DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS E INFORMÁTICA

TEMA: SEGUIMIENTO DE DATOS EN TIEMPO REAL CON APACHE KAFKA EN RASPBERRY PI; CASO PRÁCTICO: MONITOREO AMBIENTAL DE LA ACTIVIDAD DEL INVERNADERO ESPE-IASA I

AUTOR: BARBA BARBA, CRISTIAN JOSÉ

DIRECTOR: Ing. DÍAZ ZÚÑIGA, MAGI PAÚL Msc, MBA.

SANGOLQUÍ
2018

CERTIFICADO

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

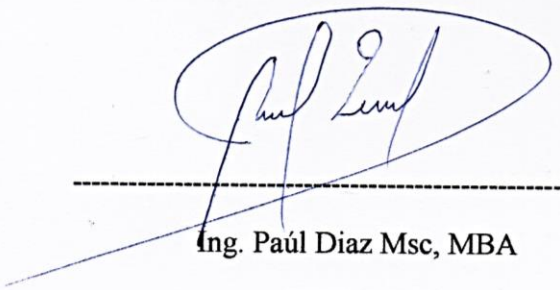
CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

CERTIFICACIÓN

Certifico que el trabajo de titulación, "**SEGUIMIENTO DE DATOS EN TIEMPO REAL CON APACHE KAFKA EN RASPBERRY PI; CASO PRÁCTICO: MONITOREO AMBIENTAL DE LA ACTIVIDAD DEL INVERNADERO ESPE-IASA I**" fue realizado por el señor **CRISTIAN JOSE BARBA BARBA** el mismo que ha sido revisado en su totalidad y analizado por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 07 de septiembre del 2018

Firma:



Ing. Paúl Diaz Msc, MBA

1707249072

DIRECTOR

AUTORÍA DE RESPONSABILIDAD

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

AUTORÍA DE RESPONSABILIDAD

Yo, **CRISTIAN JOSE BARBA BARBA,** declaro que el contenido, ideas y criterios del trabajo de titulación "**SEGUIMIENTO DE DATOS EN TIEMPO REAL CON APACHE KAFKA EN RASPBERRY PI; CASO PRÁCTICO: MONITOREO AMBIENTAL DE LA ACTIVIDAD DEL INVERNADERO ESPE-IASA I"** es de mi autoría y responsabilidad, cumpliendo con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Consecuentemente el contenido de la investigación mencionada es verás.

Sangolquí, 07 de septiembre del 2018

Firma:

CRISTIAN JOSE BARBA BARBA

C.C.1722111547

AUTORIZACIÓN

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

AUTORIZACIÓN

Yo, **CRISTIAN JOSE BARBA BARBA**, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación **"SEGUIMIENTO DE DATOS EN TIEMPO REAL CON APACHE KAFKA EN RASPBERRY PI; CASO PRÁCTICO: MONITOREO AMBIENTAL DE LA ACTIVIDAD DEL INVERNADERO ESPE-IASA I"** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.

Sangolquí, 07 de septiembre del 2018

Firma:

CRISTIAN JOSE BARBA BARBA

C.C.1722111547

DEDICATORIA

Dedico este trabajo a mi familia: Pedro, Lola, Xavier, Carolina y Anthony quienes han sido el pilar fundamental a lo largo de mi carrera Universitaria, sabiéndome guiar. Dios les pague por todo esa comprensión y amor.

A Leslie quien desde los inicios de mi carrera universitaria me apoyo en los momentos más difíciles estuvo a mi lado con paciencia y supo brindarme su apoyo este trabajo es en dedicación tuya.

A mis abuelos y a mis tías que siempre se dedicaron a mí y me apoyaron en mi vida y saber valorar cada momento de ella.

Cristian José Barba

AGRADECIMIENTO

Agradezco primero a mi Dios todo poderoso que me dio la sabiduría y fortaleza para salir de todo triunfante, te agradezco padre bondadoso por alcanzar esta meta en mi vida.

Agradezco a mis amados padres y hermanos que siempre estuvieron a mi lado, que gracias al esfuerzo de ellos pude culminar mi carrera universitaria.

A cada uno de mis profesores quienes con su paciencia supieron infundidme el conocimiento necesario para estar preparado en mi vida profesional.

Al ingeniero Ing. Paul Diaz Msc, MBA, quien a lo largo de mi carrera universitaria ha sido un excelente maestro y un amigo más por brindarme su confianza para mi desarrollo académico y para el progreso de esta investigación con todo su conocimiento y paciencia.

A mi compañero René Carrera quien ha sido vital en esta investigación con toda su enseñanza y experiencia a lo largo del proyecto.

A mis amigos de la carrera de sistemas por todo lo adquirido a lo largo de estos años de estudio y las experiencias vividas.

Gracias a todos.

Cristian José Barba

INDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

GLOSARIO

Rasberry Pi: Raspbian es el software principal y básico para dispositivos Raspberry Pi, oficialmente respaldado por la Fundación Raspberry Pi. De hecho, es un sistema operativo, basado en Debian y optimizado para hardware Raspberry Pi.

Arduino: Es una plataforma electrónica de código abierto, basada en hardware y software fácil de usar. Las placas Arduino pueden leer entradas (sensor) y convertirlo en una salida.

Confluent: Fue creada por los mentores de Apache Kafka, ofrece una ejecución completa de Kafka para Enterprise, para ayudarlo a administrar su negocio en tiempo real.

Nodo: cada vez que se hable de nodo haremos referencia a la placa de Arduino con los 4 sensores de toma de datos.

Grafana: Es un programa interactivo, que permite al usuario generar dashboard en datos almacenados en un archivo de disco. La principal ventaja y originalidad del sistema es que el usuario puede determinar interactivamente el efecto visual de un dashboard y verlo inmediatamente en la pantalla.

Dashboard: Es una herramienta para gestionar la información que realiza un seguimiento visual, analiza y muestra indicadores clave de rendimiento (KPI), métricas y puntos de datos clave para inspeccionar el estado de un negocio, departamento o proceso específico. Son personalizables para satisfacer las necesidades específicas de un departamento y una empresa.

Crontab: es un comando UNIX que crea una tabla o lista de comandos, cada uno de los cuales debe ser ejecutado por el sistema operativo en un momento específico.

RESUMEN

En la actualidad, se hace imprescindible que los invernaderos dispongan de herramientas que permitan monitorear la información de su producción. Por tal razón, contar con una solución de IoT (Internet de las cosas), reduciendo el tiempo necesario para la obtención de datos en tiempo real. Sin embargo, los encargados de la producción de invernaderos exigen nuevas y revolucionarias capacidades a este tipo de soluciones y la industria está respondiendo para cumplir las crecientes necesidades de sus clientes. Si la información de negocio está incompleta, dañada o no es válida, los datos corruptos pueden hacer que los encargados de monitorear el estado del invernadero tomen decisiones que en realidad disminuyen el rendimiento y la rentabilidad. Dentro la floricultura en el Ecuador: un punto de fallo son los concentradores de métricas medioambientales. Los posibles fallos pueden ser del propio hardware del concentrador: no pudiendo recolectar las métricas de los sensores o de algún proceso crítico: ya sea del sistema operativo o la aplicación que se esté ejecutando en local. Uno de los problemas de las bases de datos en tiempo real está diseñadas para ser organizadas en tablas de tal forma que se pueda entender; lastimosamente, cuando estos sistemas fueron diseñados se pensaba en sistemas pequeños, estructurados y centralizados, pero eso cambio y la información dejó de ser tan “estructurada”, los sistemas crecieron a un ritmo exponencial y se hizo necesario distribuir la información, lo que ocasionó que estas bases de datos sean más lentas. Los motivos de la elección de Kafka considerados es que es una aplicación de código abierto es decir libre distribución y de mucho rendimiento que actualmente es utilizado por empresas internacionales de renombres. Con Zookeeper conjuntamente hacen de Kafka una gran alternativa para el procesamiento de datos en tiempo Real.

PALABRAS CLAVE:

- **INTERNET DE LAS COSAS.**
- **FLORICULTURA.**
- **APACHE KAFKA.**
- **GRAFANA.**
- **TIEMPO REAL.**

ABSTRACT

At present, it is essential that greenhouses have tools to monitor the information on their production. For this reason, to have a solution of IoT (Internet of things), reducing the time required to obtain data in real time. However, green house producers demand new and revolutionary capabilities from these solutions and the industry is responding to meet the growing needs of their customers. If business information is incomplete, corrupted or invalid, corrupt data can cause greenhouse condition monitors to make decisions that decrease performance and profitability. Within Ecuador's floriculture sector: one point of failure is the concentration of environmental metrics. The possible failures can be of the hardware of the concentrator itself: not being able to collect the metrics of the sensors or of some critical process: either of the operating system or the application that is being executed in local. One of the problems with real-time databases is that they are designed to be organized in tables in such a way that they can be understood; unfortunately, when these systems were designed, small, structured and centralized systems were thought of, but that changed and the information stopped being so "structured", the systems grew at an exponential rate and it became necessary to distribute the information, which caused these databases to be slower. The reasons for the choice of Kafka considered is that it is an open source application i.e. free distribution and high performance that is currently used by renowned international companies. Together with Zookeeper they make Kafka a great alternative for real-time data processing.

KEYWORDS:

- **INTERNET OF THINGS**
- **FLORICULTURA.**
- **APACHE KAFKA.**
- **GRAFANA.**
- **REAL TIME.**

CAPÍTULO I
INTRODUCCIÓN

1.1 ANTECEDENTES

En una era donde el despunte y apogeo del Big Data, los extensos volúmenes de datos cada vez son mayores y las necesidades de tener resultados en tiempo real han puesto en primera línea a las tecnologías streaming. Este nuevo concepto amplía un entorno de muchas posibilidades en la obtención y el procesamiento de datos.

"Las tecnologías de streaming están causando sensación últimamente. En lo que podremos estar definiendo el amanecer de la revolución del big data, si bien existen frameworks de procesamiento como Apache Kafka, pero estos se centran en flujos de tareas bach (o por lotes)" (Campo, 2017). Pero en tiempo real no se podía obtener datos por su baja latencia. Considerando que restos datos en cuanto a volumen crecían constantemente, y para tener información sobre estos se tiene que requerir a más herramientas para poder solventar este problema.

Existieron numerosas herramientas que se presentaron para solucionar este inconveniente. Muchas de estas herramientas se basan en una solución que ya existía (como Spark Streaming, Kafka Streams o Akka Streams) y también existen otras como (Flink o Apex) pero estas otras nuevas utilizan un concepto de streaming de datos más ligero. Hay incluso una capa de abstracción que puede usar diferentes runners (Beam).

Si bien los sistemas de mensajería más comunes como JMS, ActiveMQ, RabbitMQ han sido de bastante utilidad en los escenarios tradicionales, estos no son eficientes ni útiles en el manejo de escenarios de Big Data. Normalmente, este escenario requiere manejar cientos de miles de mensajes por segundo. (Campo, 2017).

1.2 PLANTEAMIENTO DEL PROBLEMA

Uno de los problemas de las bases de datos en tiempo real está diseñadas para ser organizadas en tablas de tal forma que se pueda entender; lastimosamente, cuando estos

sistemas fueron diseñados se pensaba en sistemas pequeños, estructurados y centralizados, pero eso cambio y la información dejó de ser tan “estructurada”, los sistemas crecieron a un ritmo exponencial y se hizo necesario distribuir la información, lo que ocasionó que estas bases de datos sean más lentas (Castro Romero, 2012).

“Grandes empresas como Facebook, Twitter, Amazon, Google entre otras utilizan hoy en día las bases de datos no relacionales, demostrando que este tipo de bases de datos son la alternativa para mejorar la escalabilidad, flexibilidad, velocidad en la manipulación de grandes cantidades de datos y las diferentes formas de almacenamiento de la información” (Ana Barragán, 2013). “Debido a esto, las empresas deciden cambiar el uso de bases de datos relacionales por bases de datos no relacionales, donde la elección incorrecta de base de datos genera posibles pérdidas de información o baja calidad de los datos almacenados” (PowerData, 2016),

1.3 JUSTIFICACIÓN

En la actualidad, se hace imprescindible que los invernaderos dispongan de herramientas que permitan monitorear la información de su producción. Por tal razón, contar con una solución de IoT (Internet de las cosas), reduciendo el tiempo necesario para la obtención de datos en Tiempo real.

Sin embargo, los encargados de la producción de invernaderos exigen nuevas y revolucionarias capacidades a este tipo de soluciones y la industria está respondiendo para cumplir las crecientes necesidades de sus clientes. Si la información de negocio está incompleta, dañada o no es válida, los datos corruptos pueden hacer que los encargados de monitorear el estado del invernadero tomen decisiones que en realidad disminuyen el rendimiento y la rentabilidad.

Las organizaciones deben apoyarse en herramientas juntamente con técnicas que combinen la integración, gestión y calidad de datos para que la información clave sea correctamente procesada y analizada, llegando a los usuarios correctos.

Dentro del proyecto de Monitoreo ambiental de la actividad del invernadero ESPE - IASA I: un punto de fallo son los concentradores de métricas medioambientales. Los posibles fallos pueden ser del propio hardware del concentrador: no pudiendo recolectar las métricas de

los sensores o de algún proceso crítico: ya sea del sistema operativo o la aplicación que se esté ejecutando en local.

Los motivos de la elección de Kafka considerados es que es una aplicación de código abierto es decir libre distribución y de mucho rendimiento que actualmente es utilizado por empresas internacionales de renombres. Con Zookeeper conjuntamente hacen de Kafka una gran alternativa para el procesamiento de datos en tiempo Real.

1.4 OBJETIVOS

1.4.1 Objetivo General

Diseñar y construir un sistema embebido para el seguimiento de los datos de producción de un invernadero de la ESPE-IASA I utilizando técnicas de big data como Apache Kafka en Raspberry PI en una red de sensores WSN aplicando la metodología Ralph Kimball.

1.4.2 Objetivos Específicos

- Diseñar la arquitectura apropiada del prototipo para el monitoreo ambiental de la actividad del invernadero ESPE-IASA I.
- Planificar y analizar los requerimientos para el monitoreo ambiental de la actividad del invernadero ESPE-IASA I.
- Construir un Bridge de comunicación entre los Sensores y la Raspberry Pi con Python.
- Diseñar el modelo multidimensional Big-Data de los datos monitoreados.
- Realizar el proceso ETL (Extract, Transform and Load) para el modelo multidimensional
- Proporcionar un prototipo Web en tiempo real de los datos del invernadero aplicando herramientas de monitoreo streaming Open-Source.

1.5 ALCANCE

Entrando en un contexto técnico, el desarrollo del prototipo va a cumplir con los siguientes requerimientos:

- Implantación del Sistema Empotrado para la obtención de datos del Invernadero.

- Se dispondrá de una base de datos servidor alojada en la Raspberry Pi, cuyos datos obtenidos por una red de sensores WSN con 4 sensores de ambientes: de temperatura y humedad, de luminosidad, de índice ultravioleta y de calidad de aire, con 2 nodos que permita el almacenamiento necesario para para el monitoreo ambiental de la actividad del invernadero ESPE-IASA I.
- Para la realización de la transmisión de datos en tiempo real se utilizará software y hardware libre
- Desarrollo de un Módulo (prototipo) aplicación Web, para la presentación de datos que permita monitorear la producción del invernadero: Aquí se presentan Dashboard en tiempo real.
- Manual técnico para la gestión del monitoreo de Datos en tiempo real y la interpretación de las métricas tomadas.

1.6 FACTIBILIDAD

1.6.1 Factibilidad Técnica

Para realizar el proyecto se cuenta con el equipo computacional de los estudiantes a cargo del proyecto.

El tesista cuenta por su parte con su computador con el rendimiento necesario para soportar las herramientas que se van a utilizar, descritas en la sección anterior.

HARDWARE

- Raspberry Pi Model B
- Arduino Mega
- Arduino Uno
- Sensor de Temperatura y Humedad (DHT22)
- Sensor de Luminosidad (MH Series Flying Fish)
- Sensor de índice Ultravioleta
- Sensor de Calidad de Aire (MQ135)
- Antenas de Transmisión y Recepción de Datos (NRF22L01)
- Cable USB
- Cable UTP

- MicroSD 32 Gigabytes Tipo 10X
- Laptop
- Impresora

SERVIDORES

Los servidores son las siguientes:

- Servidor de Base de Datos
- Servidor Web

SOFTWARE

- Sistema Operativo Debian
- PhpMyAdmin
- Gestor de base de datos MariaDB
- Apache Kafka
- Apache Avro
- Apache Zookeeper
- Arduino Studio
- Rasbian Stretch whit Desktop
- Python
- Grafana
- Elasticsearch
- Logstash

1.6.2 Factibilidad Económica

El costo del proyecto por cada nodo (equipo completo de monitoreo) oscila un monto de $367,00 dólares los cuales se desglosan a continuación. (Ver Tabla 1).

Tabla 1

Requerimientos del Hardware

HADWARE			
Equipo	**Cantidad**	**Costo Unitario**	**Costo Total**
Raspberry Pi 3 Model 3	1	$85,00	$85,00
Ardruino Mega	1	$ 25,00	$ 25,00
Sensor de Temperatura y Humedad (DHT22)	1	$ 15,00	$ 15,00
Sensor de Luminosidad (MH Series Flying Fish)	1	$ 4,00	$ 4,00
Sensor de índice Ultravioleta	1	$ 14,00	$ 14,00
Sensor de Calidad de Aire (MQ 135)	1	$ 7,00	$ 7,00
Antenas de Transmisión y Recepción de Datos (NRF22L01)	1	$ 44,00	$ 44,00
Memoria Micro SD 64 Gb Clase10	1	$ 32,00	$ 32,00
Conversor HDMI to VGA	1	$ 25,00	$ 25,00
Rollo de cable UTP 100m	2	$ 23,00	$ 56,00
Rollo de cable Eléctrico 100m No. 14	2	$ 30,00	$ 60,00
TOTAL			**$367,00**

1.6.3 Factibilidad Legal

En el desarrollo del presente trabajo de titulación que es un sistema embebido se estarán utilizando software generalmente libre como se propuso en el proyecto, pero si es necesario contratar servidores se procederá a la obtención y cancelar por dichos servicios.

1.6.4 Factibilidad Operativa

Por parte del tesista, se cuenta con los conocimientos y los recursos como libros y artículos necesarios para implementar el sistema, así como también se cuenta con el apoyo de docentes especializados en el tema de comunicación para conseguir los objetivos planteados en el proyecto. Los estudiantes Cristian José Barba Barba y René Alexander Carrera.

CAPÍTULO II
MARCO TEÓRICO

Para el presente proyecto de investigación es fundamental plasmar los conceptos de hardware y de software utilizados en la investigación para cual se detalla los elementos de la investigación.

2.1 SISTEMAS EMBEBIDOS

Un sistema embebido o empotrado (embedded system) puede definirse como un sistema computador de propósito especial integrado en un sistema de ingeniería más general que realiza una o varias funciones específicas, en general, cumpliendo una serie de requisitos funcionales. (Raghavan, Amol, & Neelakandan, 2006)

"Los sistemas empotrados tienen casi las mismas funciones que un computador personal, excepto que son construidos para aplicaciones muy específicas. Además, estos sistemas requieren de un sistema operativo capaz de soportar exigencias de tiempo real y que sea confiable para la manipulación de datos, dando respuestas en los intervalos de tiempo establecidos. Para cumplir con esta exigencia, dichos sistemas operativos deben ser predecibles (deterministas). En el mercado existe cierta variedad de estos sistemas de tiempo real, pero en su mayoría son costosos y cualquier necesidad en particular por parte de ellos requiere un costo adicional, por lo que es más útil recurrir a un sistema de tiempo real basado en software libre, con las ventajas asociadas en costos y en la capacidad de adaptación a necesidades específicas." (Mejía, Ríos, León, & Hidrobo, 2010)

Muchos de estos sistemas están enfocados a realizar una única tarea o un conjunto muy limitado de tareas a las que en algunos casos se exige restricciones de funcionamiento en tiempo real, coste, tamaño, consumo, etc. (García Moya & Barriga Barros, 2012)

2.1.1 Características de los sistemas embebidos.

En contraste con el computador personal, que también está formado por una combinación de hardware y software, no está diseñado para un uso específico, y es posible darle muchos usos diferentes, de hecho, se consideran como dispositivos de propósito general.

Hay muchas variables a considerar con respecto a este tema, pero vamos a mencionar unas pocas:

- Concurrencia: todos los componentes del sistema monitoreado recurren al sistema embebido al mismo tiempo y este último debe actuar en consecuencia.
- Eficacia: deben responder con gran rapidez a los cambios en el sistema controlado.
- Bajo Consumo: estos sistemas generalmente son de bajo consumo (mayor autonomía). Precio Bajo: son de precios relativamente bajos dadas sus funcionalidades, esto varía según la empresa distribuidora.
- Tamaño Pequeño: Los sistemas empotrados tienen muy pocos recursos de memoria y E/S. El sistema empotrado debe ajustarse al sistema que monitorea.

Un sistema embebido generalmente consiste en una tarjeta de circuito impreso que soporta un microcontrolador, el cual tiene grabado en su memoria de código el programa de control y una serie de componentes auxiliares. Dicha tarjeta se conecta a la fuente de alimentación y a los dispositivos de entrada salida que gobiernan el sistema embebido. (Adhikari, 2013).

2.1.2 Arquitectura de los sistemas embebidos

No hay una arquitectura general disponible que satisface todas las necesidades de un sistema embebido, sin embargo, intrínsecamente involucra una combinación de uno o más tipos de arquitectura de software parcial similares a la de un PC, a continuación, se detalla los elementos básicos:

- **Procesador:** un procesador es el cerebro principal dentro de cualquier sistema embebido. Este es un factor importante que afecta el rendimiento del sistema. Existen diferentes procesadores disponibles en el mercado. Un sistema embebido puede utilizar un microprocesador o un microcontrolador. El procesador viene en diferentes arquitecturas como 8-bit, 16-bit y 32-bit. El procesador de 8 bits se utiliza generalmente en una pequeña aplicación donde necesitamos algo de computación básica como entrada y salida sin procesamiento pesado. Para aplicaciones de gama alta en las que el rendimiento es importante y se necesita una interfaz gráfica de usuario, utilizamos procesadores de 16 o 32 bits

- **Memoria**: si estamos usando un microcontrolador como AT89s51, AT89s52 o ATmega. La memoria está disponible en el chip. Generalmente hablamos de dos tipos de memoria en los sistemas embebidos.

  La memoria RAM es una memoria volátil y se utiliza para el almacenamiento temporal de los datos. Y la selección de la misma depende de la necesidad del usuario y de la aplicación.

  La memoria ROM o memoria de códigos. Se utiliza para el almacenamiento del programa. Una vez que el sistema es alimentado, el sistema obtiene el código de la memoria ROM. La EEPROM es una memoria única. El contenido puede ser borrado y reprogramado por una entrada de pulsos de alto voltaje. Esto se utiliza para almacenar los datos por el propio programa. Supongamos que tenemos un registrador de datos de temperatura. Y necesita almacenar los datos cada una hora. Esto significa que necesitamos los datos en tiempo de ejecución después de que se inicie el sistema.

  El sistema leerá la temperatura y la almacenará en la memoria EEPROM. Y será permanente. Y puedes recuperar los datos más tarde. Así que un desarrollador de sistemas embebidos decide qué memoria usar para su aplicación.
- **Puertos de comunicación**: El hardware de los sistemas embebidos tiene diferentes tipos de puertos de comunicación para comunicarse con los otros dispositivos embebidos. Por ejemplo, la popular placa Arduino UNO tiene ATmega328 IC y tiene el siguiente puerto de comunicación (TAR, I2C, SPI). Para el envío de datos de una placa a otra podemos utilizar estos protocolos serie. Pero para eso, necesitamos programarlo.
- **Entrada y salida**: Para interactuar con los sistemas embebidos necesitamos entrada de datos. La entrada puede ser proporcionada por el usuario o por algún sensor. A veces algunos sistemas necesitan más entrada o salida. Por lo tanto, la selección del procesador se basará en E/S. Estas entradas y salidas se dividen generalmente en puertos como P0, P1, P2 y P3 en microcontroladores 8051. Y PA, PB, PC y PD en la serie ATmega del microcontrolador. Las E/S deben configurarse para entrada o salida basándose en el registro proporcionado. Y para eso, necesitamos referirnos a la hoja de datos del fabricante.

- **Circuitos de Aplicaciones Específicas:** Algunos componentes de hardware son comunes durante el diseño de los sistemas embebidos. Pero algunos son diferentes y dependen de la necesidad de la aplicación. Al igual que un sensor de temperatura, necesita un sensor de temperatura para detectar la temperatura. Mientras que otros manuales un detector de alcohol tiene un sensor para detectar el nivel de alcohol. (Max FIZZ Technologies, 2018)

2.2 ARDUINO

Arduino es una plataforma de código abierto fácil de usar. Arduino tiene una aplicación se utiliza un microcontrolador y un entorno de desarrollo integrado (IDE) para programarlo. La placa Arduino puede ser programada directamente desde el PC usando FTDI, lo cual es fácil comparado con otras plataformas similares.

Las ventajas son las siguientes:

- **Circuitos de Aplicaciones Específicas:** Algunos componentes de hardware son comunes durante el diseño.
- **Bajo costo:** Las placas Arduino son de coste relativamente bajo en comparación con otras plataformas de microcontroladores.
- **Multiplataforma:** El software de Arduino (IDE) es compatible con la directiva Sistemas operativos Windows, Macintosh OSX y Linux.
- **Fácil de usar:** El software de Arduino (IDE) es amigable y fácil de usar, uso para principiantes y muy flexible para programadores expertos.
- **Código abierto:** El Arduino es un software de código abierto y puede ser programado con lenguajes C, C++ o AVR-C. Así, una variedad de módulos puede ser diseñado por los usuarios.

La plataforma Arduino está compuesta por un microcontrolador. Se puede conectar a un PC a través de un cable USB. Es de libre acceso y se puede descargar fácilmente.

También puede ser modificado por un programador. Diferentes versiones de Arduino están disponibles en el mercado dependiendo de las necesidades del usuario.

2.2.1 Arduino Uno

El Arduino/Genuino Uno tiene un microcontrolador ATmega328 integrado. Dispone de seis puertos de entrada analógica (A0-A5). Cada pin puede operar a 0-5 V. Tiene 14 pines digitales de entrada/salida (I/O) de los cuales 6 son de salida PWM, 6 analógicas, 2 KB SRAM, 1 KB EEPROM, y opera a 16 MHz de frecuencia (Figura 1 y Tabla 2).:

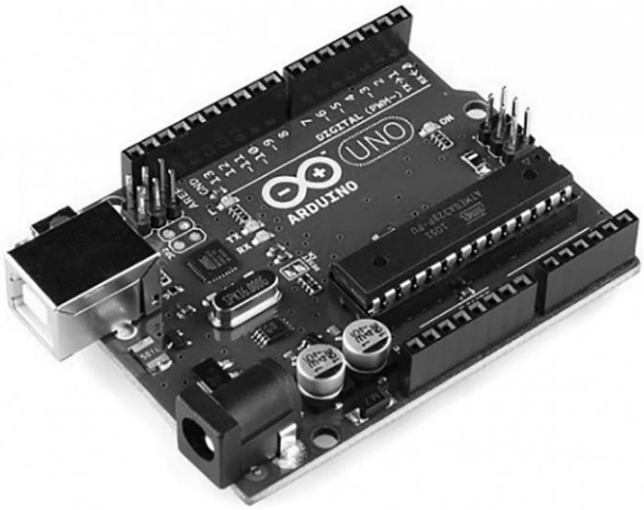


Figura 1 Arduino Uno
Fuente: (Singh, Gehlot, Singh, & Choudhury, 2017)

Tabla 2

Descripción de los pines del Arduino Uno

Pin	Descripción
Vin	El voltaje externo de la placa Arduino
5 V	+5 V salida regulada
3. 3	V A bordo 3.3 V alimentación
GND	Tierra
IOREF	Proporciona la referencia de tensión y selecciona la fuente de alimentación adecuada
Serial	Transmite y recibe datos seriales, Pins: 0(Rx) 1(Tx)
Interrupciones externas	Disparar una interrupción en valor bajo, Pines: 2 y 3

CONTINÚA

PWM	Proporciona salida PWM de 8 bits, Pines: 3,5,6,9,10,11
SPI	Soporta comunicación SPI, Pins: 10 (SS), 11 (MOSI), 12 (MISO),y 13 (SCK)
LED	impulsado por Pin 13
TWI	Soporta comunicación TWI, Pins: A4 (SDA), A5 (SCL)
AREF	Tensión de referencia para las entradas analógicas
Reset	Se utiliza para reiniciar el microcontrolador integrado.

2.2.2 Arduino Mega

El Arduino Uno tiene incorporado el microcontrolador ATmega2560. Tiene 16 entradas analógicas, 54 E/S digitales, conexión USB, 4 UART, conector de alimentación y un botón de reinicio. Funciona a una frecuencia de 16 MHz. La tarjeta puede funcionar con 5-12 V de alimentación externa; si se suministra más de esto, puede dañarla. Tiene una memoria flash de 256 KB, 8 KB SRAM y 4 KB EEPROM. (Figura 2 y Tabla 3).

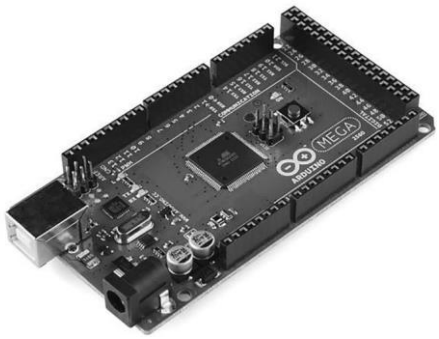


Figura 2 Arduino Mega
Fuente: (Singh, Gehlot, Singh, & Choudhury, 2017)

Tabla 3

Descripción de los pines del Arduino Mega

Pin	Descripción
Vin	El voltaje externo de la placa Arduino
+5V	Salida regulada de +5V
3.3V	Entrada Alimentación 3.3V

CONTINÚA

GND	Tierra
IOREF	Proporciona la referencia de voltaje y selecciona una potencia apropiada de partida
Serial0	Transmite y recibe datos seriales, Pins: 0(Rx) 1(Tx)
Serial1	Transmite y recibe datos seriales, Pines: 19(Rx) 18(Tx)
Serial2	Transmite y recibe datos seriales, Pins: 17(Rx) 16(Tx)
Interrupciones externas	Dispara una interrupción en valor bajo, Pines: 2 (interrupción0), 3 (interrupción1),18 (interrupción5), 19 (interrupción4), y 20 (interrupción2)
PWM	Proporciona salida PWM de 8 bits, Pines: 2-13 y 44-46
SPI	Soporta comunicación SPI, Pins: 53 (SS), 51 (MOSI), 50 (MISO), y 52 (SCK)
LED	LED accionado por Pin 13
TWI	Soporta comunicación TWI, Pines: 20 (SDA), 21 (SCL)
AREF	Tensión de referencia para las entradas analógicas
Reset	Este se utiliza para reiniciar el microcontrolador integrado.

2.3 RASPBERRY PI

La Raspberry Pi es una serie de computadoras del tamaño de una tarjeta de crédito, con tablero simple creadas en el Reino Unidos por la Fundación Raspberry Pi con el propósito de promover y fomentar la enseñanza de informática básica. La fundación desarrolla recursos y herramientas gratuitas para ayudar a la gente a educarse sobre informática, y cómo hacer las cosas con las computadoras. El inicio de Raspberry Pi comenzó en 2006.

El 19 de febrero de 2012 se anunciaron dos modelos: Modelos A y B. El modelo B+ se anunció en julio de 2014. Pi 3 Modelo B se anuncia el 29 de febrero de 2016 (Raspberry Pi Foundation, n.d.). Raspberry Pi es un minicomputador de bajo costo. Es posible conectar Monitor de PC así como televisión a la Raspberry Pi. El teclado y mouse se pueden conectar también a la Rapsberry Pi. Todos los modelos que tienen un sistema Broadcom (conjunto de sistemas integrados) en un chip, incluyen una unidad de procesamiento central (CPU) compatible con ARM y una unidad de procesamiento gráfico en el chip. Los intervalos de velocidad de la CPU son de 700 MHz a 1,2 GHz para el Pi 3. El rango de memoria integrada

es de 256 MB a 1 GB de RAM. Las tarjetas Secure Digital (SD) se utilizan para almacenar la información del sistema operativo.

La mayoría de placas de Raspberry Pi tienen puertos USB, entrada HDMI, puerto DSI, entrada a audio, 40 pines GPIO, Bluetooth incorporado, WIFI, entre otras entradas.

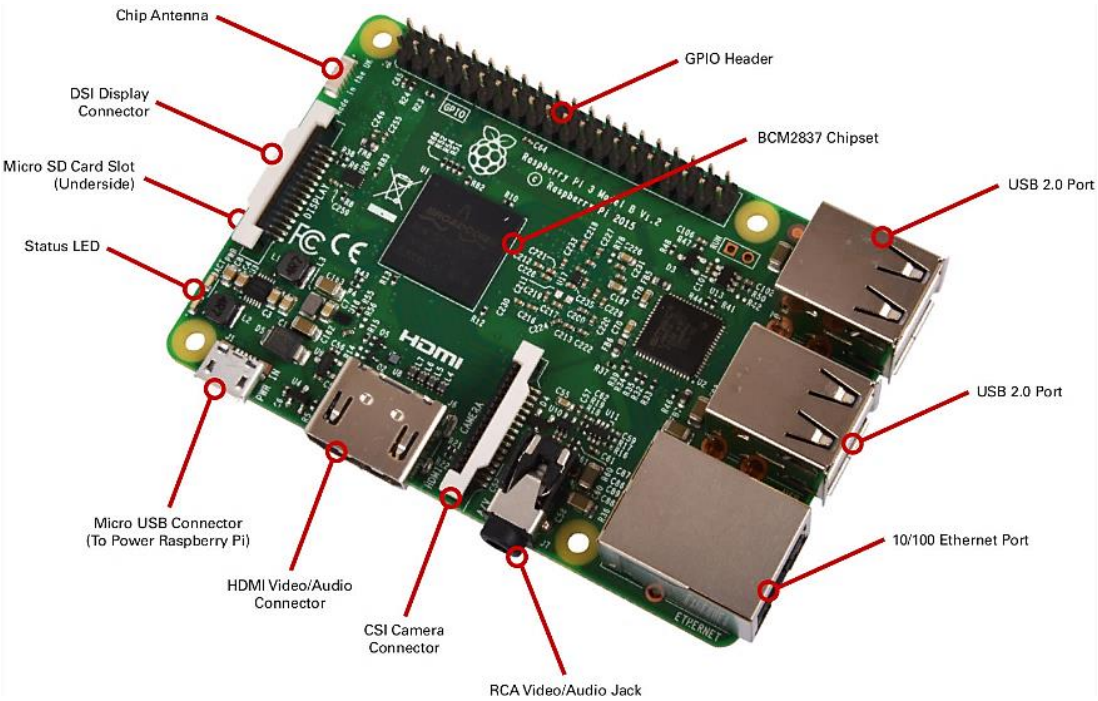


Figura 3 Rasbperry Pi 3 Model B
Fuente: (CNXSOFT, 2016)

Rapsberry Pi tiene su propio sistema operativo llamado Raspbian que es una derivación de Debian, Ubuntu mate, snappy Ubuntu, Pidora, Linutop, Arch Linux ARM y así entre otros son los varios sistemas operativos usados para la Raspberry Pi.

Raspberry Pi soporta diferentes lenguajes de programación como C++, Python, SQL y HTSQL. Utiliza C++ para la programación Arduino.HTSQL (Hyper Text Structured Query Lenguaje) para proporcionar una interfaz web amigable a la base de datos que es sencillo de consultar a través del navegador web. También soporta lenguajes de programación como java, java script, php entre otros.

2.3.1 Módulos de Potencia de Raspberry Pi

La Raspberry Pi tiene cuatro modos de potencia distintos que son los estados en que se puede encontrar el equipo dependiendo la necesidad que el usuario requiera disponer del equipo (Digi-Key Electronics, s.f.):

- **El modo de ejecución:** la unidad central de proceso (CPU) y todas las funciones del núcleo ARM11 están disponibles y encendida.
- **El modo de espera:** los relojes del núcleo principal se apagan (las partes de la CPU que las instrucciones de proceso ya no están funcionando) aunque los circuitos de energía en el núcleo siguen activos. En este modo, conocido como En el modo "Wait for Interrupt" (WFI) o “espera a que se interrumpa” en español, el núcleo puede ser rápidamente encendido por un proceso que genera una especie de llamada a la CPU o llamada interrupción. Esta interrupción detendrá cualquier proceso actual y hará lo que el proceso de llamada ha solicitado.
- **El modo de apagado:** no hay energía y por ende el equipo no realiza ninguna función.
- **El modo inactivo**: el núcleo está apagado y todas las cachés se dejan encendidas.

2.3.2 Comparativa de Modelos de Raspberry Pi

Cuando el momento en que la cuestión de porque se van a utilizar Raspberrys Pi 3 Modelos B se hace presente, es indispensable hacer énfasis de las necesidades que requiere el proyecto y que modelo puede solventar todas o la mayoría de los requerimientos.

A continuación, se muestran las características de los modelos anteriores en la Tabla 4:

Tabla 4

Comparativa de Modelos de Raspberrys Pi

	Model A+	**Model B+**	**2 Model B**	**3 Model B**
SoC	Broadcom BCM2885	Broadcom BCM2835	Broadcom BCM2836	Broadcom BCM2837
CPU	700MHz ARMI 176JZF-S	700MHz ARM1176JZF-S	900MHz Quad-core ARM Cortex-A7	1.2GHz QUAD ARM Cortex-A53

CONTINÚA

GPU	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV
RAM	256Mb	512Mb	1Gb	1 Gb
USB	1	4	4	4
Video	Jack, HDMI	Jack, HDMI	Jack, HDMI	Jack, HDMt
Audio	Jack, HDMI	Jack, HDMI	Jack, HDM[	Jack, HDMI
Boot	MicroSD	MicroSD	MicroSD	MicroSD
Red	-	Ethernet 10/100	Ethernet 10/100	Eth. 10/100, Wifi, BT
Consumo	400mA / 2w / 5v	500mA / 2,5w / 5v	800mA / 4w / 5v	2,5A / 12,5w / 5v
Alim.	MicroUSB / GAO	MicroUSB / GPIO	MicroUSB / GPIO	MicroUSB / GPIO
Tamaño	65 x 56 mm	85 x 56 mm	85 x 56 mm	85 x 56 mm
Precio	$ 20	$ 35	$ 35	$ 35

Fuente: (Como Hacer, 2014)

El Raspberry Pi 3 Modelo B se caracteriza principalmente porque tiene conexión Wifi y bluetooth, por otro lado, su procesador es superior a las de versiones anteriores y asi brinda soporte de todos los procesos los así que por esas razones se ha elegido este modelo.

2.4 APACHE KAFKA

Apache Kafka es una plataforma de streaming distribuida que se utiliza para procesar datos en tiempo real, esta plataforma se escala de forma elástica en lugar de tener un corredor de mensajes individual para cada aplicación (Shapira, Narkhede, & Palino, 2017). Apache Kafka (Apache Kafka, 2017) no es comúnmente considerado como una base de datos. Por lo general, se considera como un agente de mensajes, es decir, un programa intermediario que transfiere mensajes (unidades de información de propósito general) de forma asincrónica o sincrónica de un programa a otro a través de un tema. En este sentido, ha sido identificada en investigaciones anteriores como una tecnología adecuada para aplicaciones del internet de las

cosas (IoT) (Ta, Liu, & GW, 2016). Sin embargo, los desarrolladores de Kafka en LinkedIn lo implementaron con un mayor alcance de uso en mente, incluyendo el "almacenamiento de datos fuente de la verdad" (Wang, Koshy, & Subramanian, 2015).

2.4.1 Origen de Apache Kafka.

Kafka se creó para abordar el problema de la canalización de datos en Linkedln. Fue diseñado para proporcionar un sistema de mensajería de alto rendimiento que puede manejar muchos tipos de datos y proporcionar datos limpios y estructurados sobre la actividad del usuario y las métricas del sistema en tiempo real, esto declara Jeff Weiner, CEO de LinkedIn

2.4.1.1 Problema de LinkedIn.

Existía en LinkedIn un sistema de recolección y visualización de métricas basadas en sondeos como el rendimiento de las aplicaciones, el uso de memoria en la CPU, que utilizan recolectores que eran personalizadas para guardar y presentar datos que eran almacenados en el interior. Pero en el sistema de monitoreo presentaba demasiadas fallas como los grandes intervalos entre métricas y el impedimento de que los dueños de aplicaciones puedan hacer sus propias métricas. El sistema era considerado algo manual y no automático ya que solicitaba la intervención humana para la ejecución de muchas de las tareas simples.

El seguimiento se basaba en la dosificación horaria de lotes, este procesamiento por lotes no era el modelo adecuado para el seguimiento de alertas en tiempo real. Sin embargo, los datos de monitorización y seguimiento compartían muchos recursos lo que afectaban al rendimiento de las aplicaciones.

Al principio, se investigaron a fondo las soluciones de código abierto existentes para encontrar un sistema nuevo que proporcionara acceso en tiempo real a los datos y se ampliara para manejar la cantidad de tráfico de mensajes necesarios. Se crearon prototipos de sistemas manejando ActiveMQ, pero el inconveniente era que no se podía manejar la gran escala de mensajes, en esta prueba de sistemas se encontraron muchas fallas en ActiveMQ. Con todos estos antecedentes se tomó la decisión de seguir adelante con una infraestructura personalizada para la canalización de datos.

2.4.1.2 Nacimiento de Kafka.

El equipo inicial de desarrollo de LinkedIn incluyó a Neha Narkhede y, más tarde, a Jun Rao ellos dos liderados por Jay Kreps, un ingeniero de software que previamente era responsable del desarrollo y lanzamiento de Voldemort, un sistema de almacenamiento distribuido. Juntos, se propusieron crear un sistema de mensajería que pudiera satisfacer las necesidades de los sistemas de monitoreo y rastreo, y escalares para el futuro. Los objetivos principales fueron:

- Desvincular a consumidores y productores utilizando un modelo push-pull.
- Dentro del sistema de mensajería suministrar persistencia a los para que existan múltiples consumidores.
- Que la estabilidad horizontal del sistema sea permitida para que crezca a medida que crecen los flujos de datos.

El resultado de estos principales objetivos fue un sistema de mensajería de publicación/suscripción que tenía una interfaz común de los sistemas de mensajería, pero con la gran ventaja de almacenamiento más parecida a un sistema de agregación de registros. Combinado con la adopción de Apache Avro para la serialización de mensajes, Kafka fue efectivo en el seguimiento de la actividad de los usuarios y manejo de métricas a una escala de miles de millones de mensajes al día.

2.4.1.3 Código Abierto.

A finales del 2010, Kafka fue impulsado como un proyecto de código abierto en GitHub. En Julio del 2011 fue propuesto y aceptado como un proyecto experimental de la Apache Software Foundation a medida que comenzó a ganar atención en la comunidad de código abierto. Apache Kafka dejó de ser un proyecto experimenta y se consolidó en octubre de 2012. Desde entonces, se ha trabajado continuamente y se ha encontrado una sólida comunidad de contribuyentes y comprometidos fuera de LinkedIn. En el 2014, Jay Kreps, Neha Narkhede y Jun Rao dejaron LinkedIn para fundar Confluent, una empresa centrada en el desarrollo, el apoyo empresarial y la formación de Apache Kafka.

2.4.1.4 ¿Por qué el nombre “Kafka”?

La gente a menudo pregunta cómo Kafka consiguió su nombre y si tiene algo que ver con la aplicación en sí. Jay Kreps su fundador ofreció la siguiente reflexión: Pensé que como Kafka era un sistema optimizado para la escritura, el uso de un nombre de escritor tendría sentido. Había tomado muchas clases de literatura en la universidad y me gustaba Franz Kafka. Además, el nombre sonaba bien para un proyecto de código abierto. Así que básicamente no hay mucha relación nada más gusto del fundador por su afinidad al escritor Franz Kafka.

2.4.2 Características de Apache Kafka.

Apache Kafka se caracteriza principalmente por las siguientes particularidades:

- **Varios Productores:** Kafka es capaz de manejar sin problemas a múltiples productores, ya sea que esos clientes estén usando muchos temas o el mismo tema. Esto hace que el sistema sea ideal para agregar datos de muchos sistemas frontales y hacerlos consistentes.
- **Múltiples Consumidores:** Kafka está construido para que cualquier flujo de mensajes sean leídos por múltiples consumidores sin interferir entre sí. Esto marca una diferencia con muchos sistemas de colas, donde una vez que un mensaje es consumido por un cliente, no está disponible para ningún otro.
- **Retención basada en disco:** Los mensajes se envían al disco y se almacenan con reglas de retención configurables. La retención basada en disco significa que, si un consumidor se queda atrás, ya sea debido a una ráfaga de tráfico o por un lento procesamiento, no hay riesgo de perder datos.
- **Escalable**: La escalabilidad flexible de Kafka facilita el manejo de cualquier cantidad de datos. Los clústeres que necesitan tolerar más fallas simultáneas pueden configurarse con factores de replicación más altos.
- **Alto rendimiento**: Todas las anteriores características se unen para hacer de Apache Kafka un sistema de mensajería de publicación/suscripción con un excelente rendimiento bajo una alta carga, esto se puede hacer a la vez que se proporciona una latencia de mensajes en segundos desde la producción de un mensaje hasta la disponibilidad para los consumidores.

- **Soporte para varias plataformas de desarrollo**: Plataformas como Python, .Net, Java, PHP, entre otros soporta el sistema de Apache Kafka con una fácil integración de estas plataformas para sus clientes.

2.4.3 Arquitectura de Apache Kafka.

La arquitectura de publicación-suscripción es la que implementa Kafka, con funcionalidades similares a la de un sistema de archivos distribuido combinadas con la de un sistema de mensajería convencional, La figura 1 muestra una visión general de la arquitectura Kafka y sus 4 elementos primordiales que interactúan:

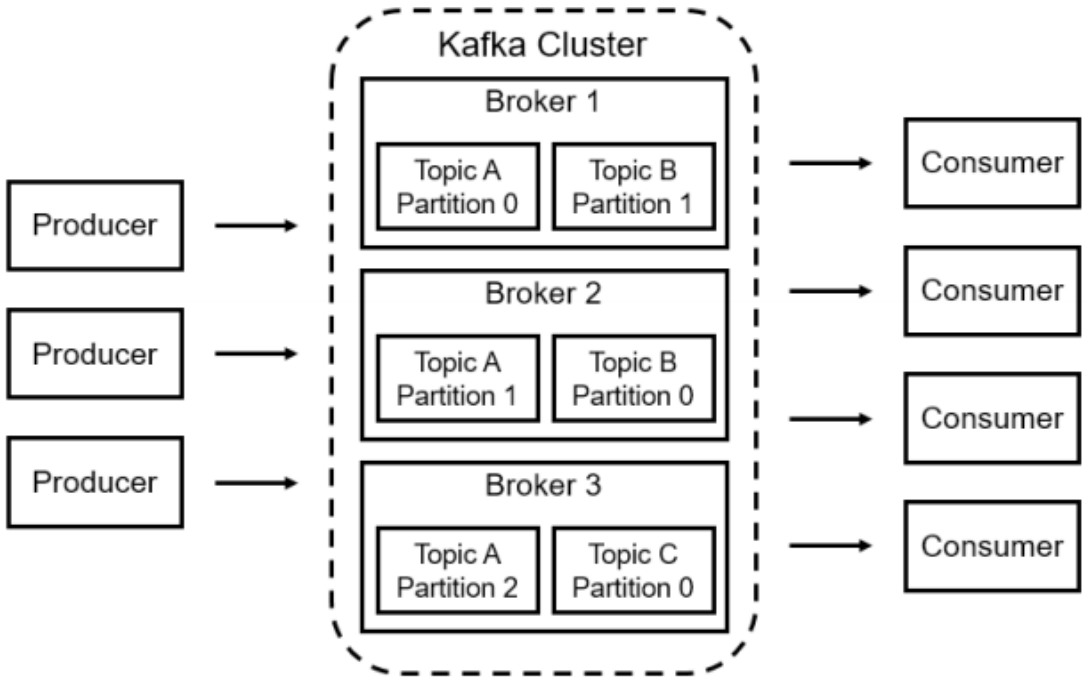


Figura 4 Arquitectura de Apache Kafka
Fuente: (Clarke, Akeroyd, & Moore, 2017)

2.4.3.1 Topic y Logs

Un Topic (tema) es donde se publican los mensajes. Muchos consumidores pueden suscribirse a un tema para procesar los datos escritos en él. Los consumidores controlan el offset - posición de los registros - y pueden consumir mensajes en cualquier orden, ya sea restableciendo un offset más antiguo o saltando al más reciente, como puede verse en la figura 9. Esto permite al consumidor la flexibilidad de reprocesar datos antiguos o saltar al log (registro) más reciente e inmediatamente obtener los datos actuales. (Apache Kafka, 2017)

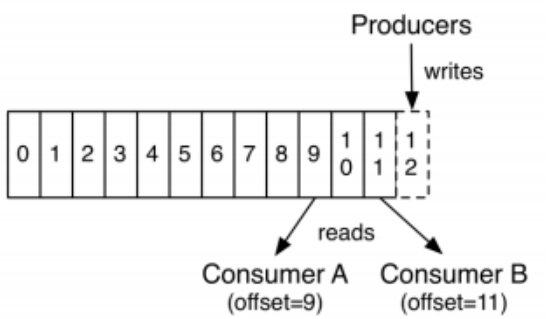


Figura 5 Registro de productores y consumidores
Fuente: (Huong, 2017)

Según Apache (2017), el clúster Kafka mantiene un registro estructurado de confirmaciones para cada tema. Como puede verse en la figura 10, cada partición de dicho tema se mantiene en una secuencia de registros agregados continuamente.

Las particiones aquí se distribuyen entre los servidores en el clúster Kafka mientras se replican entre los servidores para la tolerancia a fallos. Los corredores de Kafka son apátridas: no rastrean el consumo, dejando la eliminación de mensajes a una política de retención configurable.

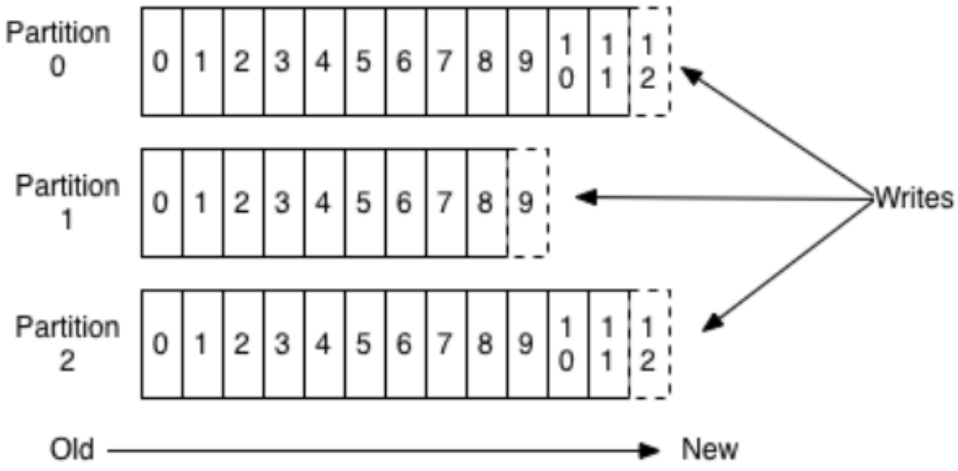


Figura 6 Anatomía de un tema de Kafka
Fuente: (Apache Kafka, 2017)

2.4.3.2 Broker

Los broker (corredores) son nodos de almacenamiento primario que constan de toda la red de colas. Reciben los datos enviados por los productores de datos, los almacenan y los envían cuando los consumidores los solicitan. En un sistema tradicional de brokers de Kafka, un grupo de máquinas que ejecutan el sistema ZooKeeper mantendrá la coordinación, partición de datos y procesamiento de información de compensación de consumo y tolerancia a fallas para todos los nodos del broker.

Una partición puede ser asignada a múltiples brokers, lo que resultará en que la partición sea replicada (como se muestra en la Figura 1-7). Esto proporciona redundancia de mensajes en la partición, de modo que otro corredor puede asumir el liderazgo si hay una falla del broker. Sin embargo, todos los consumidores y productores que operan en esa partición deben conectarse con el líder.

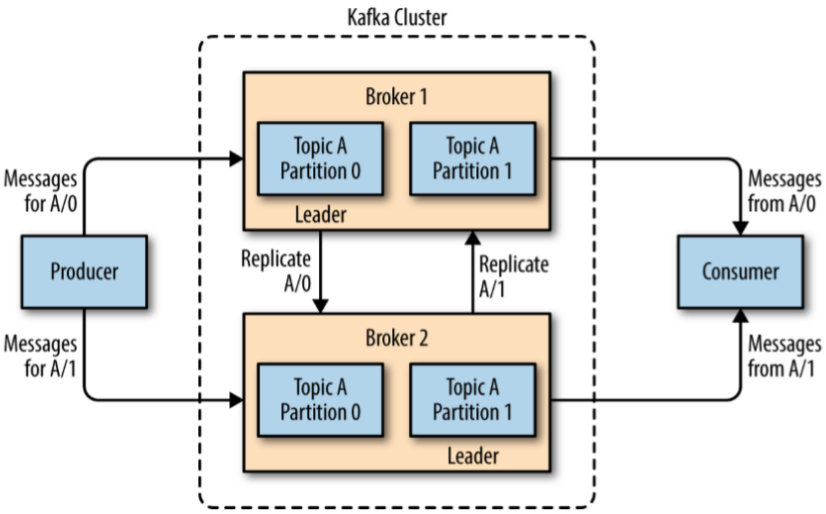


Figura 7 La replicación de particiones en un clúster

Fuente: (Shapira, Narkhede, & Palino, 2017)

2.4.3.3 Producers

Los producers (productores) son contribuyentes de datos primarios que producen mensajes y los empujan a la cola de mensajes para que los consumidores de datos puedan recuperarlos más tarde. Los productores se comunican directamente con uno de los brokers del sistema de colas y obtienen información sobre la partición y división de los datos salientes de

los mensajes y los almacenan en los nodos correspondientes dentro de la nube de colas. Cuando se almacenan datos, primero se debe establecer un topic y los consumidores recuperan todos los datos dentro de ese topic.

2.4.3.4 Consumers

Los consumidores se suscriben a un determinado tema y recuperan todos los mensajes disponibles almacenados bajo ese tema. Cada consumidor de un grupo de consumidores recibirá datos de uno o más corredores que almacenan mensajes sobre el tema solicitado. El número de consumidores no puede ser mayor que el número de particiones concedidas a ese tema.

Los consumidores trabajan como parte de un grupo de consumidores, que es uno o más consumidores que trabajan juntos para consumir un tema. El grupo asegura que cada partición sólo sea consumida por un miembro. En la Figura 1-6, hay tres consumidores en un solo grupo que consumen un tema. Dos de los consumidores trabajan desde una partición cada uno, mientras que el tercer consumidor trabaja desde dos particiones.

De esta manera, los consumidores pueden escalar horizontalmente para consumir temas con un gran número de mensajes. Además, si un solo consumidor falla, los miembros restantes del grupo reequilibrarán las particiones que se están consumiendo para hacerse cargo del miembro faltante.

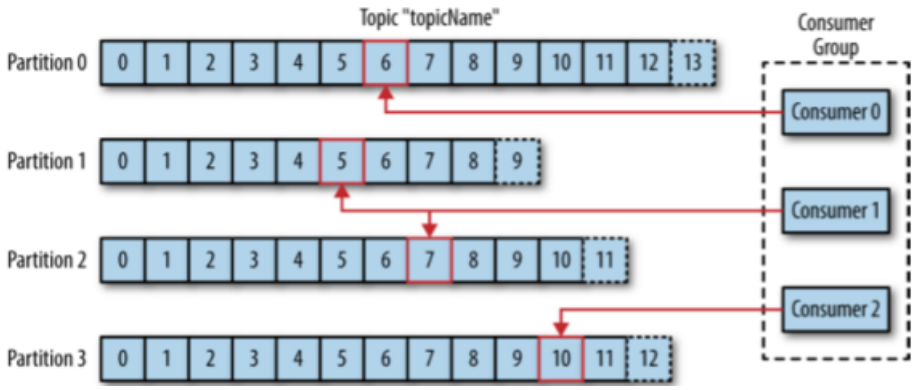


Figura 8 Un grupo de consumidores lectura de un tema

Fuente: (Shapira, Narkhede, & Palino, 2017)

2.5 APACHE ZOOKEEPER

Para ayudar a mantener un grupo de servidores Kafka, Apache Kafka utiliza Zookeeper para ayudar a hacer un seguimiento de los metadatos de los clústeres y de la información relativa a consumidores. Zookeeper trabaja como administrador de valores clave, lo que puede ayudar a los aspectos de sincronización para Kafka, como la elección del líder, la detección de fallos, gestión de grupos de miembros y gestión de metadatos (Junqueira & Reed, 2013).

Zookeeper y Kafka trabaja en simbiosis, donde Zookeeper intenta ofrecer más control sobre los problemas que surgen en un sistema con múltiples clústeres.

Ejemplos de fallos que pueden producirse cuando se ha distribuido la coordinación de múltiples servidores, es que los mensajes de un proceso a otro pueden ser retrasado, la sincronización de reloj de diferentes servidores puede conducir a decisiones incorrectas sobre cuándo ha llegado un determinado mensaje (Shapira, Narkhede, & Palino, 2017).

2.5.1 Origen de Apache ZooKeeper.

ZooKeeper fue diseñado para ser un servicio robusto que permite a los desarrolladores de aplicaciones enfocarse principalmente en la lógica de sus aplicaciones más que en la coordinación. Expone una API simple, inspirada en la API del sistema de archivos, que permite a los desarrolladores implementar tareas de coordinación comunes, como la elección de un servidor maestro, la gestión de la pertenencia a un grupo y la gestión de metadatos. ZooKeeper es una librería de aplicaciones con dos implementaciones principales de las APIs-Java y C-y un componente de servicio implementado en Java que se ejecuta en un conjunto de servidores dedicados. Tener un conjunto de servidores permite a ZooKeeper tolerar fallas y escalar el rendimiento.

2.5.1.1 Misión de Apache ZooKeeper

ZooKeeper permite tareas de coordinación para sistemas distribuidos. Una tarea de coordinación es una tarea que implica múltiples procesos. Tal tarea puede ser para fines de cooperación o para regular la controversia.

2.5.1.2 ¿El Origen del Nombre “ZooKeeper”?

ZooKeeper fue desarrollado en Yahoo! Research. En ese momento el grupo de fundadores había estado trabajando con el equipo de Hadoop y ha comenzado una variedad de proyectos con los nombres de animales, siendo el Apache Pig el más conocido. Como estaban pensando en diferentes nombres posibles, uno de los miembros del grupo mencionó que debían evitar otro nombre de animal porque su gerente pensó que estaba empezando a sonar como si viviéramos en un zoológico. Fue entonces cuando hizo énfasis: los sistemas distribuidos son un zoológico. Son caóticos y difíciles de y ZooKeeper está destinado a mantenerlos bajo control.

2.5.2 Características de Apache Zookeper.

Al programar con ZooKeeper, los desarrolladores diseñan sus aplicaciones como un conjunto de clientes que se conectan a los servidores de ZooKeeper e invocan operaciones en ellos a través de la API de cliente de ZooKeeper. Entre los puntos fuertes de la API de ZooKeeper, proporciona:

- Fuerte consistencia, orden y durabilidad garantizan
- La capacidad de implementar primitivas de sincronización típicas
- Una manera más simple de tratar muchos aspectos de la concurrencia que a menudo conducen a un comportamiento incorrecto en sistemas distribuidos reales.

2.5.3 Arquitectura de Apache ZooKeeper.

Bien ahora necesitamos entender mejor cómo funciona realmente el servicio. Las aplicaciones hacen llamadas a ZooKeeper a través de una biblioteca cliente. La biblioteca cliente es responsable de la interacción con los servidores de ZooKeeper.

La Figura 2-5 muestra la relación entre clientes y servidores. Cada cliente importa la biblioteca del cliente, y luego puede comunicarse con cualquier nodo de ZooKeeper.

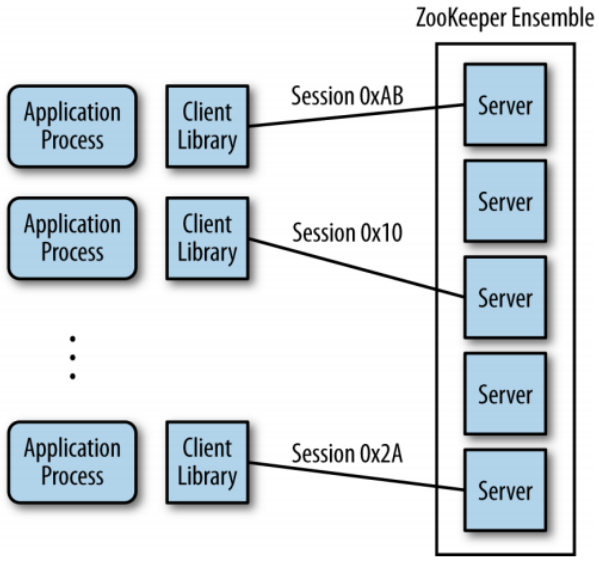


Figura 9 Visión general de la arquitectura de ZooKeeper

Fuente: (Shapira, Narkhede, & Palino, 2017)

Los servidores ZooKeeper funcionan en dos modos: autónomo y quórum. El modo autónomo es más o menos lo que dice el término: hay un solo servidor, y el estado de ZooKeeper no es replicado. En modo quórum, un grupo de servidores de ZooKeeper, que llamamos ZooKeeper ensamble, replica el estado, y juntos sirven las peticiones de los clientes. A partir de este punto no, usamos el término "ZooKeeper ensemble" para denotar una instalación de servidores. Este podría contener un único servidor y funcionar en modo autónomo o contener un archivo de servidores y operar en modo quórum.

2.6 METODOLOGÍA RALPH KIMBALL

El presente estudio se ha basado en el ciclo de vida desarrollado por Ralph Kimball (Kimball, Ross, Becker, Mundy, & Thornthwaite, 2015), en el que se han identificado tres fases diferentes. La primera fase se refiere a una selección de los componentes de hardware y software aplicados. La segunda fase establece el modelo dimensional y la elaboración de los procesos ETL. La tercera fase consistió en las especificaciones de la integración de herramientas de visualización de datos en tiempo real. Adicionalmente, para lograr un desarrollo preciso, Kimball estableció cuatro principios básicos:

- **Enfoque en el negocio,** que identifica los requerimientos del negocio y el valor asociado para desarrollar vínculos con el negocio;

- **Construir una infraestructura adecuada,** para lo cual se debe diseñar una base de información de alto rendimiento, en la que se reflejen los requerimientos del negocio;
- **Entregas en incrementos significativos,** lo que establece un incremento en términos de 6 a 12 meses, resultando en una evolución considerable del proyecto;
- **Ofrecer la solución completa**, en la que se entreguen todos los elementos que aporten valor a los usuarios del negocio. A partir de las consideraciones anteriores, el presente proyecto se ha dividido en dos módulos, cuyo diseño y desarrollo se detallan a continuación en la Figura 10.

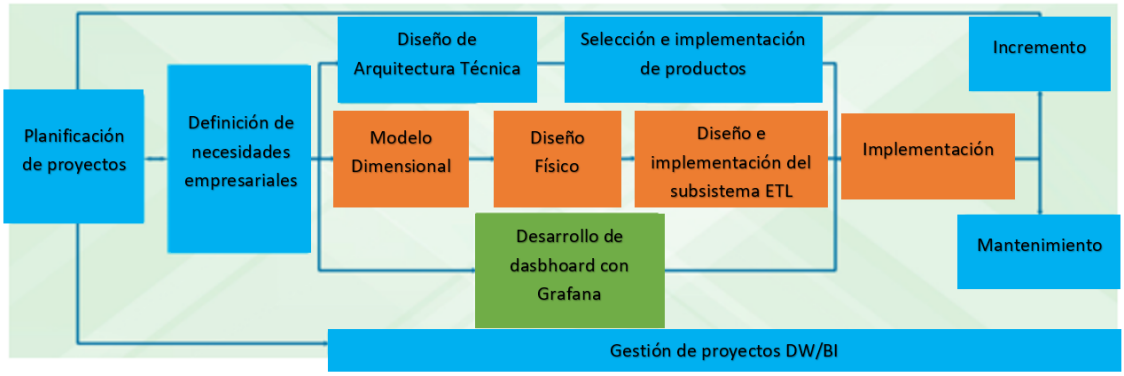


Figura 10 Metodología Ralph Kimball

El Ciclo de Vida de la Metodología Ralph Kimball ensamblado con la integración de herramientas de visualización. Esta combinación ilustrada detalla las diferentes fases para el proceso preciso con el que hemos sido capaces de racionalizar y establecer un sistema que se centra estrictamente en las necesidades del sistema en tiempo real con apache Kafka.

2.7 TRABAJOS RELACIONADOS

Entre los proyectos más sobresalientes que el tesista ha tomado en cuenta y que guardan estricta relación con el actual trabajo de investigación realizado se tienen los siguientes:

- Tema: Distribución de Mensajes en Alta Disponibilidad con Kafka en Raspberrys Pi
- Autor: Sarabia Crespo Roberto
- Universidad: Politécnica de Madrid
- Año**:** 2017
- Resultados o conclusiones importantes: dicho proyecto multidisciplinar tiene como propósito que sirva como base para proyectos en distintos ámbitos, como es la captura de información con procesamiento, vía web también desarrollar aplicaciones con fácil acceso a la información, y también el control del entorno.
  El objetivo principal de esta investigación fue conceder alta disponibilidad a métricas almacenadas usando Linux sobre Raspberrys Pi con tecnología de Apache Kafka.

- Tema: "Big Sensor-Generated Data Streaming Using Kafka and Impala for Data Storage in Wireless Sensor Network for CO2 Monitoring"
- Autores: Rindra Wiska, Novian Habibie, Ari Wibisono, Widijanto Satyo Nugroho, and Petrus Mursanto
- Universidad: Universitas Indonesia
- Año**:** 2016
- Resultados o conclusiones importantes: La red de sensores inalámbricos (WSN) es un sistema que produce una gran cantidad de datos. Por eso, se necesita el sistema que sea capaz de manejar la gran cantidad de datos. Para superar esto, esta investigación integra el sistema WSN estándar con un gran marco de datos con un enfoque de streaming de datos. En la implementación, el sistema utilizó nodos sensores de fabricación propia que recopilan datos de CO2 junto con otros datos ambientales: temperatura del aire, humedad del aire e intensidad de la luz. Utilizando la base de datos Kafka e Impala, este sistema es capaz de transmitir una gran cantidad de datos casi en tiempo real. En general, el sistema propuesto ofrece una buena rendimiento y rendimiento fiable.

CAPÍTULO III
DISEÑO Y CONSTRUCCIÓN DEL SISTEMA EMBEBIDO

Hasta ahora se han descrito el análisis, diseño, conceptos y tecnologías junto con una visión general de las terminologías clave que se utilizan el sistema embebido. Este capítulo comienza explicando la fase de implementación de los nodos de los sensores IoT.

Este capítulo se compone de cuatro secciones en las que el capítulo 3.1 discute el análisis y diseño de las características del sistema embebido muestra la representación general del sistema embebido. El capítulo 3.2 muestra la representación gráfica del sistema embebido con la de Raspberry Pi y sus dos nodos en forma de un diagrama sistemático junto con los componentes de hardware que se utilizan para construir el sistema en primer lugar. El capítulo 3.3 describe el procedimiento paso a paso para configurar el sistema sobre la Raspberry Pi. Esta sección continúa explicando el procesamiento de datos. Mientras que el capítulo 3.4 se centra más en la configuración y diseño de dashboard de control.

3.1 ANÁLISIS Y DISEÑO DE LAS CARACTERÍSTICAS BÁSICAS DEL SISTEMA EMBEBIDO

3.2 Introducción.

La Ingeniería de Requisitos (RE) es ahora una disciplina bien establecida de investigación y práctica en el desarrollo de software y sistemas. La importancia de desarrollar y seguir prácticas eficaces de energías renovables ha sido reconocida desde hace mucho tiempo tanto por los investigadores como por los profesionales. (Kamalrudin, Ahmad, & Ikram, 2017)

En esta parte del documento de investigación tiene como objetivo delimitar las características fundamentales que fueron trazados para el desarrollo del sistema embebido. Tomando en consideración todos los elementos que intervienen con la toma de datos ambientales en el invernadero de rosas del ESPE IASA I siguiendo la metodología Ralph Kimball.

3.2.1 Funciones del sistema embebido.

La figura 11 se puede observar los Casos de Uso que se establecieron en el análisis del sistema embebido, relacionándose de forma siguiente: los datos ambientales son obtenidos por la red de sensores ambientes de la placa Arduino, que recepta: temperatura, humedad, luminosidad, índice ultravioleta y de calidad de aire cada 3 minutos esto se programa en Arduino IDE.

Las placas de Arduino se conectan a la Raspberry pi, mediante la programación en Python se tratan los datos captados y posteriormente se almacena en la base de datos.

Mediante Grafana que es la herramienta de visualización se muestra los datos recolectados en tiempo real.

Entre las funcionalidades del sistema embebido se deberá tener en cuenta los procesos de captación, tratamiento, almacenamiento, transmisión y visualización de la información ambiental del invernadero como también la frecuencia de la toma de datos según los expertos. En la Tabla 5 se detallan las funcionalidades generales del sistema.

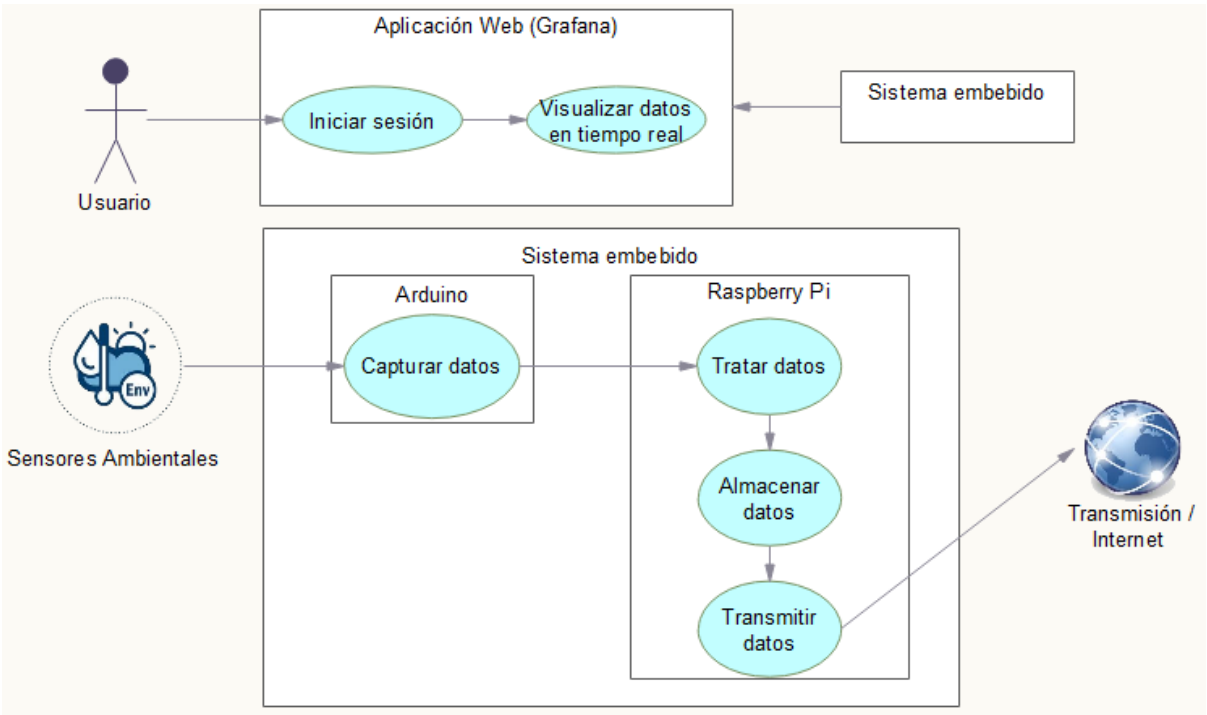


Figura 11 Casos de Uso

Tabla 5

Funcionalidades del Sistema

FUNCIONALIDAD	DESCRIPCIÓN
Captación	En este proceso es el encargado de captar los datos de temperatura, factor UV, calidad del aire, brillo y humedad; apropiadamente configurados y graduados en la placa de Arduino desde Arduino Studio.
Tratamiento	Las placas de Arduino luego de captar los datos ambientales son guardadas en una sola cadena de caracteres las cuales necesita ser tratadas en Python para su posterior Almacenamiento, aquí se agrega la fecha y hora en que fueron tomados los datos.
Almacenamiento	Una vez que los datos son tratados en Python se utiliza las librerías necesarias para guardar todos los datos en el gestor de base de datos MariaDB, el modelamiento de la base de datos debe resguardar la integridad de los datos recibidos.
Transmisión	Después de almacenar los datos, estos son transmitidos en tiempo real mediante un puente de comunicación mediante el Conector Kafka a fuente MySQL que es un Controlador JDBC
Visualización	De todos los datos almacenados y que se encuentran transmitiendo, la herramienta Web Grafana presenta en forma de gráficos lineales en tiempo real, calculando también los valores: máximos, mínimos y promedios de cada dato ambiental

3.2.2 Limitaciones del sistema embebido.

Como todo caso práctico el sistema cumple las funciones específicas, pero también se han identificado ciertas limitaciones en su desarrollo las cuales a continuación se describen:

- Para el funcionamiento del sistema se debe proveer de internet al invernadero, caso contrario el sistema funcionará localmente y no en internet.
- No se realizará la toma de datos de más de dos placas de Arduino.
- No se realizará predicciones de los datos tomados.
- No se podrá consultar información de otros sistemas embebidos.
- No se incluirá ninguna clase de alerta en referencia a los datos tomados, ya que la información obtenida por los sensores se basa en su programación de fábrica.

Las ventajas de usar una herramienta de visualización como aplicación web, que pueda interactuar con el sistema embebido para procesar información del invernadero y permitir la visualización de datos de sensores son las siguientes:

- Extender la medición de datos a más de dos placas de Arduino
- Visualizar las condiciones ambientales del invernadero en tiempo real.
- Compartir la información a usuarios específicos sobre el estado del invernadero.
- Guardar la información para análisis futuros.

3.2.3 Suposiciones y Dependencias.

El sistema embebido deberá tener una fuente de alimentación de energía eléctrica apropiada ya que el sistema monitorea a diario sin excepción, sin esta particularidad no será posible realizar ninguna de las funcionalidades del sistema.

El equipo donde se accede a la Web deberá estar conectado a internet para la visualización en tiempo real de los datos ambientales.

3.2.4 Requisitos Específicos

En la Tabla 6 se detalla los requisitos que permitirán planear, delinear, desarrollar y experimentar el sistema embebido y el software que ejecutará cada dispositivo electrónico.

Tabla 6

Requisitos Específicos

REQUISITO ESPECÍFICO	NOMBRE DEL REQUISITO	CARACTERÍSTICAS
RE01	**Capturar Datos**	En la placa de Arduino se podrá capturar las lecturas de los datos ambientales por medio de los sensores.
RE02	**Tratar Datos**	El sistema embebido procesará los datos generados por los sensores.
RE03	**Almacenar Datos**	Consiente en guardar en la base de datos la información tratada en Python.
RE04	**Transmitir Datos**	El Sistema embebido mediante conectores transmitirá la información hacia la herramienta web.
RE05	**Iniciar sesión**	El usuario debe iniciar sesión identificándose para acceder a Grafana.
RE06	**Visualizar Datos**	La herramienta Web mostrará en dashboard el monitoreo de los datos del invernadero en tiempo real.

3.2.5 Requisitos No Funcionales

En la Tabla 7 se detalla los requisitos no funcionales que deberá tener el sistema embebido.

Tabla 7

Requisitos No Funcionales

REQUISITO NO FUNCIONAL	NOMBRE DEL REQUISITO	CARACTERÍSTICAS
RNF01	**Rendimiento**	La herramienta Web y la transmisión de datos, serán configuradas de forma correctas para garantizar un desempeño adecuado. Es transcendental reiterar que al usar internet para la visualización de los datos depende mucho de la calidad de internet del proveedor de este servicio.
RNF02	**Seguridad**	El usuario final tendrá acceso a la información de forma segura ya que la herramienta web será indispensable registrase con un usuario y crear una clave de acceso.
RNF03	**Fiabilidad**	Todas las funciones detalladas en el sistema embebido deberán ejecutarse de forma fiable, en especial el almacenamiento y visualización de la información, en la base de datos garantizar la relación entre los datos y la integridad de los mismos.
RNF04	**Disponibilidad**	El Sistema embebido será delineada para que esté disponible siempre a los usuarios. Esto también depende de la Herramienta Web que se está utilizando y del proveedor de Internet.

3.3 CONSTRUCCIÓN DEL SISTEMA EMBEBIDO.

En base al análisis y diseño de las características que debe tener el sistema embebido reconocidos en el capítulo 3.1, a continuación, se bosquejará y se diseñará su arquitectura, los diagramas de base de datos y casos de uso que permitirá la construcción del prototipo.

3.3.1 Arquitectura del sistema embebido.

La implementación comienza con la conexión correcta de los componentes de hardware, tal y como se describe en el diagrama de arquitectura del sistema de la Figura 12 que se muestra la visión general de la arquitectura del sistema embebido IoT propuesto. De acuerdo con la arquitectura, hay dos placas Arduino conectadas a un concentrador central Raspberry Pi. Estas placas de Arduino utilizan cable USB para interactuar con la Raspberry Pi, produciendo así un grupo de dos nodos de sistema.

La Raspberry Pi está conectado a un Router para proporcionar Internet y comunicarse a través de la red. También hay cinco sensores ambientales conectados a cada Arduino que están jugando un papel vital para hacer nodo sensor de IoT.

La Tabla 8 ilustra los componentes de hardware requeridos junto con la cantidad para ensamblar un nodo sensor requerido.

En este sistema embebido, Raspberry Pi 3 Model B ha sido elegido para hacer el concentrador de los nodos porque este era el último modelo cuando se inició el proyecto. La Raspberry Pi 3 es compatible con los modelos más antiguos, ya que tiene un diseño idéntico. Esto significa que puede ser fácilmente intercambiable con la versión 2, sin ningún impacto en el rendimiento.

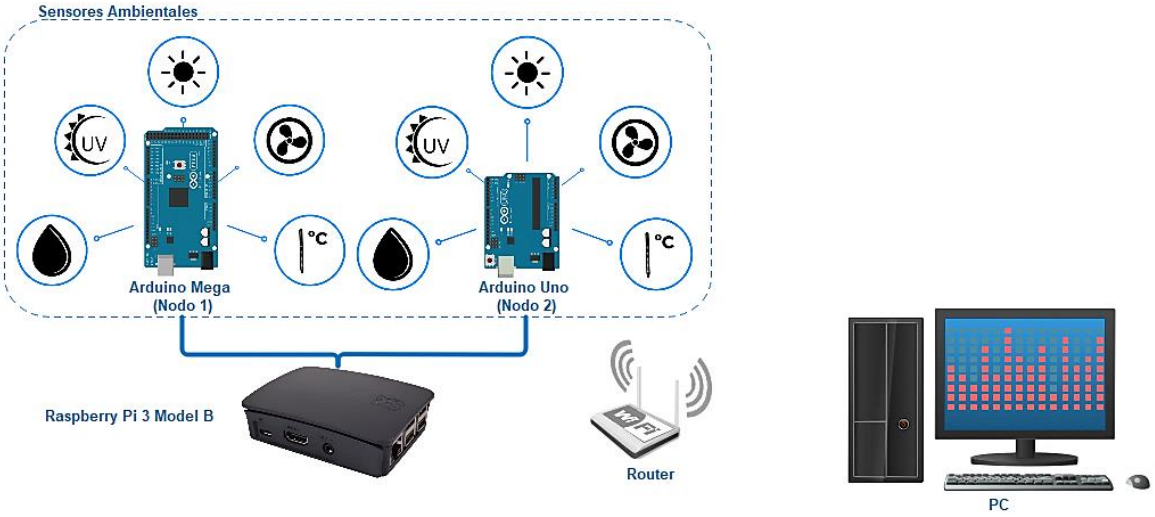


Figura 12 Arquitectura del sistema embebido

Tabla 8

Componentes de Hardware

Componentes de hardware	Cantidad
Ardruino	1
Sensor de Temperatura y Humedad (DHT22)	1
Sensor de Luminosidad (MH Series Flying Fish)	1
Sensor de índice Ultravioleta	1
Sensor de Calidad de Aire (MQ 135)	1
Antenas de Transmisión y Recepción de Datos (NRF22L01)	1
Memoria Micro SD 64 Gb Clase10	1

3.3.2 Diseño de la base de datos.

MariaDB es una base de datos central que fue creada como reemplazo de MySQL debido a su facilidad de uso. Las tablas de base de datos contienen valores de medición de los factores ambientales y para mejorar el rendimiento se ha diseñado estas tablas ya que poseen un volumen muy grande de datos esta forma de solucionar el diseño es una forma efectiva de cumplir con la rapidez de proceso.

Del mismo modo, los resultados de las mediciones del invernadero tienen características particulares que no se dejan de lado y se las considera como por ejemplo si los datos son enteros o decimales. Hay nueve diferentes atributos de la tabla en la base de datos que se enlista en la tabla 9 y los nombres de tabla en el sistema son las siguientes y se muestran en la Fig. 13.

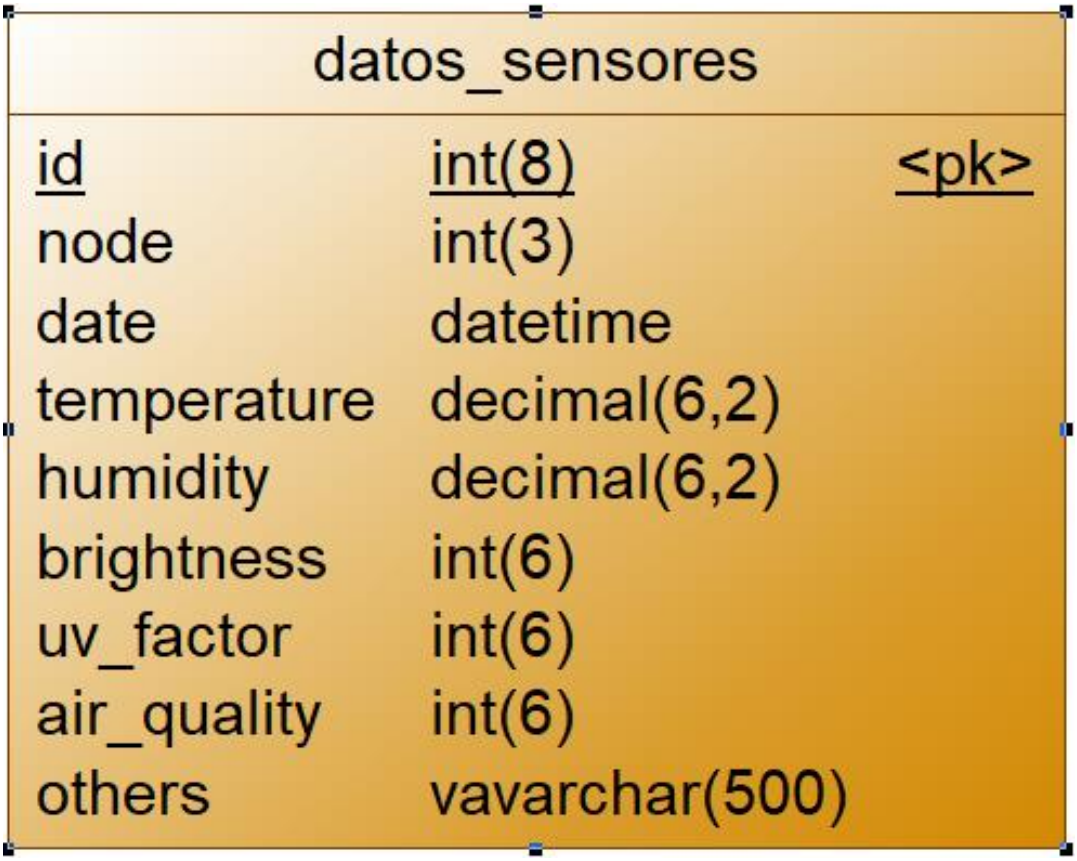


Figura 13 Diseño de la base de datos

Tabla 9

Atributos de la base *de datos*

Nombre	Detalle
Id	Es el identificador o clave primaria que toma el sistema embebido por cada registro en la base de datos.
Node	Muestra el nodo al que pertenece cada registro.
Date	La fecha y hora en que fue tomada cada registro.
Temperatura	El dato de la temperatura con dos cifras decimales.
brightness	El dato del brillo ambiental con dos cifras decimales.
uv_factor	El dato del factor uv como entero.
air_quality	El dato de la calidad de aire como entero.
others	Un campo adicional donde no se almacena nada.

3.3.3 Diagramas de casos de uso.

El diagrama de casos de uso se utiliza para obtener el comportamiento de los actores involucrados en el sistema (Eichelberger, 2008). El diagrama de casos de uso se visualiza como una alta funcionalidad del sistema basada en el punto de vista del usuario.

Se usará la herramienta PowerDesigner que es un software para hacer el desarrollo de programas informáticos basados en casos de uso específicos cuyo caso de uso está en el centro del desarrollo.

El caso de uso es el principal diagrama UML en la representación del sistema que puede utilizarse como punto de referencia para otros diagramas como el diagrama de secuencia y el diagrama de actividad, para así estar al tanto de cómo se va desarrollando el sistema que al final cumplirá con todos los requisitos.

3.3.3.1 Descripción general de los actores.

A continuación, se enlisto los actores que interactuaran con el sistema embebido.

- Usuario: puede ser administrador de dirección, también es la persona quien podrá visualizar los datos en tiempo real.

- Sensores ambientales: sensores conectados a la placa de Arduino.
- Raspberry Pi: minicomputador donde se procesa toda la información captada.
- Base de Datos: repositorio donde se guardan los datos de la aplicación.

En la Figura 13 se ha descrito los Casos de Uso que fueron identificados en el análisis y diseño de las características básicas del sistema embebido, relacionado de la siguiente representación: los sensores ambientales conectados a la placa de Arduino obtienen los datos de brillo, calidad del aire, factor uv, humedad y temperatura del invernadero. Cada tres minutos se envía esta información a la Raspberry Pi y se almacena en la base de datos MariaDB, se utiliza el controlador JDBC que conecta Kafka a MySQL con lo que se producen mensajes por cada registró insertado en la base de datos esto concede al usuario final tener los datos en tiempo real.

Mediante la herramienta Web Grafana se adquiere la visualización de los datos recogidos, el cual es muy relevante ya que nos permite ver el estado del invernadero y así tomar decisiones de ser necesario. Para mejor comprensión al caso de uso general se ha fragmentado y ha sido mostrado de manera más detallada:

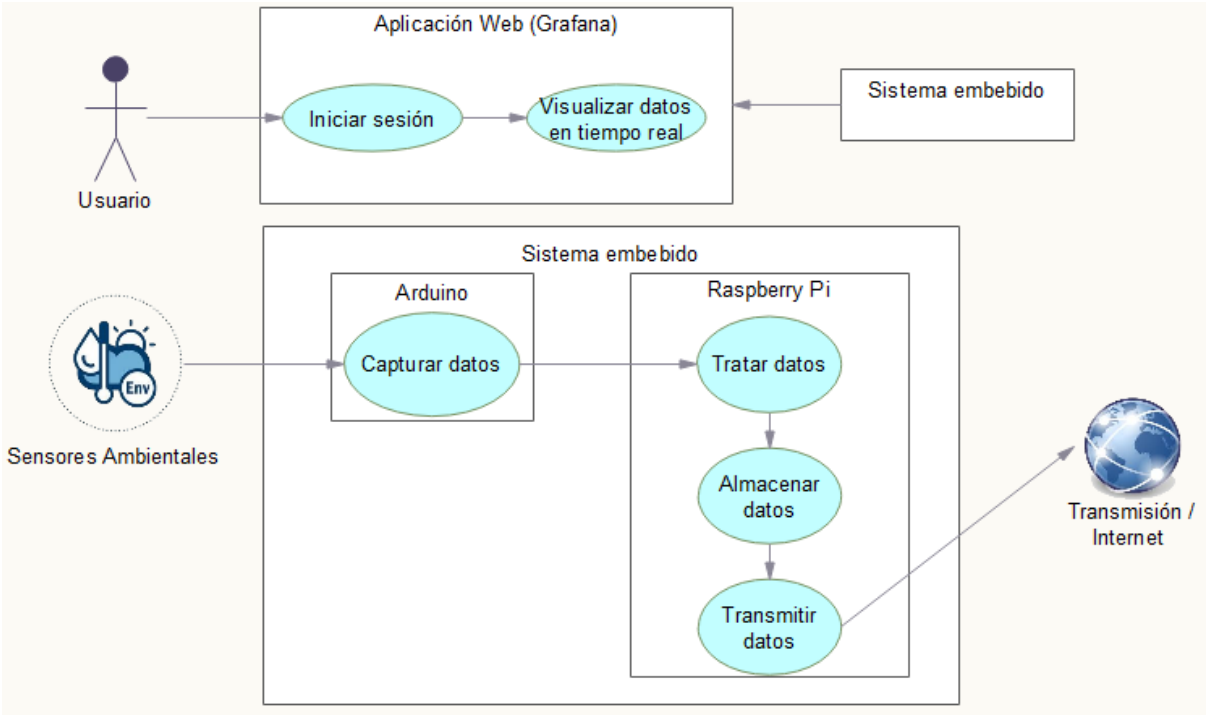


Figura 14 Casos de Uso

3.3.4 Caso de uso: Iniciar sesión.

En la Tabla 10 se detalla el caso de uso de Iniciar sesión en el sitio Web con su correspondiente diagrama de uso que se puede ver en la Figura 15.

Tabla 10

Caso de Uso I: Iniciar Sesión

Caso de uso	**Iniciar Sesión**
Actor	Usuario
Descripción	Con el nombre de usuario y contraseña de acceso. El usuario se autenticará en el sistema, la herramienta web identifica al usuario.
Entradas	Nombre de usuario y contraseña
Salidas	Ingreso al sistema
Proceso	1- Ingreso del nombre de usuario 2- Ingreso de la contraseña de usuario 3- Valida los dos datos ingresados 4- Inicio de sesión del usuario 5- Muestra la página inicial de la aplicación.
Precondiciones	• Tener un nombre de usuario registrado. • Tener una contraseña de usuario
Pos condiciones	NA
Efectos colaterales	NA
Prioridad	Alta
Excepciones	

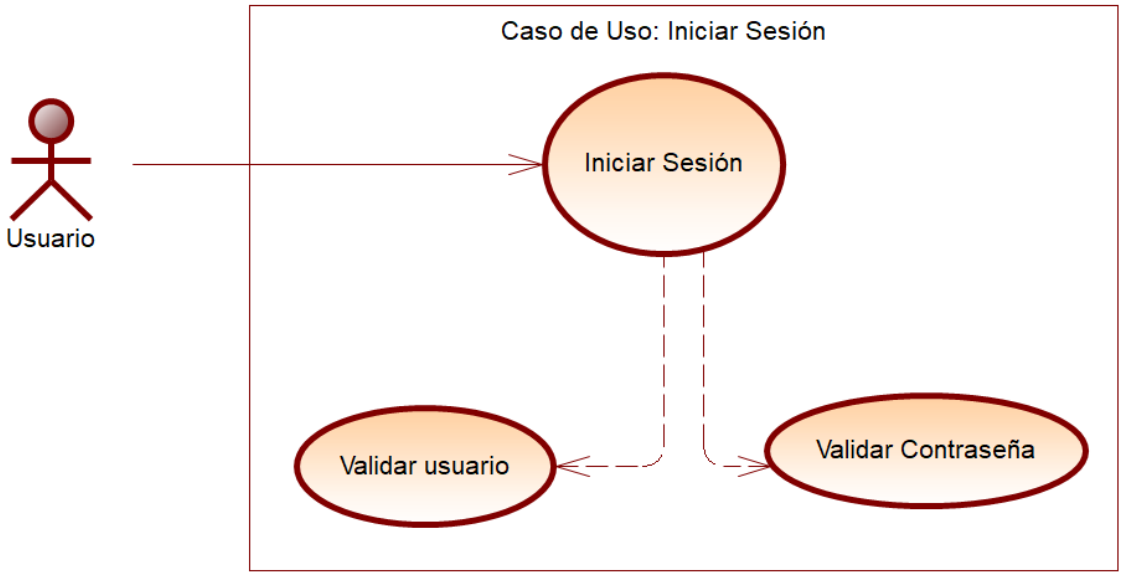


Figura 15 Caso de Uso I: Iniciar sesión

3.3.5 Caso de uso II: Visualizar datos en tiempo real.

En la Tabla 11 se detalla el caso de uso de Visualizar datos en tiempo real con su correspondiente diagrama de uso que se puede ver en la Figura 16.

Tabla 11

Caso de Uso II: Visualizar datos en tiempo real

Caso de uso	**Visualizar datos en tiempo real**
Actor	Usuario
Descripción	El usuario después de ingresar al sistema puede visualizar los datos en tiempo real
Entradas	Registros de la base de datos
Salidas	Datos en tiempo real
Proceso	

CONTINÚA

	1. Selecciona el dashboard que desea visualizar: 2. Al ver los datos podemos ver los nodos respectivos de cada dato. 3. Podremos buscar datos por intervalo de tiempo. 4. Podemos compartir los datos. 5. Podremos exportar los datos en formato json.
Precondiciones	Haber ingresado al sistema
Pos condiciones	NA
Efectos colaterales	NA
Prioridad	Media
Excepciones	

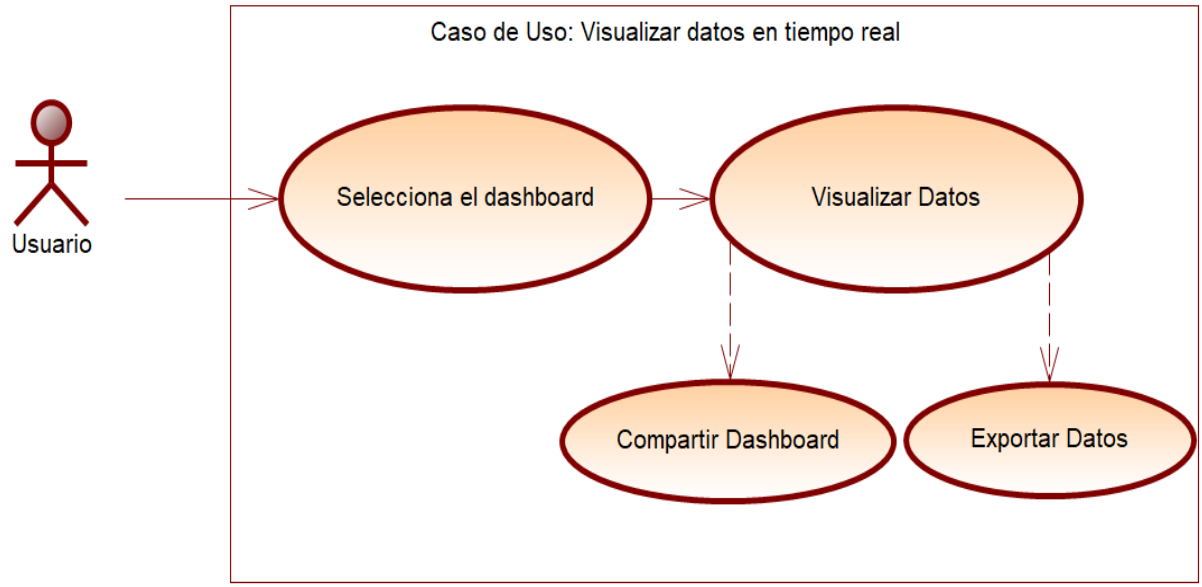


Figura 16 Caso de Uso II: Visualizar datos en tiempo real

3.3.6 Caso de uso III: Capturar datos.

En la Tabla 12 se encuentra descrito el caso de uso de Capturar datos y con su respetivo diagrama que se encuentra en la Figura 17

Tabla 12

Caso de Uso III: Capturar datos

Caso de uso	**Capturar datos**
Actor	Sensores Ambientales
Descripción	En este proceso es el encargado de captar los datos de temperatura, factor UV, calidad del aire, brillo y humedad; apropiadamente configurados y graduados en la placa de Arduino desde Arduino Studio.
Entradas	Datos ambientales
Salidas	Capturar datos ambientales
Proceso	1. Programar en la placa de Arduino la captura de datos ambientales.
Precondiciones	Conectar todos los sensores ambientales a la placa de Arduino
Pos condiciones	NA
Efectos colaterales	NA
Prioridad	Media
Excepciones	

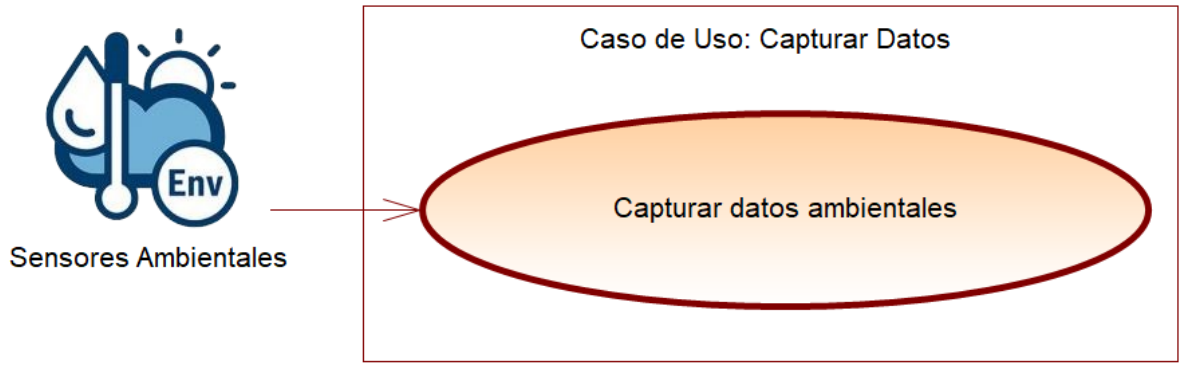


Figura 17 Caso de Uso III: Capturar Datos

3.3.7 Caso de uso VI: Tratar datos.

En la Tabla 13 se encuentra descrito el caso de uso de Tratar datos y con su respetivo diagrama que se encuentra en la Figura 18.

Tabla 13

Caso de Uso VI: Tratar datos

Caso de uso	**Tratar datos**
Actor	Raspberry Pi
Descripción	El sistema embebido procesará los datos generados por los sensores.
Entradas	Datos ambientales
Salidas	Datos ambientales tratados
Proceso	1. Las placas de Arduino luego de captar los datos ambientales son guardadas en una sola cadena de caracteres. 2. En Python la cadena de caracteres necesita ser tratadas en para su posterior Almacenamiento. 3. Se agrega la fecha y hora en que fueron tomados los datos.
Precondiciones	Captura de datos
Pos condiciones	NA
Efectos colaterales	NA
Prioridad	Media
Excepciones	

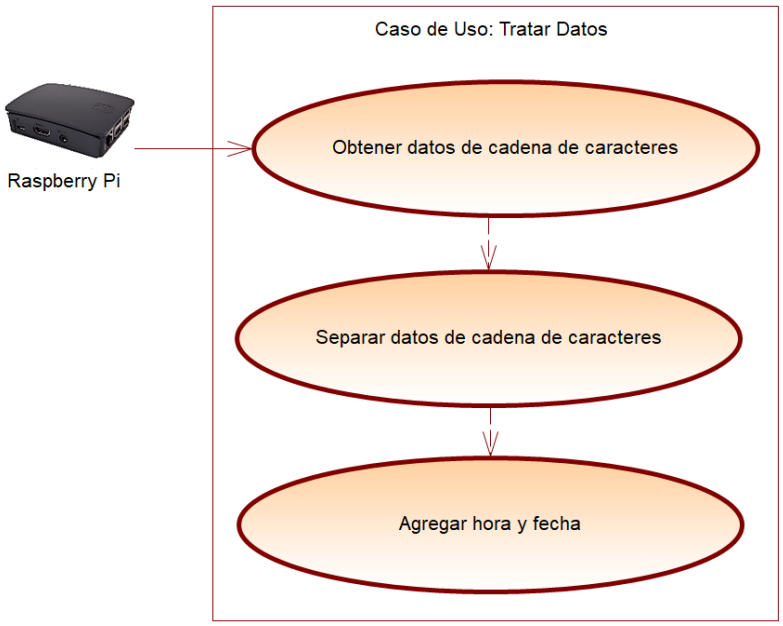


Figura 18 Caso de Uso VI: Tratar Datos

3.3.8 Caso de uso V: Almacenar datos.

En la Tabla 14 se encuentra descrito el caso de uso de Almacenar datos y con su respetivo diagrama que se encuentra en la Figura 19.

Tabla 14

Caso de Uso V: Almacenar datos

Caso de uso	**Almacenar datos**
Actor	Raspberry Pi
Descripción	Consiente en guardar en la base de datos la información tratada en Python.
Entradas	Datos ambientales
Salidas	Datos ambientales tratados
Proceso	1. Una vez que los datos son tratados en Python se utiliza las librerías necesarias para guardar todos los datos en el gestor de base de datos MariaDB, el modelamiento de la base de datos debe

CONTINÚA

	resguardar la integridad de los datos recibidos
Precondiciones	Tratar datos
Pos condiciones	NA
Efectos colaterales	NA
Prioridad	Media
Excepciones	

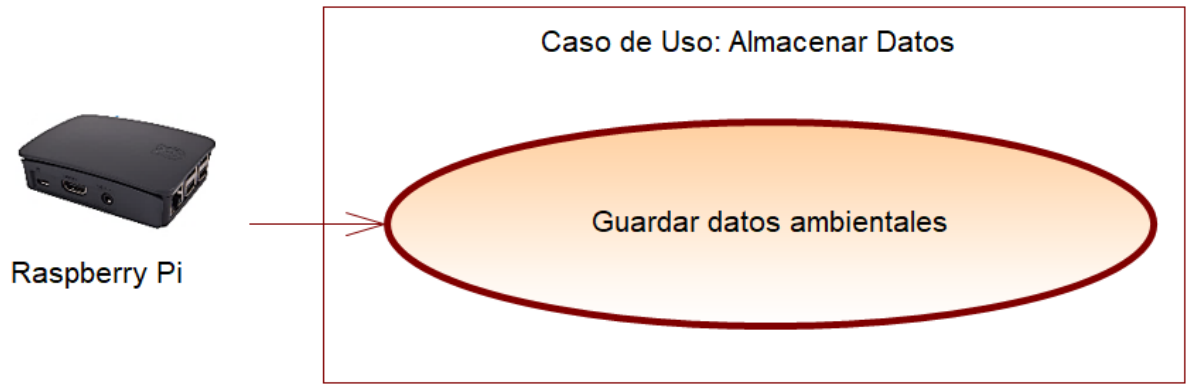


Figura 19 Caso de Uso V: Almacenar datos

3.3.9 Caso de uso VI: Transmitir datos.

En la Tabla 15 se encuentra descrito el caso de uso de Almacenar datos y con su respetivo diagrama que se encuentra en la Figura 20.

Tabla 15

Caso de Uso VI: Transmitir datos

Caso de uso	**Transmitir datos**
Actor	Raspberry Pi
Descripción	El Sistema embebido mediante conectores transmitirá la información hacia la herramienta web.
Entradas	Datos ambientales almacenados en MariaDB
Salidas	Datos ambientales en mensajes Kafka

CONTINÚA

Proceso	1. Se crea un productor en apache Kafka donde se transmiten los mensajes. 2. Después de almacenar los datos, estos son transmitidos en tiempo real mediante un puente de comunicación mediante el Conector Kafka a fuente MySQL que es un Controlador JDBC 3. Se crea un consumidor de apache Kafka donde se reciben los mensajes.
Precondiciones	Almacenar datos
Pos condiciones	NA
Efectos colaterales	NA
Prioridad	Media
Excepciones	

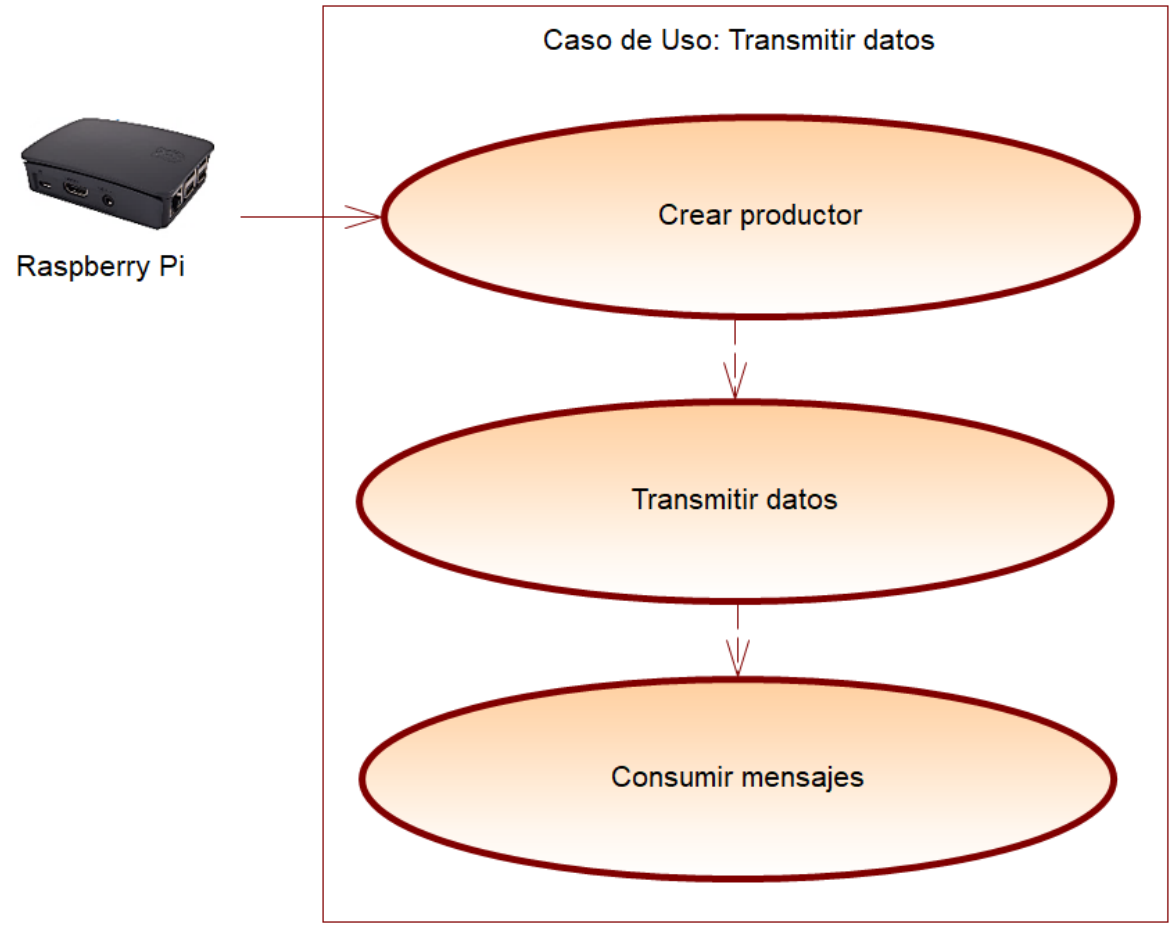


Figura 20 Caso de Uso VI: Transmitir datos

3.4 CONFIGURACIÓN Y CONEXIÓN DE MÓDULOS.

3.4.1 Introducción.

Después de conectar todos los componentes de hardware, es el momento de instalar el software en cada placa de Arduino y en la Raspberry Pi. La versión de noviembre del 2017 del sistema operativo Raspbian ha sido seleccionada ya que es una distribución Linux basada en Debian para los procesadores ARM. En el inicio del proyecto dicha versión fue la última ahora existe la última versión de junio del 2018.

Las instrucciones dadas en la página web se utilizan para instalar y configurar Raspbian en cada Pi de Frambuesa. Una vez realizada la instalación, se instala todo el software necesario para configurar la carga ETL de los datos del invernadero.

El siguiente paso es conectar la Raspberry Pi con un servidor de ordenador para que Internet esté disponible la Raspberry Pi obtenga una dirección IP de forma dinámica a través del protocolo DHCP. Como existe una sola Raspberry Pi con los dos nodos placas de Arduino no es práctico, ni muy útil dar un nombre de host único a la Raspberry Pi.

Aunque hay dos nodos de trabajo en nuestra implementación, se pueden añadir más trabajadores al para proporcionar una mayor escalabilidad. Del mismo modo, también se pueden integrar más nodos maestros Raspberry Pi. El siguiente paso es configurar los componentes de maestro y operario cada placa Arduino correspondiente, que se describe en la siguiente sección.

En este capítulo se detallará la configuración de los componentes primordiales del sistema embebido como son Zookeeper, Apache Kafka con sus adecuados conectores de entrada y de salida Kafka Connect.

Apache Kafka no es indispensable que este corriendo o ejecutándose en un clúster, pero toca tener en cuenta que existe un riesgo grande ya que si falla este único nodo trabajador el sistema dejaría de funcionar.

3.4.2 Instalación y configuración del Sistema Operativo.

El sistema Operativo que tiene una Raspberry es Rasbian una derivación de Debian basados en Linux. Debemos descargar la imagen del repositorio oficial de Raspberry Pi, Las características se detallan en la Tabla 16 y en la Figura 21.

Tabla 16

Características del Sistema Operativo

Característica	DESCRIPCIÓN
Nombre	RASPBIAN STRETCH WITH DESKTOP
Versión	Noviembre 2017
Fecha de lanzamiento	2017-11-29
Versión del Kernel	4.9
Notas de la versión	• Complemento de monitor de batería agregado para la barra de tareas: funciona en imágenes x86 o Pi-Top de primera generación • Se agregó el modo de reducción al administrador de archivos PCManFM para reducir la complejidad • Se agregó la capacidad de cambiar el nombre de los archivos en PCManFM al hacer clic en el nombre cuando se selecciona • Corrección de errores en el módulo Bluetooth ALSA para reducir el truncamiento de audio al final de la reproducción • Varios pequeños retoques, correcciones de errores y modificaciones de tema • Nuevo kernel y firmware

Figura 21 Características del Sistema Operativo

Fuente: (Raspberry Pi Foundation, s.f.)

3.4.2.1 Quemar la ISO de la Raspberry en la microSD

Es necesario escribir la imagen de Raspbian en una tarjeta MicroSD para que el sistema operativo inicialice desde la Raspberry Pi. En esta ocasión se ha manejado el software Etcher 1.3.1 que es una herramienta potente para flashear imágenes del sistema operativo construido con tecnologías web para asegurar que el flasheo de una tarjeta, garantiza que cada byte de datos se haya escrito correctamente. Como se muestra en la Figura 22.

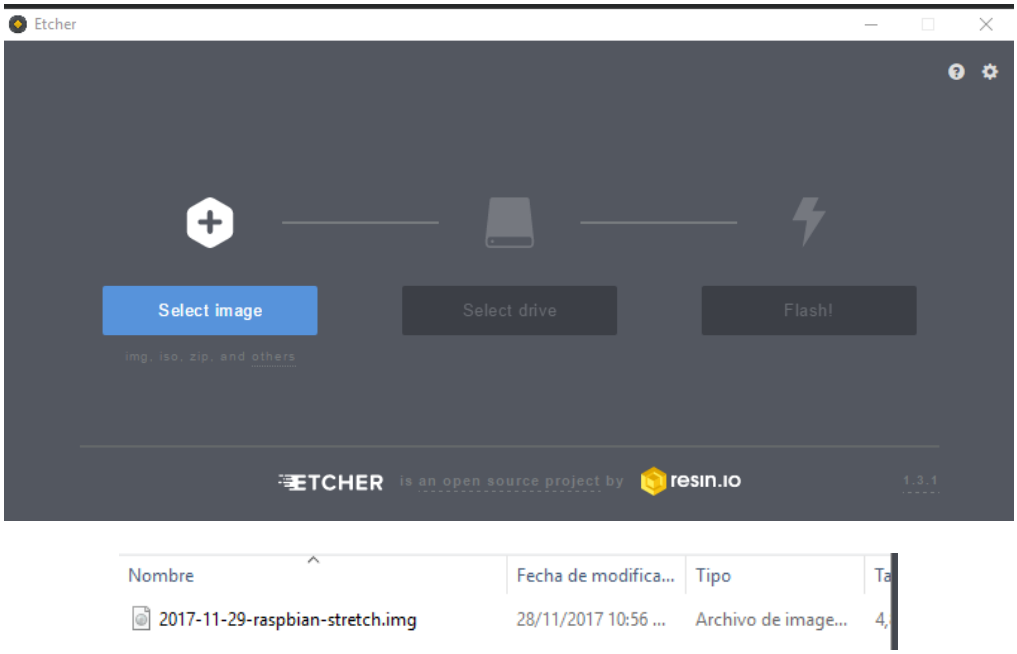


Figura 22 Etcher 1.3.1

Previo a la instalación de cualquier elemento necesario para realizar la arquitectura adecuada necesitamos configurar nuestro equipo.

3.4.2.2 Optimizar Memoria RAM

Liberamos memoria RAM de la tarjeta gráfica que viene por defecto 64MB, cambiar a 16MB ya que Apache Kafka y Zookeeper requerirán cualquier espacio adicional más de memoria RAM. Como se muestra en la Figura 23.

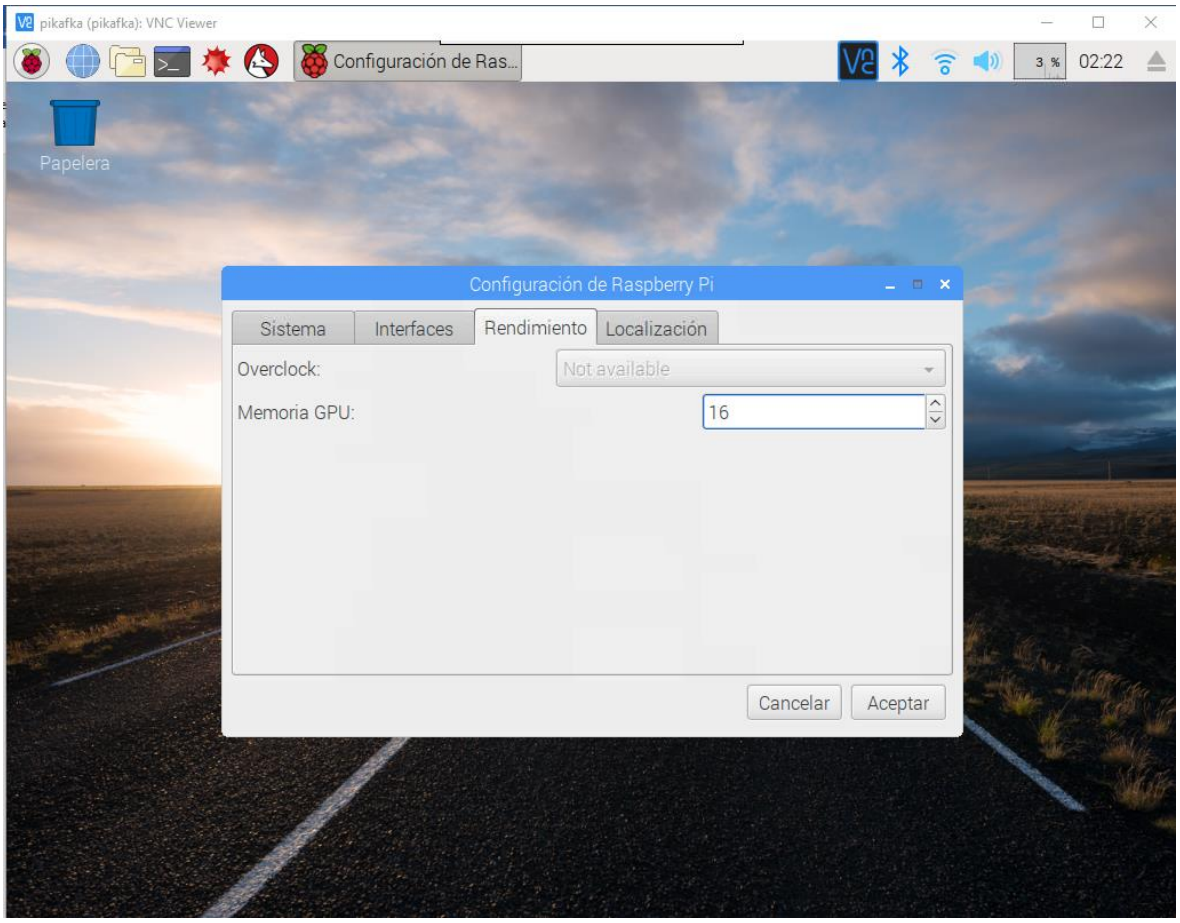


Figura 23 Optimizar Memoria RAM

3.4.2.3 Ampliación de tamaño del archivo SWAP

El tamaño predeterminado del archivo de SWAP es de 100MB y para el buen funcionamiento del sistema embebido se desea extender la cantidad de RAM en un equivalente de 4 veces el valor por defecto del sistema del sistema (l GB RAM) para prevenir dificultades

cuando la memoria esté en uso. Para cambiar la configuración, se debe cambiar los valores del archivo dphys- swapfile y poner las siguientes propiedades que se muestran en la Figura 24.

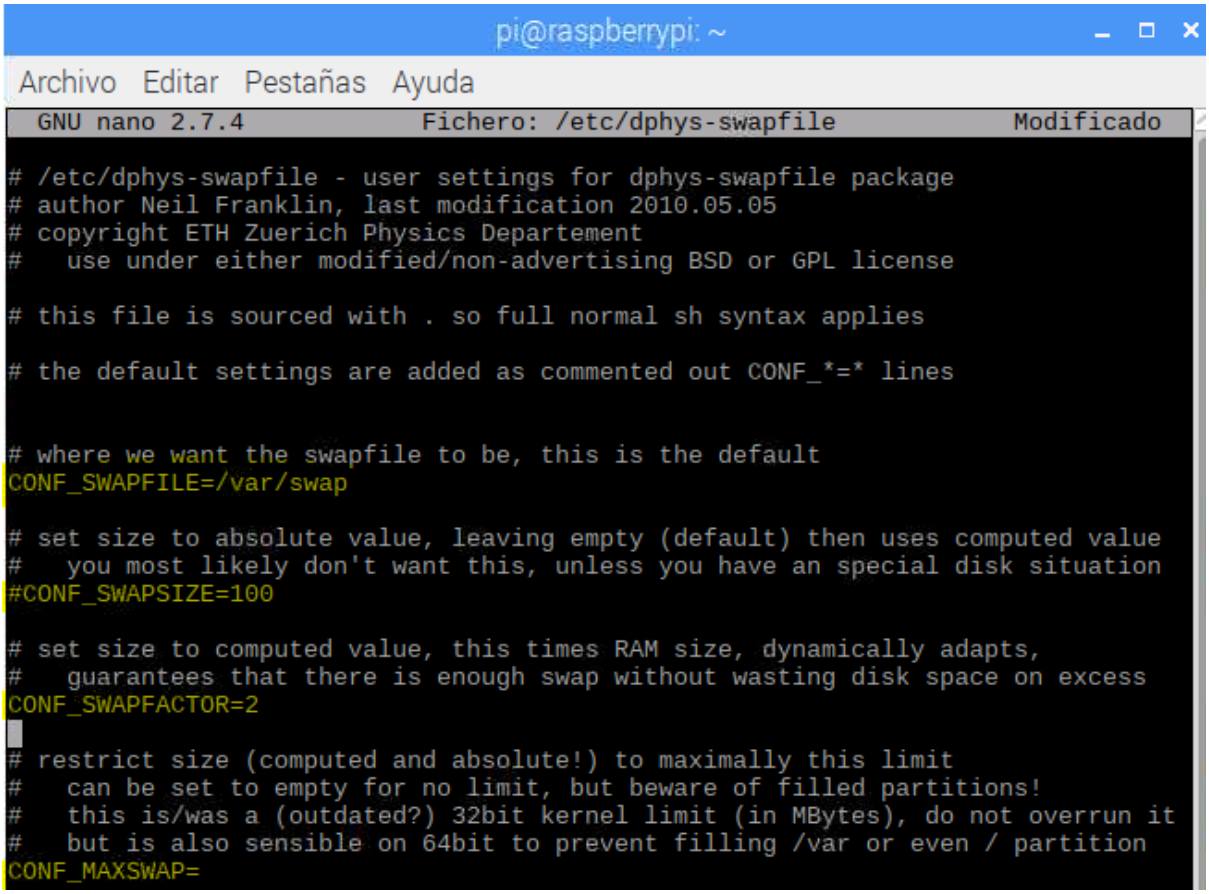


Figura 24 Ampliación de tamaño del archivo SWAP

3.4.3 Configurar placas de Arduino.

Para programar las placas de Arduino se utilizó Arduino (IDE) que es una herramienta software de código abierto, que facilita la escritura del código y grabarlo en la placa. Funciona en diferentes sistemas operativos como Mac OS X, Linux, Windows e incluso se puede instalar en la Raspberry Pi. Este software está escrito en Java y apoyado en Processing y otros programas de código abierto.

En Arduino (IDE) se escoge el tipo de placa de Arduino en la que se va a programar como se observa en la Figura 25, dependiendo el tipo de sensor y la placa de Arduino se necesitará librerías específicas para la normal ejecución del programa para el sistema embebido

es necesario importar la librería DHT.h como se muestra en la Figura 26, en esta programación se establece la frecuencia de la toma de datos cuya unidad de tiempo son segundos.

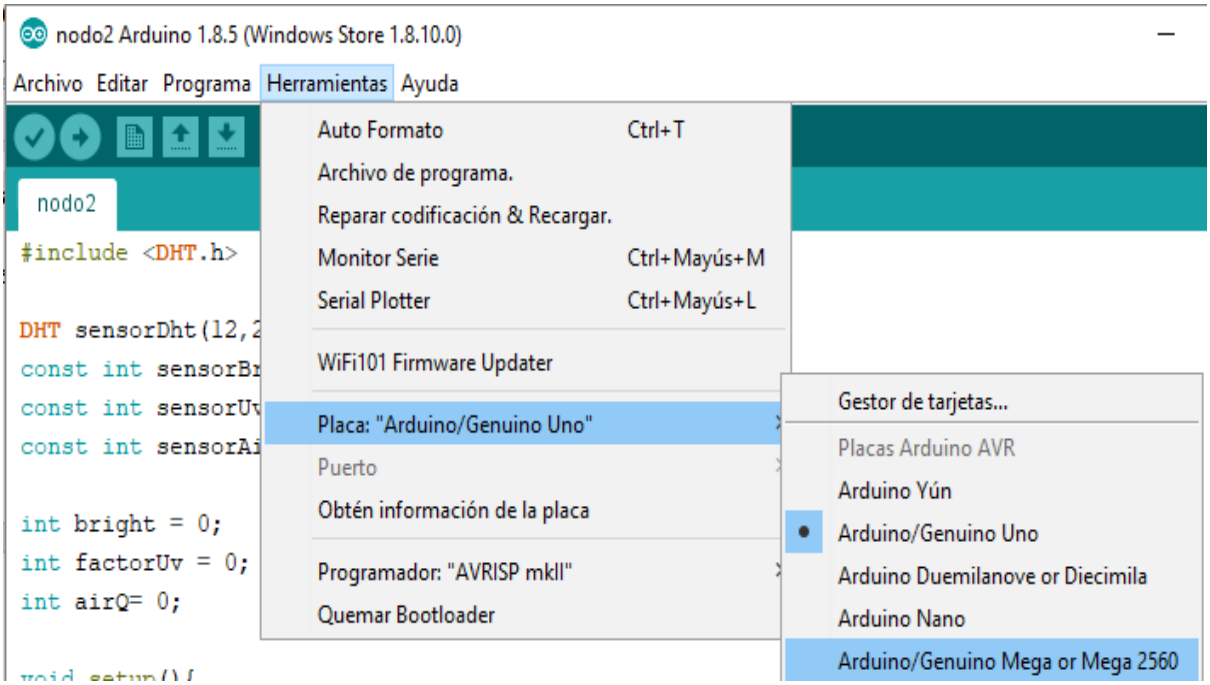


Figura 25 Elección de la placa de Arduino

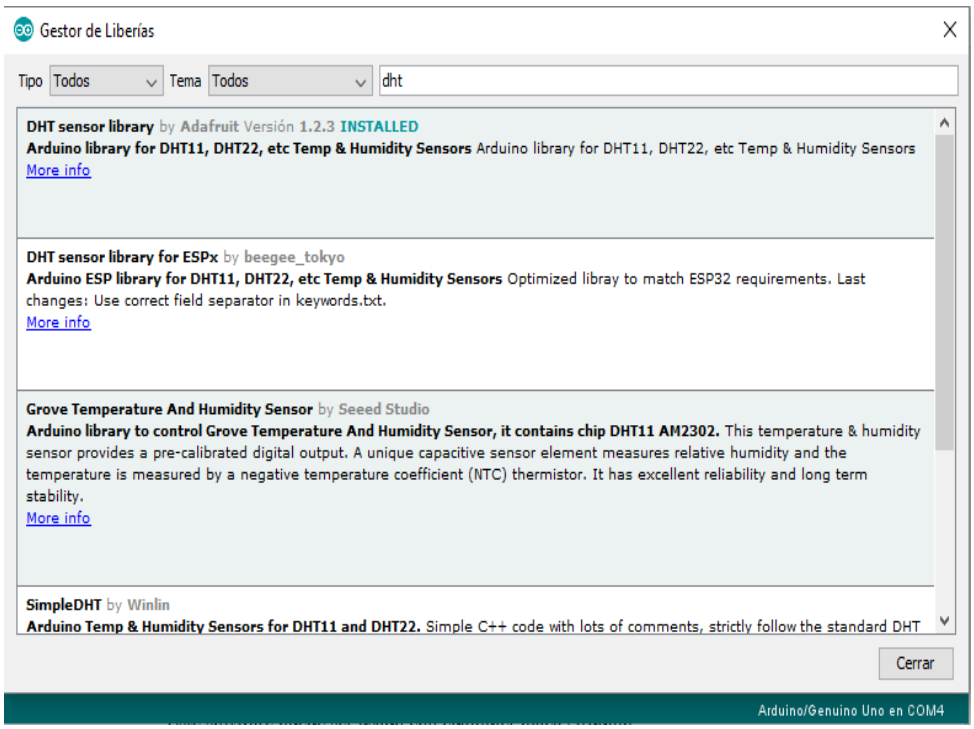


Figura 26 Importar librería DHT

3.4.4 Configurar phpMyAdmin.

Para almacenar los datos obtenidos por las placas de Arduino se programará en Python para guardar los datos de los sensores, utilizamos la herramienta phpMyAdmin para una mejor administración del proceso de guardar los datos, en la Figura 27 podemos observar la interfaz gráfica de la herramienta ya que es fácil administrar desde la Raspberry Pi

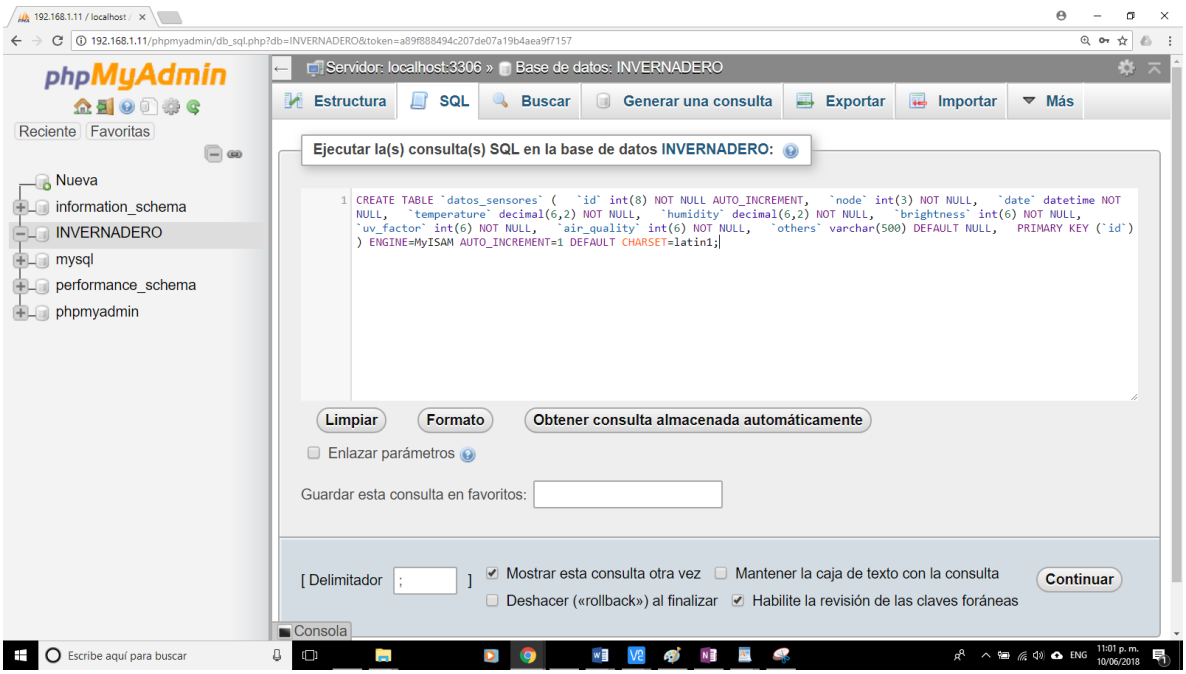


Figura 27 Administración de la base de datos en phpMyAdmin.

3.4.5 Configurar e Instalar Apache Kafka.

Como requisito indispensable la Raspberry se debe instalar java para levantar el sistema de mensajería en tiempo real, una vez instalado java comprobamos la versión de java como se muestra en la Figura 28.

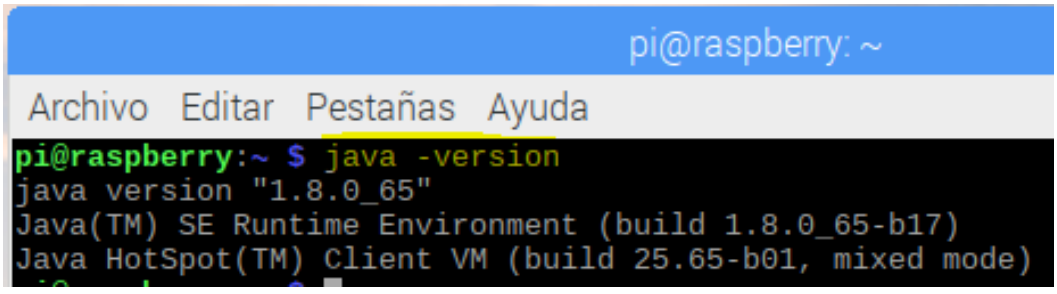


Figura 28 Versión de java.

3.4.5.1 Instalar Apache Kafka

La plataforma Confluent ha creado el empaquetado de Apache Kafka y dispone de un servidor de registro de esquemas que proporciona endpoints REST para el registro y recuperación de esquemas Avro (Confluent, 2017) . El registro de esquema Confluent que es la empresa rada por los autores originales de Apache Kafka, se ejecuta como un servicio y permite muchas operaciones CRUD (crear, leer, actualizar y eliminar) del esquema. También hace cumplir las siguientes reglas de compatibilidad de esquemas cuando se registra un nuevo esquema. Por defecto, proporciona compatibilidad con versiones anteriores, pero puede configurarse/alterarse para proporcionar cualquiera de ellas, a continuación, se enlista los pasos necesarios para instalar Kafka por medio de la plataforma Confluent.

- Para firmar las respectivas herramientas avanzadas de empaquetado (APT), en el LXTerminal de la Raspberry Pi se debe instalar Confluent con su respectiva clave publica ejecutando el siguiente comando que se muestra en la Figura 29.

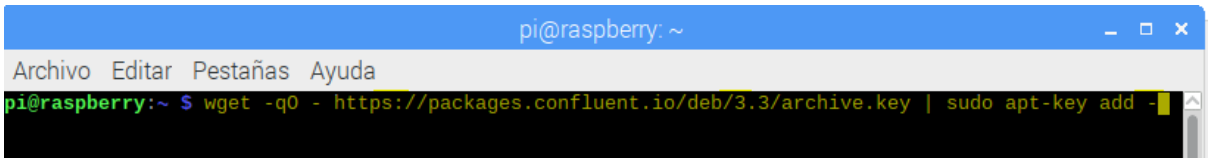


Figura 29 Instalación de la clave pública de Confluent

- En el archivo /etc/apt/sources.list añadir las líneas de comandos para agregar el repositorio de Confluent como se muestra en la Figura 30.

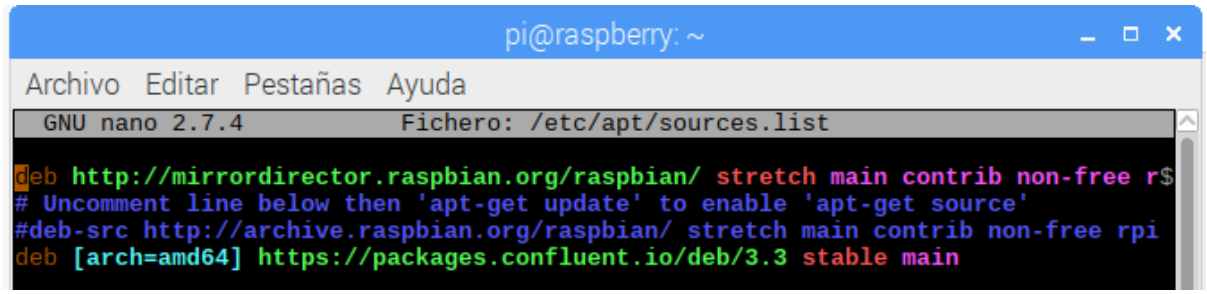


Figura 30 Agregar repositorio Confluent

- Para validar la correcta instalación de Confluent inicializamos todos los servicios disponibles con el comando que se muestra en el Figura 31.

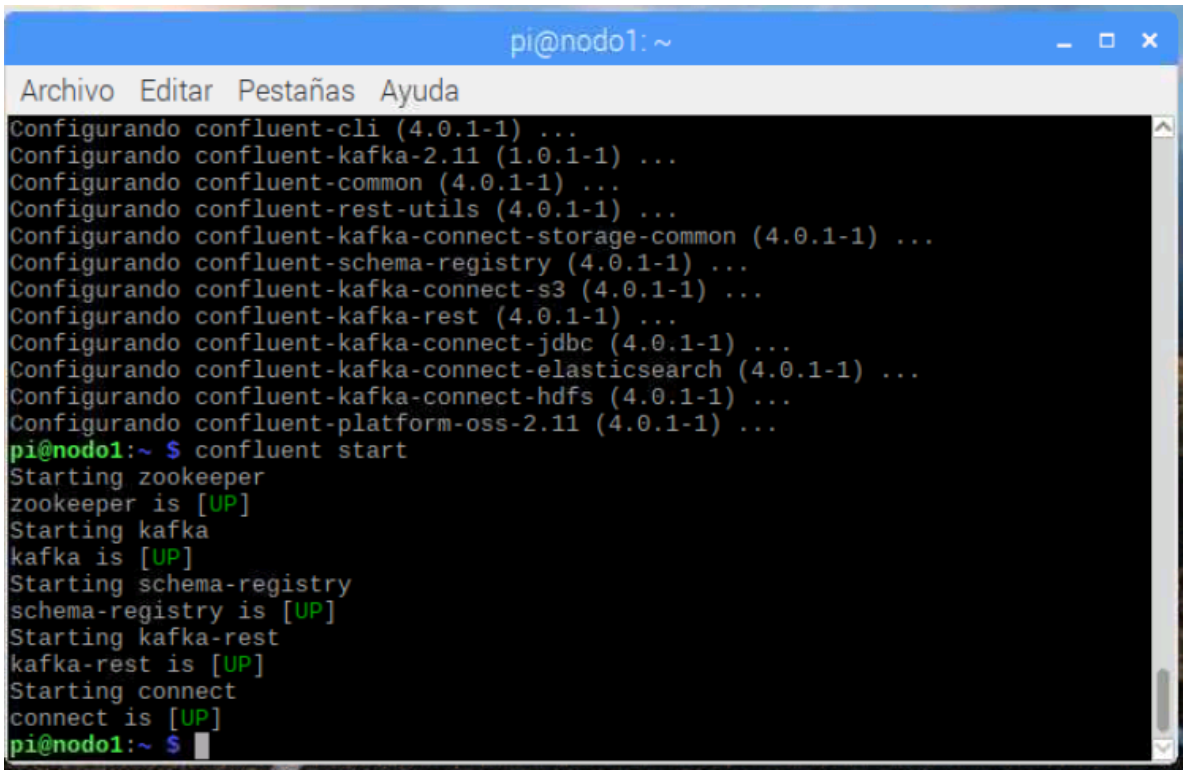


Figura 31 Inicializar servicios Confluent

3.4.6 Configurar conector Kafka a MySQL.

El conector Kafka a MySQL permite la integración de esta base de datos relacional al Kafka Clúster y de forma automática los cambios como inserciones de registros son monitoreados y esto ayuda a introducir esos cambios en el Clúster de Kafka, lo que significa que tenemos un registro de cambios, que un Tema de Kafka ha sufrido.

Previo a la configuración se requiere del conector MySQL para java para poder la base mencionada MySQL conectarse. Descargue el conector MySQL para java, mysql-connector-java-5.1.42-bin.jar, y se copia a la carpeta: /usr/share/java/kafka-connect-jdbc que corresponde a los conectores de Kafka como se muestra en el Figura 32.

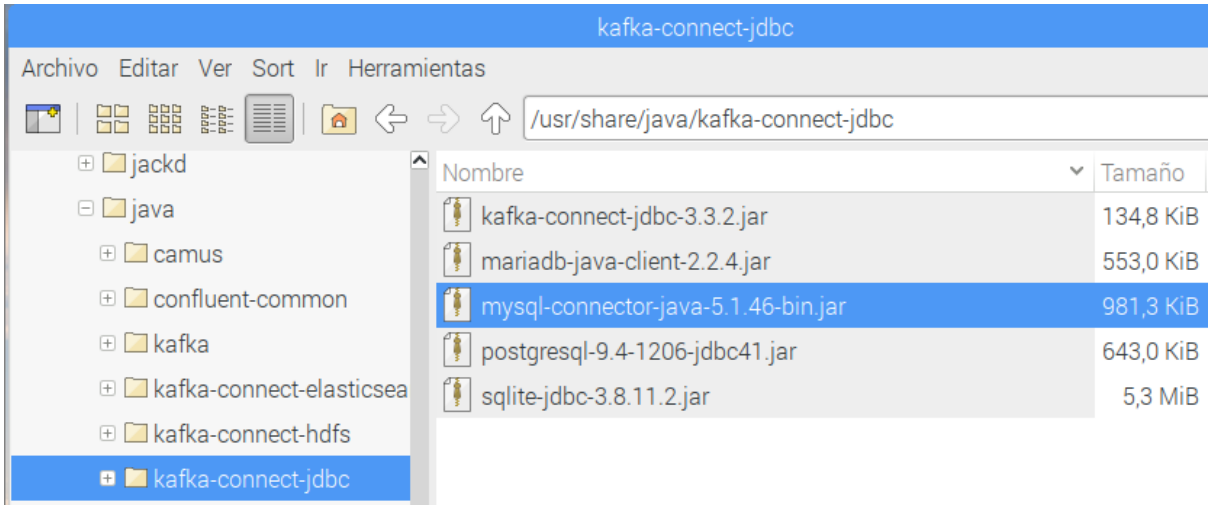


Figura 32 Conector MySQL

Para configurar las propiedades del origen de los datos del conector Kafka se tiene que crear el siguiente archivo: /etc/kafka-connect-jdbc/source-quickstart-mysql.properties donde se tiene que agregar los valores siguientes de la configuración:

- name: aquí se asigna el nombre para el conector que se va a configurar.
- connector.class: es el tipo de conector que tiene la plataforma confluent.
- tasks.max: son las tareas máximas que se asignan al conector por defecto se pondrá 1
- connection.url: aquel se debe asignar los valores que se necesitan y se tienen que ajustar para configurar la conexión de base de datos MySQL.
- incrementing.column.name: Es el nombre de la columna de incremento que se encuentra en las tablas de la base de datos para encontrar nuevas inserciones de filas esta columna no puede ser nula.
- topic.prefix: corresponde al nombre del tema al que el conector publica los mensajes.

Aclarados los parámetros la configuración del conector de Kafka a MySQL es la que se muestra en la Figura 33.

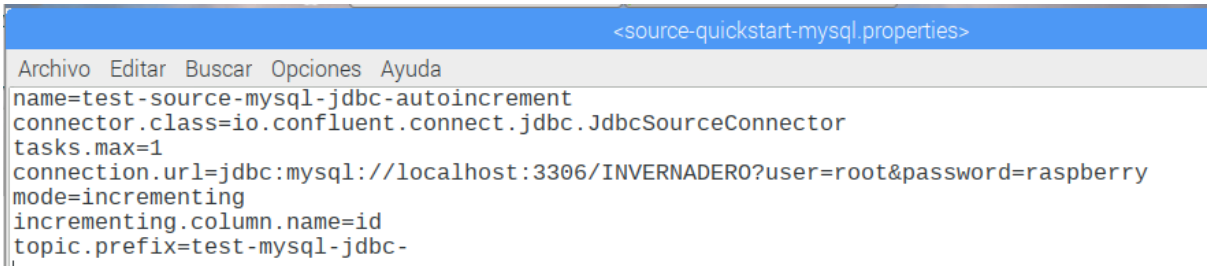


Figura 33 Configuración del conector Kafka a MySQL.

Fuente: (Tutorial Kart)

3.4.7 Instalación y configuración de Elasticsearch

ElasticSearch nos servirá como motor de búsqueda para que la configuración nos admita analizar y buscar los registros que se van apareciendo en Kafka de forma más eficiente y mucho más rápido.

Previo a la configuración se requiere que ElasticSearch sea descargado de su sitio web e instalado. Para este proyecto utilizamos la versión 5.5.2.

La configuración se realiza editando el archivo /etc/elasticsearch/elasticsearch.yml y se tiene que quitar los comentarios y cambiar las configuraciones cluster.name, network.host y http.port como se muestra en la Figura 34:

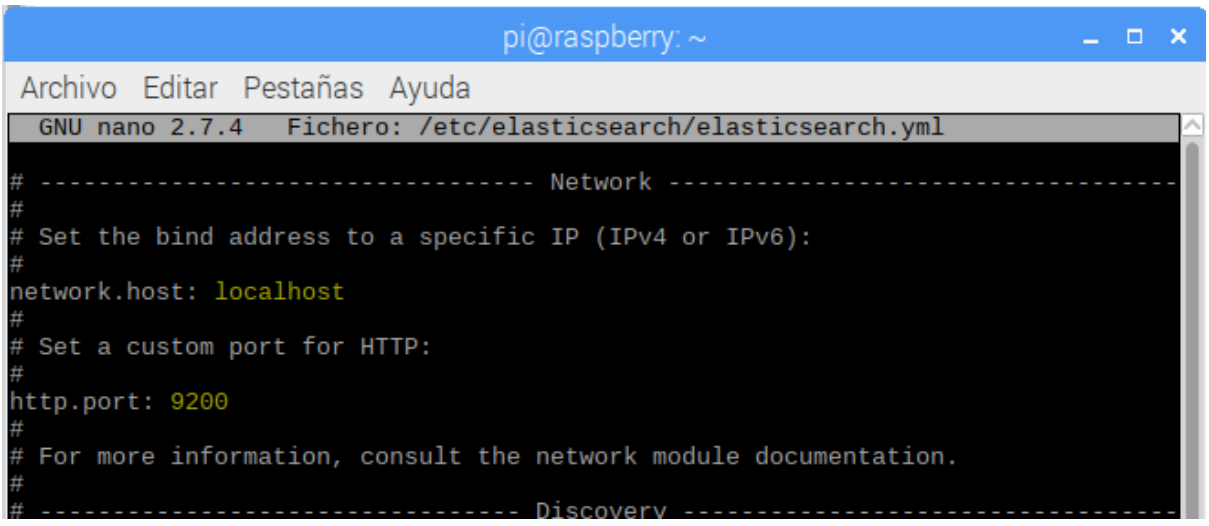


Figura 34 Configuración del ElasticSearch

3.5 DISEÑO Y DESARROLLO DE DASHBOARD DE CONTROL PARA LA VISUALIZACIÓN DE DATOS.

3.5.1 Introducción.

Para ver los datos en tiempo real necesitamos una herramienta existen muchas alternativas unas pagadas otras libres para visualizar los datos en nuestro caso el sistema embebido utilizara Grafana que es una herramienta de código abierto muy popular para visualizar los datos de los sensores.

3.5.2 Grafana.

Debido al alcance y objetivos propuestos en esta investigación, es esencial que los dashboard (tablero) de control manipule relaciones de datos, es por eso por lo que se utilizará Grafana, que tiene la capaz de manejar relaciones. Grafana se utiliza con generalmente y con mayor frecuencia para visualizar datos de series temporales y suministra una forma óptima y elegante de crear, explorar y compartir dashboard. También, Grafana posee una capa de autenticación propia. Con esta configuración, la capa de presentación de datos pude manipular las relaciones entre los datos, pero en ocasiones los inconvenientes con las consultas complejas permanecen, así como las relaciones-limitaciones de Elasticsearch.

Grafana es un programa interactivo, que permite al usuario generar dashboard en datos almacenados en un archivo de disco. La principal ventaja y originalidad del sistema es que el usuario puede determinar interactivamente el efecto visual de un dashboard y verlo inmediatamente en la pantalla. Además, cualquier reporte originado por Grafana puede ser modificado por un editor gráfico de propósito general. Este sistema permite al usuario alcanzar exactamente el dashboard correcto para cualquier libro, papel o informe.

El marco de trabajo del panel de control de Grafana (Grafana Labs, 2018) ha sido selecto como herramienta de visualización web. Permite que los usuarios creen, empleando el uso de la Web en un navegador de Internet común, sus propios tableros de control que definen las gráficas que obtienen los datos del almacén de datos de ElasticSearch con Kafka. Grafana también proporciona un rápido despliegue debido a su servidor web integrado y permite diferentes roles de usuario con la definición de políticas de autenticación y autorización personalizadas.

3.5.3 Instalación y configuración de Grafana en Raspberry Pi

Ya que Grafana es una aplicación web, también podemos crear un servidor Web en la Raspberry Pi e instalar la versión de Grafana 4.6.2 del repositorio: https://github.com/fg2it/grafana-on-raspberry/releases cuyo objetivo es de proporcionar paquetes Grafana deb para Raspberry Pi y arm64. Estos paquetes son ediciones no oficiales, reestructuradas con la minúscula cantidad de cambios y modificaciones posible del repositorio original de Grafana, en la Figura 35 se puede apreciar como descargar la versión mencionada a la Raspberry Pi.

Figura 35 Descargar de Grafana

Adicional también existen librerías o complementos adicionales que pueden ser descargados, a continuación, se describe algunos complementos que serán utilizados para diseñar los tableros de control.

- Panel de Reloj: el panel del reloj logra mostrar la hora actual o una cuenta regresiva y actualizaciones en intervalo de segundos. Se visualiza la hora en dependiendo la zona horaria configurada y también puede visualizar una cuenta regresiva para un evento.

- Panel discreto: este panel logra mostrar valores discretos de forma horizontal de un gráfico. Esto permite visualizar claramente transiciones de estado. Es una opción buena para mostrar datos de que son de tipo cadena o booleanos.

- Panel Grafana Gauge: este plugin de panel suministra un panel de indicador basado en D3 (Documentos manejados por datos) que brinda todas las capacidades de los navegadores modernos sin vincularse a un marco propietario, combinando poderosos componentes de visualización y un enfoque basado en datos para la manipulación para Grafana como se puede visualizar en la Figura 36.

Figura 36 Panel Grafana Guage

Fuente: (Grafana Labs, 2018)

- Plugin Cal-HeatMap: Este complemento suministra un calendario con heatmap, como el calendario de contribución de GitHub. Este complemento recoge datos de series temporales únicas, como datos de temperatura. Los datos se pasan al módulo javascript Cal-heatmap para graficar en un calendario un mapa de calor como ejemplo en la Figura 37.

  Para poder ver datos de series numéricas como la temperatura registrada cada intervalo de tiempo puede ser minutos, es posible que deba declarar el dominio y el intervalo de forma conveniente.

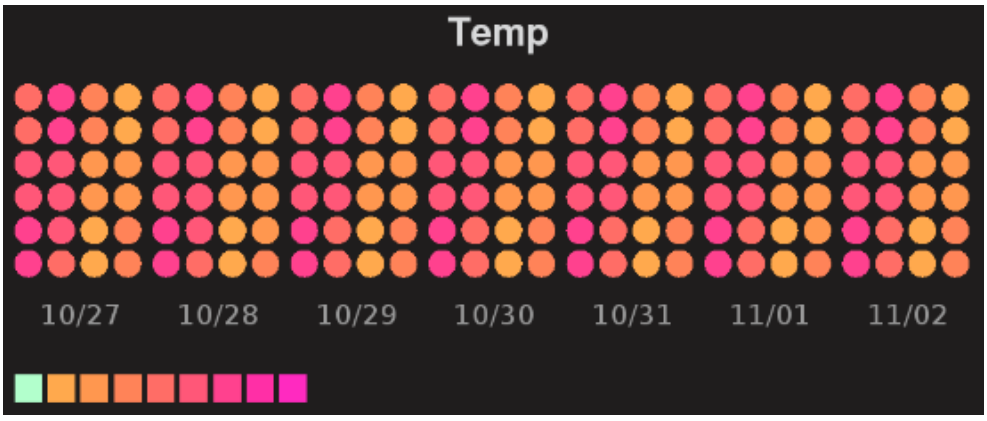


Figura 377 Plugin Cal-HeatMap; ejemplo temperatura

Fuente: (Grafana Labs, 2018)

3.5.4 Diseño del dashboard de control.

A continuación, se detalla cómo se diseña el dashboard de control donde se visualizará los datos que serán tomados por el sistema embebido. Como bien ya se explicó Grafana nos presenta muchos complementos para visualizar los registros en tiempo real utilizaremos el complemento Table que se muestra en la Figura 38.

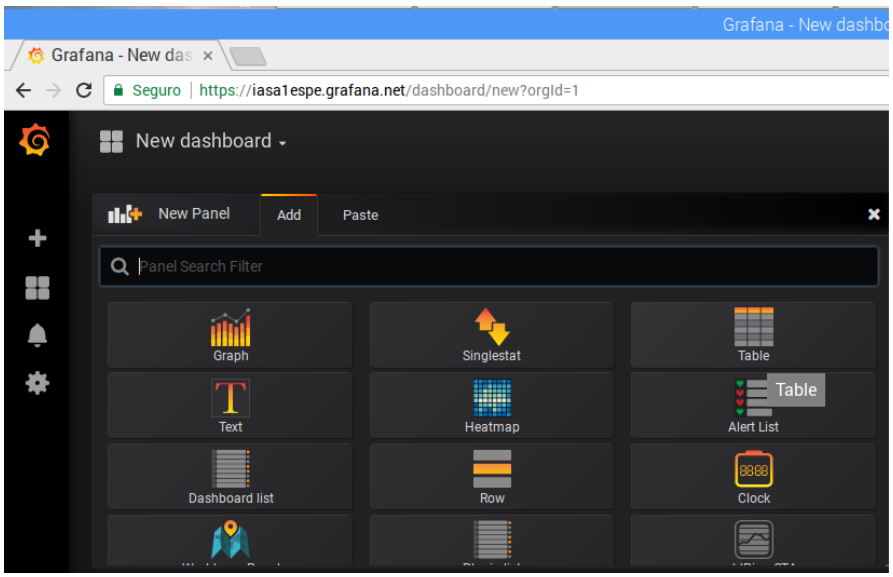


Figura 38 Complemento Table para visualizar los datos de los registros.

Como ya se mencionó se agregó el panel de Clock que muestra la hora actual en intervalo de segundos. Se agrega este panel también como se puede visualizar la hora en la Figura 49.

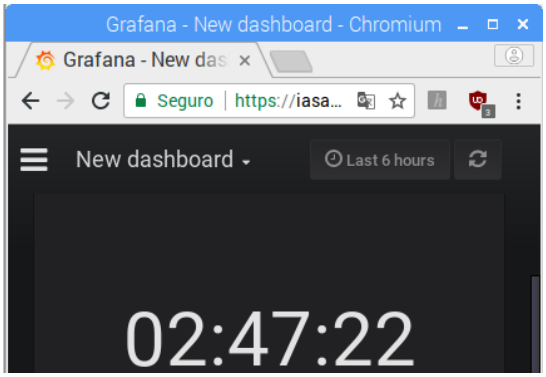


Figura 39 Complemento Panel de Clock.

Para cada sensor ambiental se presentará su promedio, el máximo y el mínimo dato por cada nodo y se utilizará el Panel Graph en donde se muestra en el eje de las “x” la hora en la

cual fue adquirido los datos y en el eje “y” el valor de la medida, como por ejemplo la temperatura que se muestra en la Figura 40.

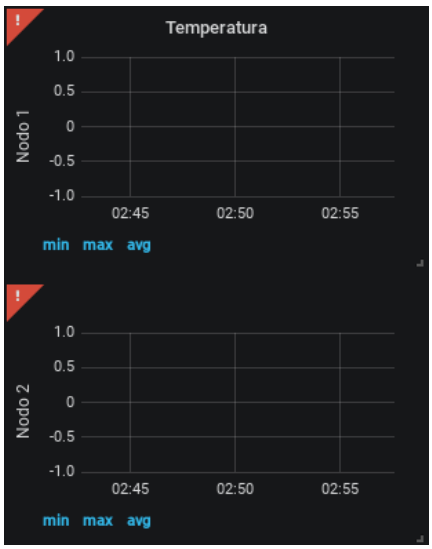


Figura 40 Complemento Panel de Graph.

Integrando todos los paneles en un solo hoja de trabajo se obtiene el dashboard de control donde se puede se visualizará los valores tomados por los sensores ambientales cuando el sistema embebido esté ejecutándose, como se muestra en la Figura 41.

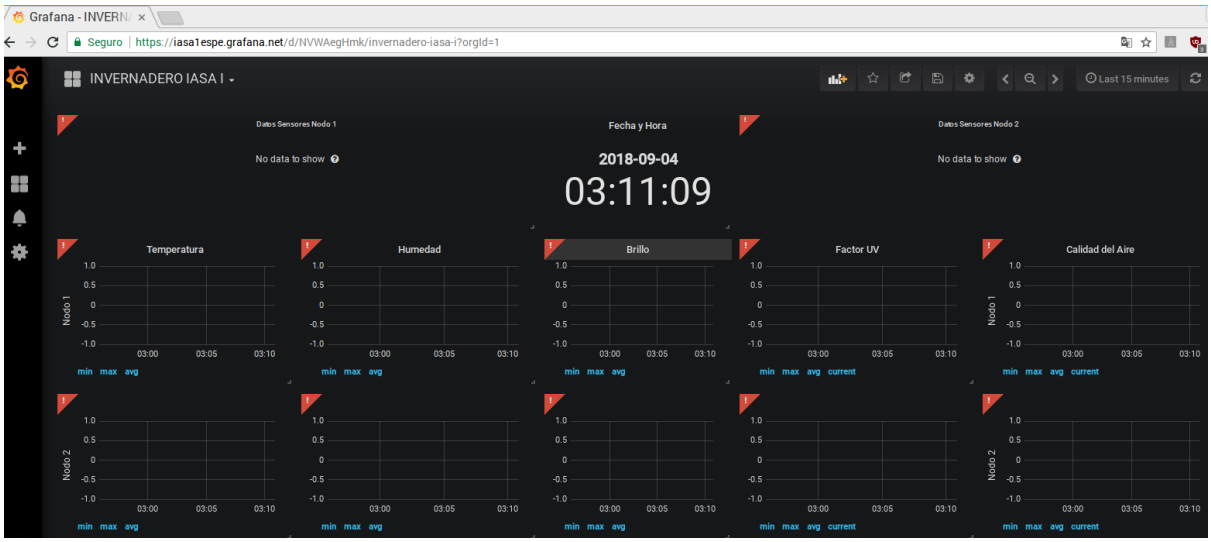


Figura 41 Dashboard de control.

CAPÍTULO IV
IMPLANTACIÓN Y PRUEBAS

4.1 IMPLANTACIÓN DEL SISTEMA EMBEBIDO.

4.1.1 Lugar de implantación del sistema embebido.

En la Figura 42 se visualiza el lugar en donde el sistema embebido fue implantado en nuestro caso práctico es el invernadero de rosas de la ESPE -IASA I, ubicado en la hacienda "El Prado" ubicado en la provincia de Pichincha del cantón Rumiñahui y parroquia Sangolquí con los siguientes puntos cardinales: Latitud: -0.383333 y Longitud: -78.4167.

Figura 42 Invernadero IASA I

4.1.2 Prototipo del sistema embebido.

En la Figura 43 se visualiza el prototipo del dispositivo de monitoreo de actividad ambiental del sistema embebido, En la Figura 44 se puede ver los componentes electrónicos o sensores ambientales que fueron implantadas. Cabe decir que la prueba obtener los datos ambientales que el invernadero presentaba posteriormente se tomaban datos manuales en la mañana y tarde y estos datos eran registrados en un documento Excel.

Figura 43 Prototipo del Sistema Embebido

Figura 44 Sensores Ambientales

4.1.3 Inicializar sistema embebido.

Después de haber implantado el sistema embebido es necesario configurar la inicialización de la programación que se configuró en la Raspberry Pi, cabe mencionar que se tiene que ejecutar de forma automática cada vez que se encienda el equipo, o en el caso de que se reinicie de igual forma comenzar su inicialización de forma automática, para lo cual se ha configurado los comandos que se tienen que ejecutar con la aplicación Crontab.

Entre las funciones de Crontab está la programación del controlador que se utilizan, entonces el sistema embebido puede funcionar sobre la base de un horario establecido, es muy ventajoso para encender y apagar los dispositivos. La forma de autenticación es por credenciales es decir el nombre de usuario y contraseña se utilizará para proteger el sistema de ser accedido por cualquier persona. En la fase final de la investigación, el sistema de monitoreo en tiempo real se evalúa cada cierto tiempo utilizando la función Crontab. Mientras tanto, el tiempo de respuesta del sistema embebido está dentro de un rango de 1-2 segundos. En la Figura 45 se muestra que se ejecutará el programa: inicializar_servicios.sh cada vez que la Raspberry pi sea encendida.

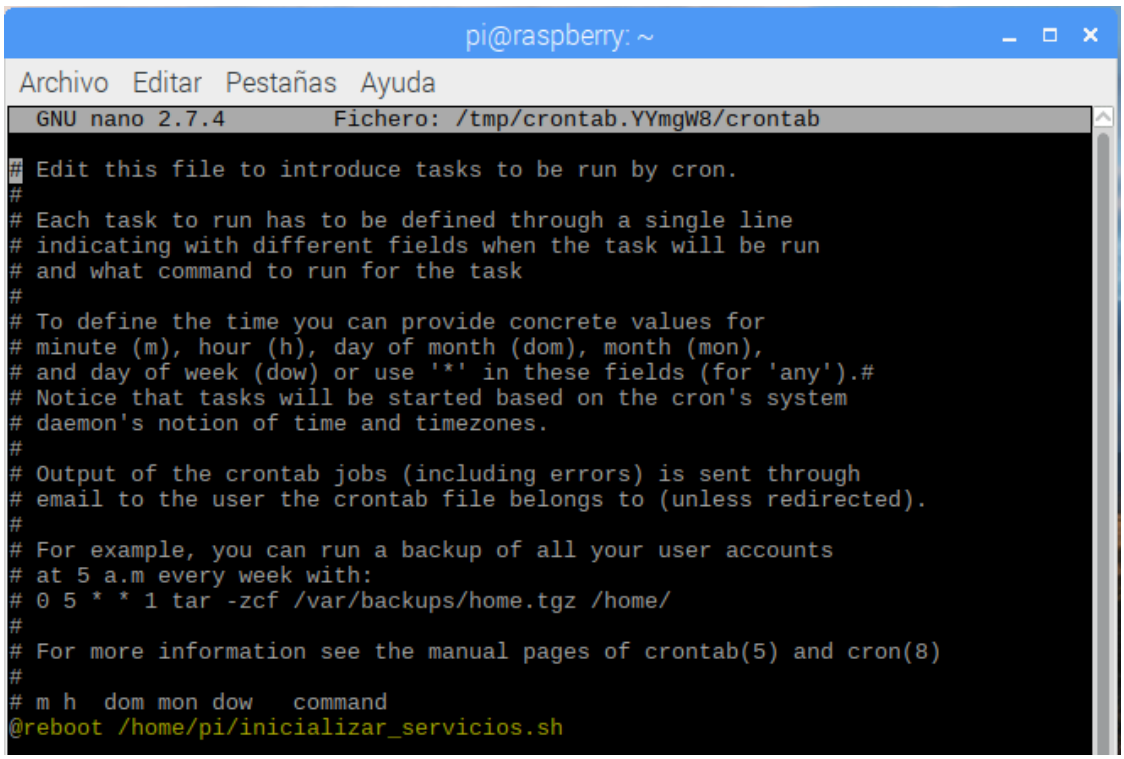


Figura 45 Inicialización del sistema por Crontab.

4.2 PRUEBAS DE CAPTACIÓN DE DATOS.

Después de haber inicializado el sistema embebido automáticamente es necesario comprobar la captación de los datos ambientales para lo cual de forma automática después de iniciar el sistema se deberá visualizar ya la toma de datos en tiempo real, cabe mencionar que se tiene que ejecutar de forma normal la captación de datos para poder visualizar como se muestra en la Figura 46.

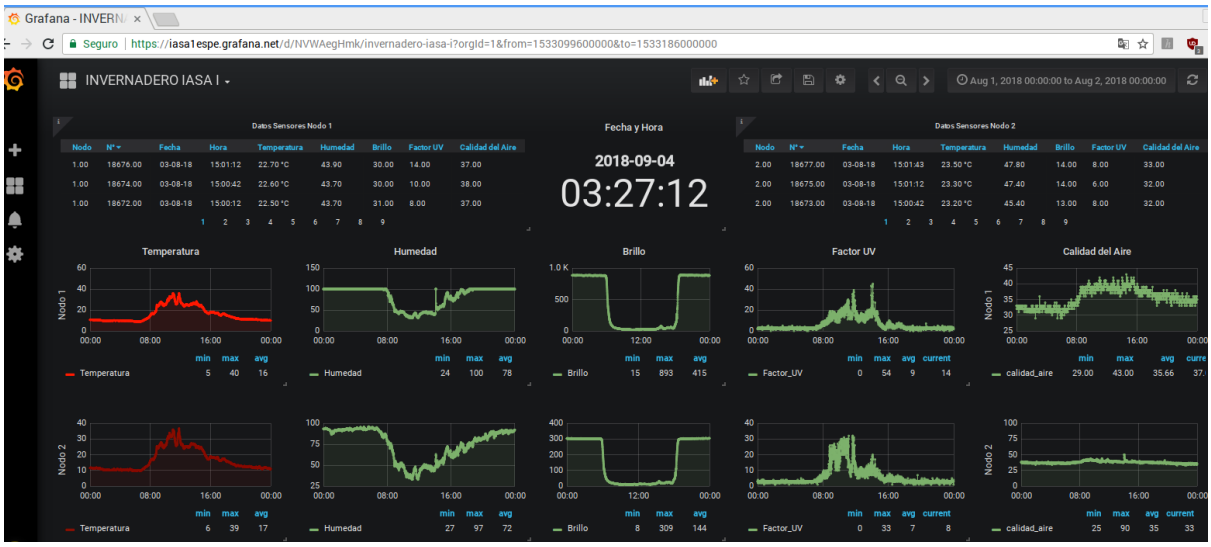


Figura 46 Visualización de la toma de datos.

4.2.1 Interpretación de los Resultados

De acuerdo con los resultados de los instrumentos aplicados, se puede llegar a las siguientes conclusiones:

Respecto a la toma de datos, se evidencia comodidad al momento de usar la aplicación, Y en lo referente a la receptividad de su interfaz los resultados arrojan que se puede apreciar la fácil visualización y entendimiento de las métricas tomadas en tiempo real ya que su usabilidad que tiene y consideran su interfaz amigable y de practico uso, por otro lado, al momento de recolectar las métricas tomadas las personas que miden los datos manualmente afirman que representa una herramienta significativa en su modelo de impartir conocimiento y ha tenido un impacto positivo, también que el sistema embebido supone un método eficiente y eficaz a la hora de evaluar el estado del invernadero.

Por otro lado, en lo que concierne a las funcionalidades del sistema embebido y a su entendimiento los datos receptados muestran que no se presentaron ningún obstáculo al momento de la toma de datos el requisito indispensable que siempre el equipo debe estar conectado a la fuente de poder.

Finalmente, en lo que respecta a tener una aplicación web por medio de la herramienta Grafana se manifiesta que la aplicación posee adaptabilidad y versatilidad en cuanto a los datos mostrados en que produce el sistema embebido y que pueden utilizarse para acceder al sistema y que se puede ingresar al sistema mediante cualquier navegador web que sea de la preferencia del usuario , además, que la herramienta web presenta una interfaz amigable para con el Usuario y asociado a eso que la aplicación resulta efectiva al momento de gestionar el intercambio de información del estado del invernadero.

CAPÍTULO V
CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

Con el rápido avance de la tecnología, los dispositivos automatizados están ganando popularidad en el campo de Internet de las cosas y los sistemas embebidos. La capacidad de estos dispositivos para conectarse y compartir recursos a través de Internet es cada vez más omnipresente. Aunque los dispositivos generan grandes volúmenes de datos, es necesario procesarlos de forma efectiva y extraer la información útil para su posterior procesamiento y almacenamiento.

Siempre que hay cambios en el determinado tiempo de un sensor ambiental, el cambio es captado primero, después es almacenado y a través del uso de un conocido sistema de mensajería llamado Apache Kafka. Utiliza el concepto de temas para publicar y suscribir mensajes. Además, las APIs de Java se utilizan para escribir aplicaciones para productores y consumidores que realizan la canalización de datos intermedia necesaria.

La idea de utilizar Kafka era proporcionar tolerancia a los fallos tolerando la pérdida de la conexión a Internet. Esto significa que, si no hay comunicación entre la Raspberry local de Kafka y la herramienta de visualización Grafana, los datos se almacenarán dentro de los temas de la base de datos MariaDB. Además, este Kafka también mostró tolerancia a fallos en el que el único corredor deberían estar en funcionamiento en nuestro caso. Esta tolerancia a fallos también se observó en el caso de tres dos nodos que se estaba ejecutando. Con este marco, era fácil elegir la ubicación del procesamiento de datos en términos de Raspberry local o remoto, independientemente de la arquitectura subyacente

La metodología Ralph Kimball es recomendable y muy apropiada, por su ágil y su ciclo de vida de desarrollo, se puede alcanzar un producto entregable y funcional en base a los requisitos inicialmente levantados de forma correcta.

La red de sensores inalámbricos (WSN) es un sistema que produce una gran cantidad de datos. Por eso, se necesita el sistema que sea capaz de manejar la gran cantidad de datos. Para superar esto, esta investigación integra el sistema WSN estándar con un gran marco de datos con un enfoque de streaming de datos. En la implementación, el sistema utilizó nodos sensores de fabricación propia que recopilan datos de calidad de aire junto con otros datos ambientales: temperatura del aire, humedad del aire, intensidad de la luz y factor UV. Usando la base de datos de Kafka e Impala, este sistema es capaz de transmitir una gran cantidad de información. de datos en tiempo casi real. En general, el sistema propuesto ofrece un buen rendimiento y un rendimiento fiable.

5.2 Recomendaciones

Después de concluir lo anterior se puede afirmar que como recomendación la implantación de más sistemas embebidos en varios puntos del invernadero permitiría que se pueda monitorear los estados ambientales siendo más efectivos al tener resultados o métricas ambientales que permitirían tomar decisiones y así beneficiar a la producción en este caso de rosas.

La comparación de la toma de datos del sistema embebido con otras herramientas de que también nos dan métricas debe ser meticulosa y tomar su respectivo tiempo, a fin de tener la mejor opción para el estudio de las condiciones del invernadero en base a los requerimientos funcionales y no funcionales.

Para el desarrollo de proyectos se recomienda tener una buena conexión a internet y también que la fuente de energía sea estable con el fin de que el sistema embebido funcione correctamente y así prevenir fallos con el sistema.

Teniendo en cuenta todos estas recomendaciones se puede decir que la implantación de un sistema embebido con mensajería instantánea permitir diagnosticar el estado del invernadero de forma asertiva y así solucionar las falencias en el sistema de toma de datos no solo en la floricultura se podría utilizar en otros ámbitos como la ganadería, la producción avícola lo que signifiquen una mejor toma de decisiones que puedan beneficiar a la producción y con un bajo costo comparado con sistemas o equipos demasiados sofisticados y con alto costo

REFERENCIAS BIBLIOGRAFICAS

Adhikari, S. (2013). *Embedded Operating Systems and Linux*. Retrieved from https://pdfs.semanticscholar.org/presentation/1bc4/330c84a32a6f459d1b3378c88878f013ed85.pdf

Ana Barragán, A. F. (2013). *Universidad Católica de Colombia.* Retrieved from https://repository.ucatolica.edu.co/bitstream/10983/690/2/IMPLEMENTACION%20DE%20UNA%20BASE%20DE%20DATOS%20NOSQL%20PARA%20LA%20GENERACION%20DE%20LA%20MATRIZ%20OD.pdf

Apache Kafka. (2017). *Apache Kafka*. Retrieved from https://kafka.apache.org/

Campo, D. (2017, Noviembre 17). *Evaluación y prueba de concepto de tecnologías de procesado en streaming y tiempo real*. Retrieved from https://blog.gft.com/es/2017/11/17/evaluacion-y-prueba-de-concepto-de-tecnologias-de-procesado-en-streaming-y-tiempo-real/

Castro Romero, A. (2012, Diciembre 12). ACM. *Revista de la Universidad Pedagógica y Tecnológica de Colombia*. Retrieved from ACM: http://revistas.uptc.edu.co/index.php/ingenieria/article/view/2115/2078

Clarke, M. J., Akeroyd, F. A., & Moore, L. A. (2017). Live Visualisation of Experiment Data at ISIS and the ESS. *ICALEPCS2017*, 431-437. Retrieved from http://inspirehep.net/record/1656170/files/tupha029.pdf

CNXSOFT. (2016). *CNXSOFT – EMBEDDED SYSTEMS NEWS*. Retrieved Febrero 04, 2018, from BigProf: https://www.cnx-software.com/wp-content/uploads/2016/02/Raspberry_Pi_3_Large.jpg

Como Hacer. (2014, Agosto 12). *Como Hacer*. Retrieved Febrero 11, 2018, from https://comohacer.eu/comparativa-y-analisis-raspberry-pi-vs-competencia/

Confluent. (2017). Retrieved from https://www.confluent.io/

Digi-Key Electronics. (n.d.). *Piot 101: Raspberry Pi + Internet of Things*. Retrieved from https://www.digikey.com/en/maker/projects/40cc047a104740fe9e63ef4e45fb1e19

Eichelberger, H. (2008, Septiembre 16-17). Automatic Layout of UML use case diagrams. *Proceeding*, 105-114. Retrieved from http://thescipub.com/pdf/10.3844/jcssp.2014.1440.1446

García Moya, A., & Barriga Barros, A. (2012, Noviembre). Curso Práctico de Sistemas Empotrados Basado en Placas de Desarrollo XUPV2P. *Revista Iberoamericana de Tecnologías del/da Aprendizaje/Aprendizagem, 7*(4), 231-237. Retrieved Agosto 1,

2018, from https://pdfs.semanticscholar.org/d421/6cbdd4864cffee30d7f60decb829ccb5a0b3.pdf

Grafana Labs. (2018). *Grafana*. Retrieved from https://grafana.com/grafana

Huong, N. (2017). *Evaluation of a Data Messaging System Solution, Case: Evaluation of Apache Kafka™ at Accanto Systems.* Lahti University of Applied Sciences, Spring. Retrieved from https://www.theseus.fi/bitstream/handle/10024/130611/Nguyen_Huong.pdf?sequence=1

Junqueira, F., & Reed, B. (2013). *Zookeeper - Distributed Process* (Primera ed.). O'Reilly. Retrieved from https://t.hao0.me/files/zookeeper.pdf

Kamalrudin, M., Ahmad, S., & Ikram, N. (2017). *Requirements Engineering for Internet of Things : 4th Asia-Pacific Symposium, Apres 2017, Melaka, Malaysia, November 9-10, 2017, Proceedings.*

Kimball, R., Ross, M., Becker, B., Mundy, J., & Thornthwaite, W. (2015). *The Kimball Group Reader: Relentlessly Practical Tools for Data Warehousing and Business Intelligence Remastered Collection* (Segunda ed.). USA. Retrieved from http://www.darsbama.com/wp-content/uploads/2018/07/Kimball-Relentlessly-Practical-Tools-for-Data-Warehousing-and-Business-Intelligence-Remastered-Collection.pdf

Max FIZZ Technologies. (2018, Febrero). *Components of Embedded System*. Retrieved Agosto 20, 2018, from Max FIZZ Technologies: https://www.maxfizz.com/components-of-embedded-system/#Communication_Ports

Mejía, C., Ríos, A., León, L., & Hidrobo, F. (2010, Abril-Julio). Una plataforma tiempo real para ejecutar algoritmos de control A real-time platform to execute control algorithms. *Revista Ciencia e Ingeniería, 31*(2), 91-100. Retrieved from Europa Ec: https://docplayer.es/1228060-Una-plataforma-tiempo-real-para-ejecutar-algoritmos-de-control-a-real-time-platform-to-execute-control-algorithms.html

PowerData. (2016). *PowertData, tu aliado estratégico*. Retrieved 08 18, 2017, from PowerData: https://cdn2.hubspot.net/hubfs/239039/Ebooks/Guia_PowerData_Tu_aliado_estrategico.pdf?t=1501669919365

Raghavan, P., Amol, L., & Neelakandan, S. (2006, Diciembre 9). *Embedded Linux System – Design and Development.* Auerbach: Auerbach Publications Taylor & Francis Group, LLC. Retrieved Julio 31, 2018, from http://www.esys.ir/Files/Ref_Books/Linux/esys.ir_Embedded.Linux.System.Design.and.Development.pdf

Raspberry Pi Foundation. (n.d.). *RaspberryPi.org*. Retrieved from RaspberryPi: https://www.raspberrypi.org/products/

Shapira, G., Narkhede, N., & Palino, T. (2017). Kafka: The Definitive Guide. In *Kafka: The Definitive Guide: Real-Time Data and Stream Processing at Scale.* O'Reilly Media, Inc. Retrieved from http://book.huihoo.com/pdf/confluent-kafka-definitive-guide-complete.pdf

Singh, R., Gehlot, A., Singh, B., & Choudhury, S. (2017). Arduino-Based Embedded Systems. In *Arduino-Based Embedded Systems* (p. 330). Boca Raton. Retrieved Febrero 04, 2018, from BigProf: https://www.taylorfrancis.com/books/9781351669542

Ta, V., Liu, C., & GW, N. (2016). Big data stream computing in healthcare real-time analytics. *Cloud Computing and Big Data Analysis (ICCCBDA)*, 37-42.

Tutorial Kart. (n.d.). *Kafka Connector to MySQL Source - Confluent JDBC Driver - Example* . Retrieved from https://www.tutorialkart.com/apache-kafka/kafka-connector-mysql-jdbc/

Wang, G., Koshy, J., & Subramanian, S. (2015, Agosto). Building a Replicated Logging System with Apache Kafka. *Proceedings of the VLDB Endowment, 8*(12), 1654-1655. Retrieved from Piano Nazionale Scuola Digitale. (PNSD): http://www.vldb.org/pvldb/vol8/p1654-wang.pdf