describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (Larman, 2003).

Patrones de Asignación de Responsabilidad (GRASP, por sus siglas en inglés General Respo nsibility Assignment Software Patterns) representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones:

- Patrón acoplamiento**:** el acoplamiento es una medida de la fuerza en la que una clase está relacionada a otras clases. Una clase con bajo o débil acoplamiento no depende de muchas otras. Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma, que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, lo que potencia la reutilización, y disminuye la dependencia entre las clases. La solución es asignar las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible, sin comprometer la funcionalidad. La Ilustración 9 representa el patrón antes mencionado en la clase DAOReporteImpl.java ya que utiliza altas de dependencias para el cumplimiento de su función permitiendo el incremento de la reutilización de código.

```
import java.time.LocalTime;
import java.util.ArrayList;
import javax.naming.spi.DirStateFactory;


public class DAOReporteImpl extends Conexion  implements DAOReporte {

    @Override
    public void registrar(Reporte rp) throws Exception {
        try {
            this.Conectar();

    PreparedStatement st = this.conexion.prepareStatement("INSERT INTO \"Reporte\" values (?,?,?,?,?,?) ");//INSERT into

          st.setDate(1,Date.valueOf(rp.getFECHA()));

          st.setTime(2,Time.valueOf(rp.getHORA()));
```

Figura 11 Bajo acoplamiento Fuente elaboración propia.

- Patrón creador: para guiar la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El diseño bien asignado permitirá soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la

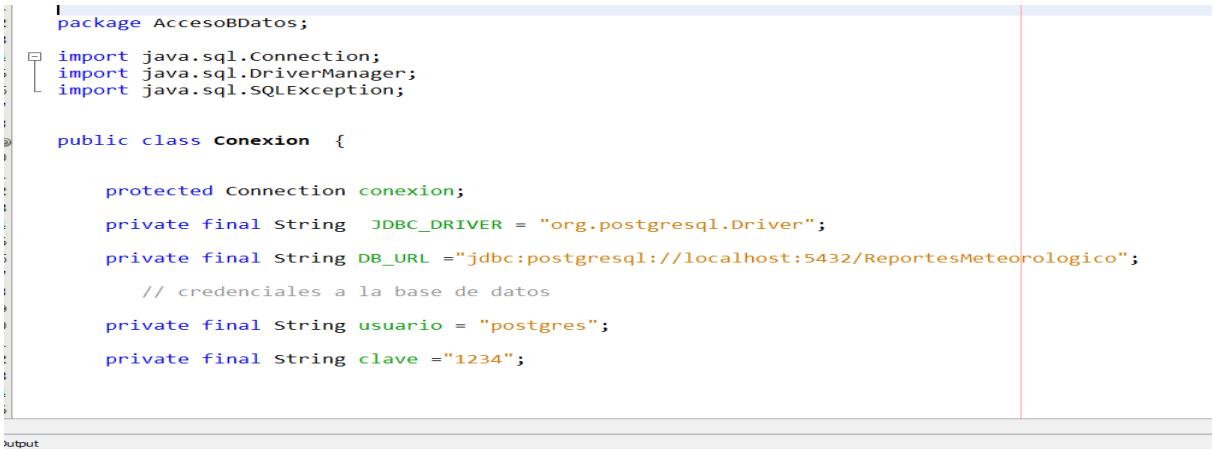
[5] GRASP: del acrónimo object-oriented design General Responsibility Assignment Software Patterns

reutilización. El propósito fundamental de este patrón es encontrar un creador que se conecte con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. En la aplicación se utiliza este patrón en la clase DAOReporteImpl.java.

```
@Override
public ArrayList consultar() throws Exception {

    ArrayList listaReportes = new ArrayList(); ;

    try {
        this.Conectar();

        PreparedStatement st = this.conexion.prepareStatement("SELECT * FROM public.\"Reporte\"");

        ResultSet rs = st.executeQuery();

        while (rs.next()) {

            Reporte rp = new Reporte();

            rp.setFECHA(rs.getDate("Fecha").toLocalDate());

            rp.setHORA(rs.getTime("Hora").toLocalTime());

            rp.setTEMPERATURA(rs.getFloat("Temperatura "));

            rp.setHUMEDARELATIVA(rs.getInt("Humedad"));

            rp.setVELOCIADVIENTO(rs.getInt("Velocidad"));

            rp.setDIRECCIONVIENTO(rs.getFloat("Direccion_Viento"));

            listaReportes .add(rp);
```

Figura 12 Patrón Creador Fuente: elaboración propia.

- Patrón Experto: Asignar responsabilidades al experto de la información, es decir, a la clase que tiene la información necesaria para realizar la tarea. Este patrón se evidencia en la clase Conexión debido a que es la única clase que posee las funcionalidades para realizar dicha acción. En la ilustración siguiente se evidencia el uso de este patrón.



```
package AccesoBDatos;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Conexion  {

    protected Connection conexion;
    private final String  JDBC_DRIVER = "org.postgresql.Driver";
    private final String DB_URL ="jdbc:postgresql://localhost:5432/ReportesMeteorologico";
       // credenciales a la base de datos
    private final String usuario = "postgres";
    private final String clave ="1234";
```

Figura 13 Representación del patrón experto de la clase Conexión Fuente: elaboración propia.

2.7.2 Patrones GoF

Los patrones de diseño según The Gang of Four (GOF) describen soluciones simples y elegantes a problemas específicos en el diseño de software orientado a objetos. El grupo GOF se dedicó a analizar los problemas recurrentes en el desarrollo de software y realizaron una clasificación y agrupación a partir de dos criterios: su propósito y alcance. Las categorías que definieron son: creacionales, estructurales y de comportamiento (Carlos A. Guerrero, 2013).

A continuación, se identifican los patrones utilizados en el desarrollo de la aplicación, de acuerdo a su clasificación.

- **Comportamiento:** este patrón consiste en permitir que un objeto modifique su comportamiento cada vez que cambie su estado interno. Esto se puede observar en los diferentes métodos asíncronos (ver figura siguiente) que realizan operación de desconexión.

```
private void jButtonDescActionPerformed(java.awt.event.ActionEvent evt) {

    if (puertoserial.isOpen()) {
        puertoserial.closePort();

      jComboBoxPUERTOSCOMM.setEnabled(true);
      jButtonOpen.setEnabled(true);
      jButtonDesc.setEnabled(false);


    }
}
```

Figura 14 Ejemplo del patrón de comportamiento aplicado en un método asíncrono elaboración propia.

- **Estructurales:** los patrones estructurales describen cómo las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos. De los patrones definidos como estructurales se utiliza la de composición:
- **Composición:** Es una colección de objetos en donde cada uno de estos objetos pueden ser una composición o un objeto sencillo. Dentro de la aplicación desarrolla se puede observar en la creación de la clase representacionDatos.java.

2.8 Tarjetas CRC

Las tarjetas CRC (Clases, Responsabilidad, Colaboración) son empleadas para la representación de las clases involucradas en el sistema definiendo las responsabilidades sobre las mismas. El formato físico de las tarjetas CRC facilita la interacción entre el cliente y el equipo de desarrollo de una forma simple y adaptable. Tiene como objetivo obtener un diseño simple y fácil de comprender por parte de los programadores *(León, 2006)*.En el desarrollo de la aplicación cliente se elaboraron 5 tarjetas CRC, que se muestran a continuación.

Tabla11 Target CRC de la clase **DAOReporteImpl**

Nombre de la clase: DAOReporteImpl	
Responsabilidades: Realizar la trasferencia de datos hacia la base datos	**Colaboradores:** Registro Conexión Controladora

Tabla 12 Target CRC de la clase **Conexión**

Nombre de la clase: Conexion	
Responsabilidades: Esta clase realiza la conexión al gestor de bases de datos	**Colaboradores:** DAOReporteImpl

Tabla 13 Tarjeta CRC de la clase **VistaPrincipal**

Nombre de la clase: VistaPrincipal	
Responsabilidades: Visualización de los datos en tablas y en gráfica	**Colaboradores:** Controladora

Tabla 14 Tarjeta CRC de la clase **Controladora**

Nombre de la clase: Controladora	
Responsabilidades: Recibe una petición y muestra la vista principal del sistema	**Colaboradores:** DAOReporteImpl VistaPrincipal

Tabla 15 Tarjeta CRC de la clase **InterDAOReporteImpl**

Nombre de la clase: : InterDAOReporteImpl	
Responsabilidades: Es una interfaz que expone los métodos de la clase	**Colaboradores:** DAOReporteImpl Registro

2.9 Conclusiones del capítulo

Con la realización del presente capítulo se define la línea base para el desarrollo de la estación Meteorológica. Se identificaron las funcionalidades específicas para su desarrollo y se definieron los patrones arquitectónicos y de diseño usados con el objetivo de lograr una mayor organización en los elementos que conforman la propuesta de solución. Además, se realizó el plan de entregas donde se indicaron las funcionalidades creadas para cada versión del sistema y las fechas en las que se publicaron estas versiones. Se obtuvo una planificación del tiempo de desarrollo de las iteraciones y el orden en que se implementaron las funcionalidades de acuerdo con la prioridad que le fue asignada.

CAPÍTULO III: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

El diseño de la solución informática conlleva a una etapa de implementación y pruebas. En el presente capítulo se dejan plasmadas las tareas de ingeniería y pruebas de aceptación a realizar durante cada iteración e HU correspondiente. De esta manera de obtiene como resultado un productor final probado y listo para su uso.

3.1 Fase de implementación

Dentro del ciclo de vida de un sistema informático se encuentra la fase de implementación. Esta es la fase más costosa y que consume más tiempo. Se dice costosa ya que muchas personas, herramientas y recursos están involucrados. La metodología XP propone que las historias de usuario deben ser implementadas en dependencia de la iteración a la que pertenezcan y las en (Maida & Pacienzia, Metodologías de desarrollo de software, 2015)

3.1.1 Tareas de ingeniería para la Iteración

Asociado a cada iteración, se encuentra la planificación de las tareas de ingeniería o programación. Cada una se caracteriza por estar asociada a una historia de usuario y varias tareas de ingeniería pueden pertenecer a una historia de usuario. Para la confección de cada una de las tareas se utilizaron tablas cuyo modelo cuenta con los siguientes campos:

- ✓ No. de tarea: numeración continua que identifica a la tarea.
- ✓ No. de HU: número de la HU a la que pertenece.
- ✓ Nombre de la tarea: identificación literal de la tarea.
- ✓ Tipo de tarea: tipo de tarea, dígase diseño, desarrollo, prueba.
- ✓ Puntos estimados: representación en porciento de la cantidad de tiempo estimada de una semana, que se utilizará para su realización.
- ✓ Fecha inicio: fecha estimada de inicio de realización.
- ✓ Fecha fin: fecha estimada de fin de realización.
- ✓ Descripción: se describe en qué consiste la tarea y qué elementos deben cumplirse para declarar la tarea terminada.

Seguidamente, se presentan las tareas de ingeniería perteneciente a cada historia de usuario

Tabla 16 Tarea 1 Desarrollar un software que permita la Adquisición de los datos.

Tarea de Ingeniería	
Número de tarea:1	Número HU:1
Nombre de la tarea : Desarrollar un software que permita la Adquisición de los datos	
Tipo de tarea: Desarrollo	Puntos estimados: 1 semana
Fecha inicio:2/10/2023	Fecha fin:6/10/2023
Se realiza la implementación de un firmware permita al microcontrolador la lectura de voltaje en sus pines entrada en correspondencia con el sensor y la conversión al dominio según el tipo de sensor a una magnitud física para su almacenamiento.	

Tabla 17 Tarea 2 Representar los valores en una pantalla de cristal líquido con I2C.

Tarea de Ingeniería	
Número de tarea:2	Número de HU:2
Nombre de la tarea: Representar los valores en un pantalla de cristal líquido con I2C	
Tipo de tarea: Desarrollo	Puntos estimados: 1 semana
Fecha inicio: 9/10/2023	Fecha fin: 13/10/2023
Descripción: se implementará un firmware que permita representar en una pantalla de cristal líquido de 20*4 los valores de los parámetros meteorológicos de la siguiente forma: Los valores de la fecha y la hora en la primera fila, en la segunda los valores de temperatura y humedad y en la tercera y cuarta los valores de velocidad y dirección del viento. Hay que tener en cuenta que se utilizó un módulo adaptador I2c que nos permite conectar el LCD al Arduino con solo 4 cables.	

Tabla 18 Tarea 3 Configurar la conexión serial del Arduino.

Tarea de Ingeniería	
Número de tarea:3	Número de HU:3
Nombre de la tarea: Configurar la conexión serial del Arduino	
Tipo de tarea: Desarrollo	Puntos estimados:1 semana
Fecha inicio: 16/10/2023	Fecha fin: 20/10/2023
Descripción: se desarrolla un firmware que nos permita la comunicación de Arduino con una comunicación UART convirtiendo los datos obtenidos en paquetes para su envió de forma serial La comunicación se realizara con una tasa de baudios 9600 con una longitud de datos de 8 bit, con un bit de parada y un bit control de flujo o reconstruyendo los datos a partir de los paquetes recibido se va transmitir los datos de forma serial.	

Tabla 19 Tarea 4 Validar los datos del puerto serial.

Tarea de Ingeniería	
Número de tarea:4	Número de HU:4
Nombre de la tarea : Validar los datos del puerto serial	
Tipo de tarea: Desarrollo	Puntos estimados:1 semana
Fecha inicio: 23/10/2023	Fecha fin: 27/10/2023
Descripción: se debe desarrollar una interfaz de usuario que se permita la conexión y desconexión a un puerto COMM seleccionado, con los parámetros siguientes: tasa de baudios 9600, longitud de datos de 8 bit, un bit de parada y un bit control de flujo.	

Tabla 20 Validación y almacenamiento de los datos.

Tarea de Ingeniería	
Número de tarea:5	Número de HU:5
Nombre de la tarea : Validación y almacenamiento de los datos	
Tipo de tarea: Desarrollo	Puntos estimados:1 semana
Fecha inicio: 30/10/2023	Fecha fin:3/11/2023
Descripción: se debe incorporar a la aplicación la función de corregir los errores ante la posible corrupción de los datos y la posibilidad de almacenar de forma automática valores validados	

Tabla 21 Representación de los datos.

Tarea de Ingeniería	
Número de tarea:6	Número. de HU:6
Nombre de la tarea : Representación de los datos	
Tipo de tarea: Desarrollo	Puntos estimados:1 semana
Fecha inicio: 6/11/2023	Fecha fin:10/11/2023
Descripción :Se le debe añadir las función al software de representar los valores de los datos en una tabla y la representación gráfica de los mismos pero con la selección por parámetros a seleccionar.	

3.2 Pruebas software

Las aplicaciones (en general cualquier mecanismo diseñado e implementado por un humano) son propensas a tener fallos. A veces, pueden contribuir al fracaso de cualquier proyecto de software, e impactar de forma negativa en toda una empresa. Los tiempos de desarrollo, los entornos de programación, las diferencias entre versiones, todo influye para que, incluso con la máxima dedicación,

puedan darse fallos que empañen la imagen y a veces la reputación, de una organización. Surge por tanto la necesidad de asegurar en lo posible, la calidad del producto.

Las pruebas de software son las investigaciones empíricas y técnicas cuyo fin es proporcionar información objetiva e independiente sobre la calidad del producto. Esta actividad forma parte del proceso de control de calidad global. Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de software y dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento del proceso de desarrollo.

Las pruebas del software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación **(S.Pressman, 2010)**. La metodología aplicada define que no debe existir ninguna funcionalidad en el programa que no haya sido probada. El equipo de desarrollo estará siempre acompañado por el cliente para convenir los detalles de los requerimientos y así poder implementarlos, probarlos y validarlos. De esta forma se brinda un resultado más completo para un producto final de manera creciente:

- ✓ **Pruebas internas:** se verifica el resultado de la implementación al probar cada construcción, y al incluir tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como diseños de casos de prueba, listas de chequeo y de ser posibles, componentes de prueba ejecutables para automatizar las pruebas.
- ✓ **Pruebas de liberación**: pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
- ✓ **Pruebas de Aceptación:** es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las que el software fue construido.

Las pruebas de software son un elemento crítico para garantizar el correcto funcionamiento de la aplicación. Entre sus metas se encuentra:

- Detectar defectos en el software.
- Verificar la integración adecuada de los componentes.
- Verificar que todos los requisitos se han implementado correctamente.
- Identificar y asegurar que los fallos encontrados durante el proceso de prueba, se han corregido antes de entregar el software al cliente.

3.2.1 Pruebas de aceptación

Las pruebas de aceptación son creadas sobre la base de las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una HU ha sido correctamente implementada. Las pruebas de aceptación son consideradas como "pruebas de caja negra". Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos (J. J. Gutiérrez, 2019)

Las pruebas de aceptación significan la satisfacción del cliente con el producto desarrollado; es por esto, que el cliente es la persona adecuada para diseñarlas. Se tomó como criterio de aprobación de cada iteración el 100% de los casos de prueba exitosos para pasar de iteración.

El cliente debe especificar uno o diversos escenarios para comprobar que una HU ha sido correctamente implementada. Para la representación de las pruebas de aceptación se definieron los siguientes elementos:

- ✓ **Código de la prueba:** representa al caso de prueba, incluye el número de HU y de la prueba.
- ✓ **Nombre de Historia de Usuario:** nombre de la historia de usuario.
- ✓ **Descripción de la prueba:** acción que debe realizar el sistema.
- ✓ **Condiciones de ejecución:** describe las características y elementos que debe contener el sistema para realizar el caso de prueba.
- ✓ **Entrada /Pasos de ejecución:** incluye las entradas necesarias para realizar el sistema, además de los pasos para realizar el caso de prueba.
- ✓ **Resultado Esperado:** descripción de la respuesta del sistema ante el caso de prueba.
- ✓ **Resultado Obtenido:** respuesta visual del sistema después de realizar el caso de prueba. **Evaluación de la Prueba:** clasificación de la prueba en satisfactoria o insatisfactoria.

- ✓

A continuación, se muestran las pruebas de aceptación realizadas a las 6 HU.

Tabla 22 Código de la Prueba HU1,2,3

Código de la Prueba HU1,2,3	**Nombre de Historia de Usuario:** Representar los valores en las magnitudes físicas en el LCD.
Descripción de la Prueba: La prueba consiste en la observación de los valores de los parámetros físicos censados por los sensores.	
Condiciones de Ejecución: La estación de meteorología debe de estar conectada a la fuente de alimentación.	
Entrada /Pasos de ejecución Conectar la placa expansión a la tarjeta Arduino con todos los sensores y el LCD más el conversor de comunicación serial Rs232 a radio frecuencia.	
Resultado Esperados: Se debe mostrar los valores que los parámetros ambientales.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 23 Código de la Prueba HU4.

Código de la Prueba HU4	**Nombre de Historia de Usuario:** Selección puerto Comm
Descripción de la Prueba: En esta prueba se debe confirmar que el sistema sea capaz de realizar la conexión y desconexión así como la lectura de un puerto serial seleccionado.	
Condiciones de Ejecución: Para realizar la prueba la estación meteorológica debe de estar conectada con la aplicación por los conectores serial.	
Entrada /Pasos de ejecución Abrir la aplicación en la pc, conectar la estación de meteorología, selección del puerto serial e iniciar la conexión.	

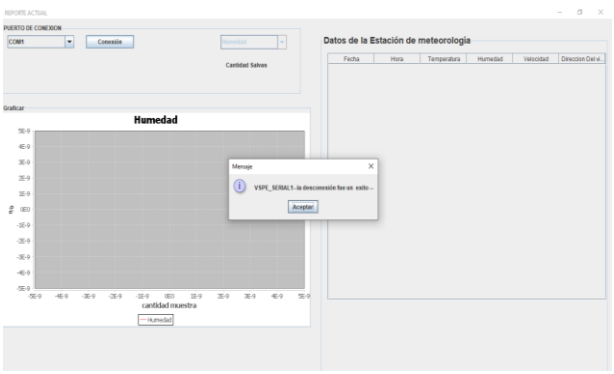
Resultado Esperados: La aplicación lanza una Excepción.
Resultado Esperado: 
Evaluación de la Prueba: Satisfactoria.

Tabla 24 Código de la Prueba HU5.

Código de la Prueba HU5	**Nombre de Historia de Usuario:** Almacenar los datos
Descripción de la Prueba: La prueba consiste en la verificación contante de los datos que se envían o se leen el puerto serio. El objetivo de la esta prueba es la reacción del sistema cuando exista algún problema con la lectura de los datos el sistema tenga una reacción	
Condiciones de Ejecución: Para realizar la prueba la estación meteorológica debe de estar conectada con la aplicación Y debe de haberse iniciado la conexión.	
Entrada /Pasos de ejecución Abrir la aplicación en la pc, conectar la estación de meteorología, selección del puerto serial e iniciar la conexión.	
Resultado Esperados: La aplicación lanza una Excepción	
Resultado Esperado:	

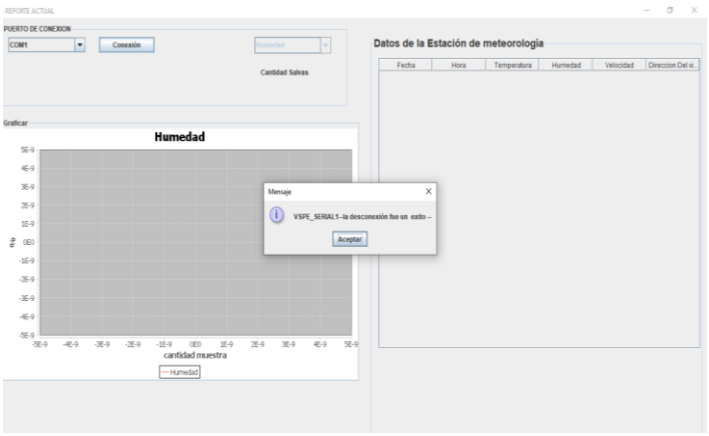
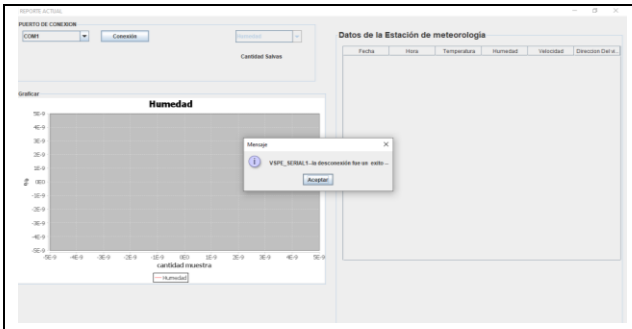

Evaluación de la Prueba: Satisfactoria

Tabla 25 Código de la Prueba HU4.

Código de la Prueba HU3 P1	**Nombre de Historia de Usuario:** Conexión con la Estación de meteorología
Descripción de la Prueba: La prueba consiste en activar la propuesta de solución El objetivo de esta prueba es ver la reacción de la aplicación cuando no puede establecer la conexión con el módulo de la radio línea	
Condiciones de Ejecución: El prototipo de estación meteorológica debe de estar conectada para la realización de la prueba.	
Entrada /Pasos de ejecución Abrir la aplicación en la pc, conectar la estación de meteorología, selección del puerto serial e iniciar la conexión.	
Resultado Esperados: La aplicación lanza una Excepción	
Resultado Esperado:	



Evaluación de la Prueba: Satisfactoria

El proceso de pruebas a cualquier software se realiza a través de iteraciones, donde, a medida que se procede con una nueva iteración deben haberse erradicado los defectos encontrados en la anterior, para garantizar que al final del proceso el producto quede libre de la mayor cantidad de errores posibles y listo para entregar al cliente

Se realizaron un total de 4 casos de prueba durante la etapa de Pruebas, específicamente Pruebas de Aceptación. Estas pruebas se desarrollaron por iteraciones. Los resultados obtenidos durante las mismas se muestran a continuación a través de un gráfico de barras con los porcientos de pruebas con resultados satisfactorios o no satisfactorios.

Como se muestra en el gráfico de barras cada iteración se realizó una prueba específica pruebas según la HU, Para las cuales para las cuales en las iteraciones1, 2, 3 y 4 se realizaron 1,2, 3 y 4 casos de prueba respectivamente obteniéndose un 100% de resultados satisfactorios y 0% de resultados no satisfactorios demostrando el cumplimiento adecuando de los requisitos propuestos para estas iteraciones.

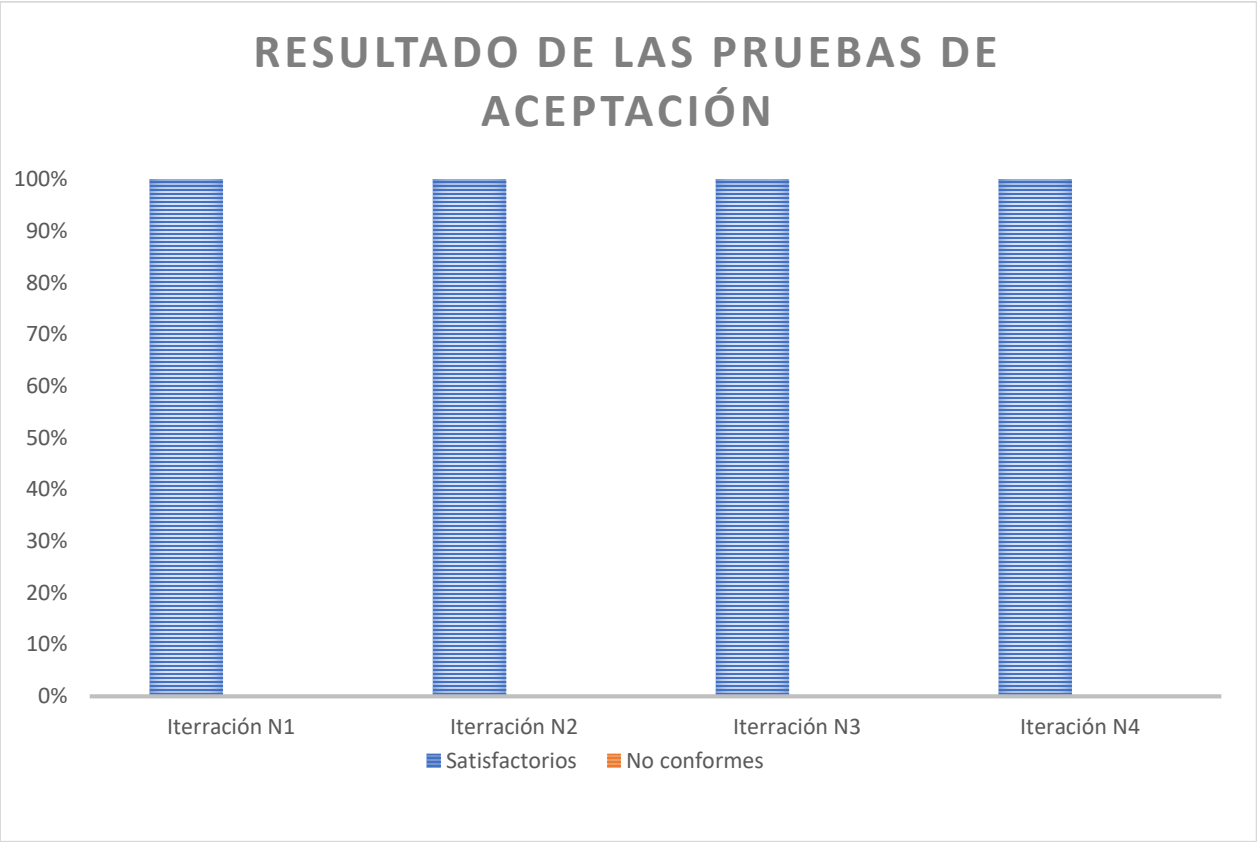


Figura 15 Resultados por iteración.

3.6 Conclusiones parciales

Con la ejecución de las fases de implementación y prueba se elaboraron los diferentes casos de prueba, a los que se le realizaron pruebas de aceptación y se tomó como criterio de aprobación el 100% de los casos de prueba satisfactorios por iteración. Se determinó que la propuesta de solución desarrollada cumple con todos los requerimientos acordados con el cliente.

Conclusiones

La investigación permitió arribara las siguientes conclusiones:

- ✓ La recopilación de los referentes teóricos asociados al desarrollo de una estación de meteorológica con Arduino posibilitó un mejor entendimiento del contexto de la investigación y de la problemática a resolver, así como identificar las tecnologías adecuadas y menos costosas.

- ✓ Se obtuvo una estación meteorológica basada en dispositivos Arduino que tiene como objetivo contribuir al estudio de la meteorología en Cuba, y que permite realizar el análisis de los parámetros meteorológico en microclimas y en lugares de difícil acceso de forma remota.

- ✓ Mediante la realización de las pruebas de aceptación se validó que la propuesta de solución cumple con los requerimientos pactados con el cliente y permitió el cumplimiento del objetivo trazado en la investigación.

RECOMENDACIONES

El objetivo de este trabajo fue alcanzado, pero durante su desarrollo surgieron nuevas ideas que serían recomendables tener en cuenta para su futuro perfeccionamiento como son:

- Mejorar la calidad de los sensores.
- Implementación de protocolo MQTT en el servidor (plataforma Arduino).
- Se recomienda el huso de la plataforma Internet de las Cosas para la gestionar los datos.

ANEXOS

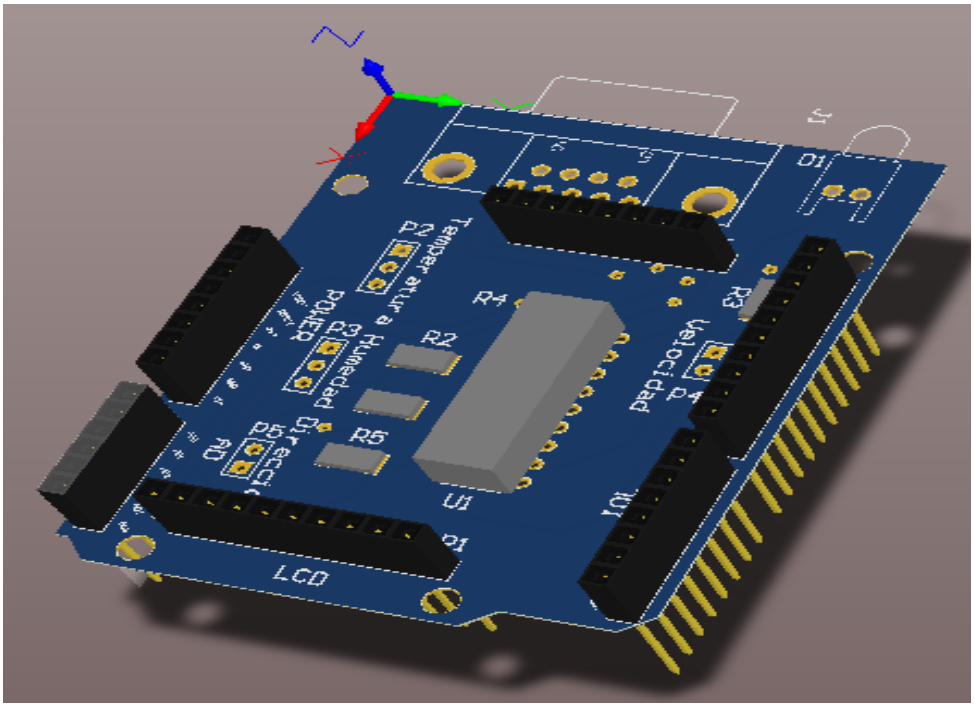


Figura 16 Vista en 3D del diseño de “shield” para el huso del Arduino como parte de la estación de meteorologica Fuente: elaboración propia.

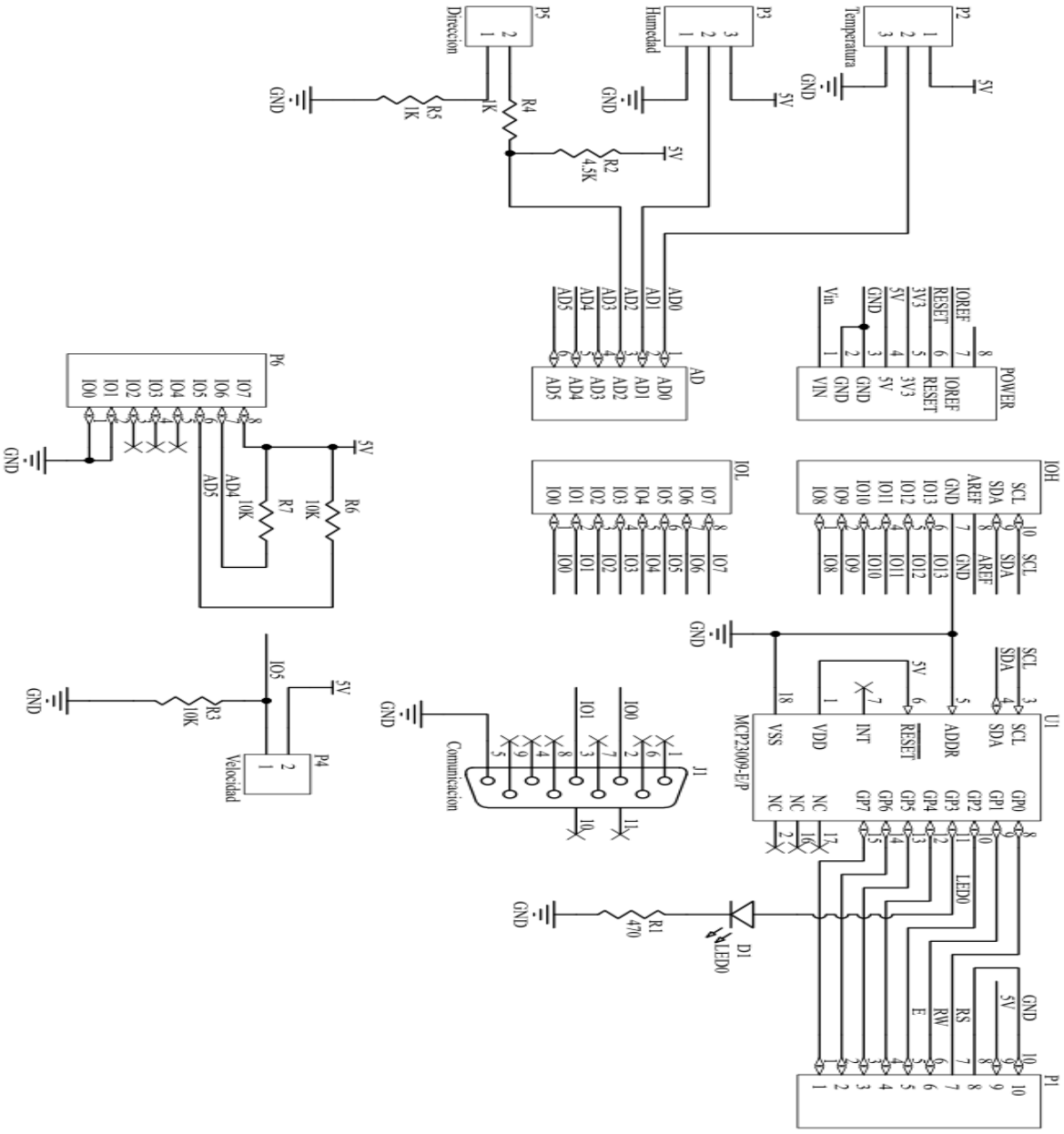


Imagen 1 Esquema eléctrico de la tarjeta extensora para Arduino (Fuente: Elaboración propia).

BIBLIOGRAFÍA

1. *ARDUINO Y EL INTERNET DE LAS COSAS.*
2. **AliExpress. 2020.** https://es.aliexpress.com/item/32893647929.html?gatewayAdapt=glo2esp. [En línea] 2020.
3. **Alina Karla Quesada Somano, Alberto Medina León. 2020.** *Métodos teóricos de la Investigación .* Matanza : s.n., 2020.
4. **Arduino. 2023.** Arduino. [En línea] noviembre de 2023. https://www.arduino.cc/en/Main/Software.
5. **Artero, Óscar Torrente. febrero 2013.** *ARDUINO Curso práctico de formación.* s.l. : Alfaomega Grupo Editor, S.A. de C.V., México, febrero 2013.
6. **Bareño, Carlos I. Camargo. 2011.** *Metodolog´ıa Para la Transferencia Tecnologica en la industria Electronica basada en software libre y Hardware CopyLeft Software Libre y .* Colombia : s.n., 2011.
7. **Campbell scientific. 2023.** Sensor Temperatura y Humedad Relativa del aire. [En línea] 2023. https://www.campbellsci.es/ee181.
8. **CAMPBELL SCIENTIFIC SPAIN. 2023.** Estaciones meteorológicas automáticas. [En línea] 2023. https://www.campbellsci.es/automated-weather-stations.
9. **Campbell scientifico. 2023.** EE181 Sensor Temperatura y Humedad Relativa del aire. [En línea] 2023. https://www.campbellsci.es/ee181.
10. **Carlos A. Guerrero, Johanna M. Suárez y Luz E. Gutiérrez. 2013.** *Patrones de Diseño GOF (La Banda de los Cuatro) en el contexto deProcesos de Desarrollo de Aplicaciones Orientadas a la Web.* Colombia : s.n., 2013.
11. **Carrillo Flores, René Alfonso, Yánez Bucheli, Rubén Arturo y Chancusig Llano, Darío Xavier. 2017.** *Desarrollo del sistema de control presupuestario para el Departamento de Financiero de la Facultad de Ingeniería Ciencias Físicas y Matemática.* Universidad Central del Ecuador. Ecuador : Quito: UCE, 2017.
12. **Dixys, Hernández Rojas, y otros. 2018.** ARDUINO Y EL INTERNET DE LAS COSAS . s.l. : Editorial Área de Innovación y Desarrollo,S.L., 2018.
13. **Erich Gamma, Richard Helm. 2010.** *Patrones de Diseño.* 2010.
14. **F., Orlando Franco. 2018.** *Redes de Estaciones Meteorológicas Automáticas y sus Aplicaciones Productiva.* Chile : s.n., 2018.

15. **Fernandez, M. 2023.** *Puesta en producción segura. Colombia: Ediciones de la U.* Colombia : Ediciones de la U., 2023.
16. **Fernández, Oscar Mauricio. 2019.** http://codigoelectronica.com/blog/lm35-datasheet. [En línea] 4 de 11 de 2019. [Citado el: 10 de 10 de 2023.]
17. **Fidel. 1960.** *Discurso en acto celebrado por la Sociedad Rspeleológica de Cuba, Academia de Ciencia.* 1960.
18. **Fraden, . 2010.** *Handbook of Modern Sensors.* 7ma. New York : # Springer ScienceþBusiness Media, LLC 2010, 2010. Library of Congress Control Number: 2010932807.
19. **G. Medina García, J. Grageda Grageda, J. A. Ruiz Corral y A. D. Báez González,. 2008.** *Uso de estaciones meteorológicas en la agricultura.* 2008.
20. **Gaines, Jeff, Boyd, Geraldine y Copley, Della. 2017.** Visual Paradigm Online. *Visual Paradigm Online.* [En línea] 2017. https://online.visual-paradigm.com/es/features/.
21. **GATTINONI, N., BOCA, T. y C, R.,. 2011.** *Comparación entre observaciones meteorológicas obtenidas de estaciones convencionales automáticas a partir de la estimación de parámetros estadísticos.* España : s.n., 2011.
22. **Gomes, José Manuel Piñeiro. 2022.** *Entorno de desarrollo i.* Mexico : Ediciones Paraninfo,SA, 2022. primera Edicion.
23. **Gutiérrez, José Manuel Ruiz. 2007.** *Manual de Programación.* an Francisco, California, 94105, USA : s.n., 2007.
24. **Gutiérrez, Sergio E. Flores. 2008.** *Diccionario de Términos Meteorológicos más usuales.* Mexico : Organización Mexicana de Meteorólogos A. C, 2008.
25. **Halfacree, Garet. 2020.** *La guia oficial de Raspberry Pi para para principiante.* 2020.
26. **Hiken, Arthur. 2023.** Una onza de prevención: seguridad y protección a través de estándares de codificación de software. [En línea] 4 de 9 de 2023.
27. **Ing. Alina Karla Quesada Somano, Dr. C. Alberto Medina León. 2020.** *MÉTODOS TEÓRICOS DE INVESTIGACIÓN: ANÁLISIS-SÍNTESIS, INDUCCIÓN-DEDUCCIÓN, ABSTRACTO -CONCRETO E HISTÓRICO-LÓGICO.* Universidad de Matanzas, Matanzas : Universidad de Matanzas. http://monografias.umcc.cu/monos20.htm., 2020.
28. **Ingeniería MCI . 2014.** ¿Qué es Arduino? | Arduino.cl - Compra tu Arduino en Línea. [En línea] 20 de 11 de 2014. [Citado el: 26 de 6 de 2023.] https://arduino.cl/que-es-arduino/.
29. **J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres. 2019.** *PRUEBAS DEL SISTEMA EN PROGRAMACIÓN EXTREMA .* Colombia : s.n., 2019.

30. **Larman, Craig. 2016.** *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* 3ra . s.l. : Prentice-Hall, 2016.
31. **—. 2003.** *UML y Patrones.Introducción al analisis y diseño orientado a objetos.* Mexico : PRENTICE HALL, 2003.
32. **Lázaro, Eduardo. 2022.** neoguias.com. [En línea] 20 de 1 de 2022. https://www.neoguias.com/tipos-notacion-nombres/.
33. **León, Alina Karla Quesada Somano y Alberto Medina. 2020.** https://www.researchgate.net/publication/347987929_METODOS_TEORICOS_DE_INVESTIGACION_ANALISIS-SINTESIS_INDUCCION-DEDUCCION_ABSTRACTO_-CONCRETO_E_HISTORICO-LOGICO. *www.researchgate.net.* [En línea] 2020.
34. **León, Pedro J. Ponce de. 2006.** *INTRODUCCIÓN AL DISEÑO ORIENTADO A OBJETOS.* 2006.
35. **Llamas, Luis. 2014.** Comunicación de Arduino con puerto serie. [En línea] 17 de Abril de 2014. https://www.luisllamas.es/arduino-puerto-serie/.
36. **Luis Miguel Echeverry, Luz Elena Carmona. 2007.** *Caso práctico de la metodologia Agil XP al desarrollo de software.* Colombia : s.n., 2007.
37. **Maida, EG, Pacienzia, J. Metodologías de desarrollo de software. 2015.** *Tesis de Licenciatura en Sistemas y Computación .Facultad de Química e Ingenieria "Fray Rogelio Bacon". Universidad Católica Argentina, 2015.* 2015.
38. **Maldonado l., Isaac y Aravena S., René. 2006.** *Redes de estaciones meteorológicas automáticas y sus aplicaciones productivas.* Chile : s.n., 2006.
39. **María José Pérez Pérez, Francisco J. González Cabrera. 2018.** *Guía Comparativa de Metodologías Ágiles.* 2018.
40. **marin, Rafael. 2019.** Los gestores de bases de datos más usados en la actualidad. [En línea] 16 de 04 de 2019. https://www.inesem.es/revistadigital/informatica-y-tics/los-gestores-de-bases-de-datos-mas-usados/.
41. **Martínez, Elizabeth Verdecia. 2015.** *Extensiones de Visual Paradigm .* La habana : s.n., 2015.
42. **Mayné, Jordi. 2003.** *Sensores Acondicionador y procesador de señal.* 2003.
43. **MCI electronics. 2023.** Arduino.cl. [En línea] 2023. https://arduino.cl/producto/sensores-para-estacion-meteorologica/.
44. **MCI ELECTRONICS. 2023.** raspberryPI. [En línea] 2023. https://raspberrypi.cl/que-es-raspberry/.

45. **MCL electronics. 2022.** Arduino.cl. [En línea] 2022. https://arduino.cl.
46. **Melendez, . 2022.** Estaciones Meteorológicas: definición, tipos e instrumentos. [Online] sadvance, 09/ 02 2022. [Cited: 28 06 2023.] https://somosadvance.com/expertise/estaciones-meteorologicas/.
47. **—. 2022.** Advance. [En línea] 09 de 02 de 2022. https://somosadvance.com.
48. *Métodos científicos de indagación y.* **Alipio Omar Pérez Jacinto, Andrés Rodríguez Jiménez. 2017.** 2017.
49. *Métodos científicos de indagación y de construccion del conocimiento.* **Andrés Rodríguez Jiménez, Alipio Omar Pérez Jacinto. 2017.** 17 de 3 de 2017, Revista EAN, 82, pp.179-200. https://doi.org/10.21158/01208160.n82.2017.1647.
50. *Métodos científicos de indagación y de construcción del conocimiento.* **Jiménez, Andrés Rodríguez. 2016.** Colombia : s.n., 2016, Revista Escuela de Administración de Negocios.
51. **Monk, Simon. 2014..** *Programming Arduino.* 2014.
52. **MONTALVO LEZAMA, B.,. 2014.** *Prototipo didáctico de una estación meteorológica monitoreada a Distancia. S.l.: Instituto Politecnico Nacional de Mexico.* 2014.
53. **Morales, Eurelio. 2020.** Descubre el nuevo pgAdmin 4 para trabajar con PostGIS. [En línea] 10 de 5 de 2020. https://mappinggis.com/2017/11/descubre-el-nuevo-pgadmin-4-para-trabajar-con-postgis/#:~:text=pgAdmin%20es%20una%20herramienta%20indispensable,bases%20de%20datos%20espaciales%20PostGIS..
54. *NetBeans IDE 4.1 la alternativa a eclipse.* **M. Domínguez-Dorado. 2023.** 13, España : C/del rio ter,Nave 13, 13 de marzo de 2023, PROGRAMACION.
55. **Olivera Sosa, Ángel Gabriel. 2019.** . *Planificación y modelado.* 2019.
56. **ORACLE. 2023.** *Introducción al lenguaje Java.* 2023.
57. **PCE Ibérica S.L. 2022.** http://www.pce-iberica.es/. [En línea] 2022.
58. **Pérez, Velazco y Mir. 2010.** *Desarrollo de la extensión para la configuración del módulo de adquisición de datos del SCADA Guardián del ALBA.* 2010.
59. **Pressman. 2002.** 2002.
60. **Pressman, Roger S. 2005.** *Ingeniería del Software.* 2005.
61. **Raspberry Pi . 2023.** [En línea] 2023. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/.
62. **Roger S. Pressman, Ph.D. 2015.** *INGENIERÍA DEL SOFTWARE. UN ENFOQUE PRÁCTICO.* Mexico : Industria Editorial Mexicana,, 2015. Reg. Núm. 736.

63. **Rosa Maria Rodriguez, Agueda Benito,Adelaida Portela Lozano. 2004.** *Meteorología y Climatología.* España : s.n., 2004.
64. **Ruz, Fidel Alejandro Castro. 1960.** Discurso en acto celebrado por la Sociedad Espeleológica de Cuba,Academia de Ciencia. 15 de 1 de 1960.
65. **S.Pressman, Roger. 2010.** *Ingeniería del software Un enfoque práctico.* s.l. : 7ma edición, 2010.
66. **Sánchez, Tamara Rodríguez. 2014.** *Metodología de desarrllo para Actividad productiva de la UC.* La Habana : s.n., 2014.
67. **Slavkova, Olga. 1993.** *Heramientas case.* 1993.
68. **Smith, Alan G. 2011.** *Introducción a arduino.* 2011.
69. **Solis, Diego Caceres. 23.** El Raspberry Pi es un dispositivo electrónico e informático que también se puede describir como una minicomputadora de bajo costo y tamaño compacto; tiene la capacidad de conec-tar un monitor o televisor mediante sus conexiones HDMI, y se puede administra. [En línea] 11 de 5 de 23. https://openwebinars.net/blog/comparativa-arduino-vs-particle-vs-raspberry-pi/.
70. **Solís, Manuel Calero. 2003.** *Una explicación de la programación extrema (XP).* Madrid : s.n., 2003.
71. **Sommerville, Ian. I. 2005.** *ngeniería de Software.* 2005.
72. **Tamara, R. S. 2015.** *Metodología de desarrollo para la Actividad productiva de la UCI.* 2015.
73. **Weis, Olga. 2020.** Serial Port Monitor. [En línea] 44 de 2 de 2020. https://www.serial-port-monitor.org/es/articles/serial-communication/.
74. **Wilson, Jose Manuel Ayala. 2020.** Blog sobre arquitectura. [En línea] 2020. http://jmaw.blogspot.com/2011/04/h2-margin-bottom-0.html..