Control automático adaptativo por medio del ciclo MAPE-K, en el Internet de las Cosas, usando el System On Chip *ESP32*

César Augusto Álvarez Gaspar

Universidad Autónoma de Manizales
Maestría en Ingeniería
Manizales, Colombia
2019

Control automático adaptativo por medio del ciclo MAPE-K, en el Internet de las Cosas, usando el System On Chip *ESP32*

César Augusto Álvarez Gaspar

Trabajo de grado presentado como requisito parcial para optar al título de:
Magister en Ingeniería

Director(a):
Ph.D. José Luis Rodríguez Sotelo
Codirector(a):
M.Sc. Rubén Darío Flórez Hurtado

Línea de Investigación:
Mecatrónica y Control
Grupo de Investigación:
Automática

Universidad Autónoma de Manizales
Maestría en Ingeniería
Manizales, Colombia
2019

La ciencia puede divertirnos y fascinarnos, pero es la ingeniería la que cambia el mundo

Isaac Asimov

A Dios, mi familia y a todas las personas que creyeron en mí. Gracias por todos los momentos en los cuales me apoyaron y me dieron las herramientas para continuar.

Agradecimientos

Agradezco a la Universidad Autónoma de Manizales por la formación postgradual que me brindo. A la Universidad del Quindío en donde me forme en pregrado y en la que actualmente laboro. Al Ph.D. José Luis Rodríguez Sotelo por su acompañamiento en esta jornada. A mis compañeros de curso, por los momentos de alegría y camaradería durante estos años. Finalmente, y no menos importante a Dios y a mi familia por darme la fuerza para alcanzar mis sueños.

Resumen

El Internet de las Cosas (IoT) es una revolución tecnológica, en la que se interconectan personas, procesos, datos y objetos inanimados, usando el Internet como puente entre ellos. Estas interacciones ofrecen nuevas posibilidades para mejorar la calidad de vida de las personas al igual manera que mejorar algunos procesos industriales. Esta tesis de Maestría aportó al desarrollo de aplicaciones basadas en el Internet de las Cosas, al proporcionar un *Objeto de Internet* que capaz de realizar control automático adaptativo por medio del ciclo Monitorear, Analizar, Planificar, Ejecutar y Conocer (MAPE-K). Este *Objeto de Internet* se desarrollo por medio del System On Chip *ESP32*, de la empresa Espressif. El *Objeto de Internet* se probo en sistemas electrónicos Lineales e Invariantes en el Tiempo (LTI). El sistema desarrollado mostró ser capaz de realizar el control de estos y adaptarse a cambios en su dinámica interna. Durante el desarrollo de esta tesis se investigó en las áreas del control automático, sistemas embebidos, inteligencia artificial y desarrollo de Aplicaciones Web.

Palabras clave: Internet de las Cosas, Control Adaptativo, MAPE-K, Esp32.

Abstract

The Internet of Things (IoT) is a technological revolution, in which people, processes, data and inanimate objects are interconnected, using the Internet as a bridge between them. These interactions offer new possibilities to improve the quality of life of people as well as improve some industrial processes. This Master's thesis contributed to the development of applications based on the Internet of Things, by providing an Internet Object capable of performing adaptive automatic control through the Monitor, Analyze, Plan, Execute and Know cycle (MAPE-K). This Internet Object was developed by means of the System On Chip ESP32, from the company Espressif. The Internet Object was tested in Linear and Time Invariant (LTI) electronic systems. The developed system showed to be capable of controlling these and adapting to changes in it's internal dynamics. During the development of this thesis, research was carried out in the areas of automatic control, embedded systems, artificial intelligence and the development of Web Applications.

Contenido

Lista de Figuras

Lista de Tablas

Lista de símbolos

Símbolos con letras latinas

Símbolo	Término	Unidad SI	Definición
e	Error	V	Diferencia entre r e y
h	Señal Actuante	V	Señal filtrada de entrada del sistema a controlar
r	Referencia	V	Señal de referencia para un sistema de control
u	Señal actuante	V	Señal de entrada del sistema a controlar
y	Variable Controlada	V	Señal de salida del sistema a controlar

Abreviaturas

Abreviatura	Término
ADC	Conversor Análogo Digital
AJAX	Asynchronous Javascript and XML
API	Application Programming Interface
CAN	Campus Area Network
CPU	Unidad Central de Procesamiento
CRUD	Crear, Leer, Actualizar y Borrar (Create, Read, Update, Delete)
CSS	Hoja de Estilo en Cascada

Abreviatura	**Término**
DAC	Conversor Digital Análogo
EN	Pin de habilitación de funcionamiento (Enabled)
Ftp	File transfer protocol
GND	Nivel de voltaje de 0V (Ground)
GPIO	Puerto digital de propósito general
HTML	Lenguaje de Marcas de Hipertexto
Http	Hypertext transfer protocol
IDE	Entorno de Desarrollo Integrado
IoT	Internet de las Cosas (Internet of Things)
I^2C	Inter-Integrated Circuit
JS	JavaScript
JSON	JavaScript Object Notation
LAN	Local Area Network
LED	Diodo Emisor de Luz
LoRa	Long Range
LTE	Long Term Evolucition
LTI	Lineal Invariante en el Tiempo (Linear Time-Invariant)
MAN	Metropolitan Area Network
MAPE-K	Monitorear, Analizar, Planificar, Ejecutar y Conocer
MIT	Massachusetts Institute of Technology
MQTT	Message Queue Telemetry Transport
M2M	Comunicación Maquina a Maquina
MVC	Modelo - Vista - Controlador
OLED	Diodo Organico Emisor de Luz
ORM	Object - Relational Mapping

Abreviatura	**Término**
PaaS	Platform as a Service
PAN	Personal Area Network
PCB	Printed Circuit Board
PHP	Hypertext Preprocessor
PID	Proporcional, Integrativo y Diferencial
PWM	Modulación por Ancho de Pulso
P2M	Comunicación Persona a Maquina
P2P	Comunicación Persona a Persona
RAM	Memoria de Acceso Aleatorio
ROM	Memoria de Solo Lectura
RFID	Identificación por medio de Radio Frecuencia.
SNA	Sistema Nervioso Autónomo
SPI	Serial Peripheral Interface
SoC	System on Chip
SQL	Structured Query Languaje
SVP	Pin positivo de Amplificador de bajo ruido en ESP32
USB	Universal Serial Bus
TCP/IP	Transmission Control Protocol/Internet Protocol
WiFi	Protocolo de Internet inalámbrico regido por la Norma IEEE 802.11.
3V3	Nivel de voltaje de 3,3V
μC	Microcontrolador

Lista de Anexos

1. Presentación

El Internet de las Cosas (IoT) es un entorno donde los objetos inanimados se conectan a las personas, datos, procesos y entre sí mismos por intermedio de la Internet, con el objetivo de mejorar la calidad de vida de los hombres y mujeres, en conjunto con la efectividad de los desarrollos industriales [1]. A fin de llevar esto a cabo, se dota a cualquier objeto inanimado de un computador embebido capaz de conectarse al ciberespacio, creando un *Objeto de Internet*. Algunos de estos poseen sensores y actuadores que los dotan de la capacidad de interactuar con su medio ambiente. Son numerosos los aportes propuestos con base en el uso del IoT. De ellos se ilustra el caso de una silla que pueda monitorear la postura de su ocupante y guardar las estadísticas de su uso. Este registro puede ser consultado por un médico durante una consulta y a partir del reporte del mueble producir un análisis clínico con la finalidad de tomar acciones preventivas o correctivas para el paciente. Similar al anterior ejemplo se proponen múltiples aplicaciones más, tantas como la imaginación de investigadores, ingenieros, profesionales y entusiastas lo permitan.

Durante los últimos tres siglos se han realizado una serie de cambios profundos al desarrollo de la humanidad enmarcados en los avances procesos productivos. Según Schwab [65] son cuatro revoluciones industriales enmarcadas en las tecnologías disponibles en cada época. La primera ocurre en *1794* con la aplicación del telar mecánico a la manufactura textil. La segunda aparece con la implementación de la cinta transportadora en el matadero de Cincinnati, en *1870*. La tercera con el uso del primer Controlador Lógico Programable (PLC), en 1969, el Modicon 084. Finalmente, en la última revolución es protagonista el uso de sistemas físicos cibernéticos, como el Internet de las Cosas (IoT), el Sistema de Posicionamiento Global (GPS), la Inteligencia Artificial (IA), la impresión 3D y la Big Data, por mencionar algunos. El inicio real de esta industria 4.0 no es muy claro.

Los efectos del Internet de las Cosas en la industria 4.0 se perfilan en el sector salud, agrícola, logístico y de control de procesos [11]. En la sanidad, se plantea el desarrollo de

dispositivos para realizar pruebas clínicas no invasivas y con una alta periodicidad, como los exámenes cardíacos, visuales, auditivos y químicos. A nivel agrario, se habla de la agricultura de precisión en la cual el crecimiento de los cultivos es monitoreado y acompañado, a fin de brindar excelentes productos. En la producción vinícola, por mostrar un ejemplo, se monitorea las variables climáticas de temperatura, humedad y precipitación con el fin de predecir la naturaleza del sabor del vino añejado. Posteriormente, esta información es presenta a enólogos y compradores. El IoT posee su origen en la logística, con sus antepasados las etiquetas de Identificación por Radio Frecuencia (RFID), usados en la gestión de órdenes. En este sentido, adicionalmente, el Internet de las Cosas añade el monitoreo de condiciones ambientales claves en la conservación de mercancías. En el control de procesos, los *Objetos de Internet* conectan materias primas con la gerencia en función de la transformación de las mismas, en artículos finales, garantizando su calidad y cantidad.

Los *Objetos de Internet* están formados por computadoras pequeñas diseñadas para un único fin, llamados sistemas embebidos. Estos sistemas se componen de una parte material llamada Hardware y una intangible llamada Firmware, equivalente al Software. Una característica distintiva del IoT es la conexión que realiza entre personas, procesos, datos y objetos inanimados [1]. Esta comunicación se puede clasificar como interacciones: Persona a Persona (P2P), Persona a Maquina (P2M) y finalmente la Maquina a Maquina (M2M). En la figura **1-1** se observan los actores presentes en estos tipos de interacción. En el caso de la P2P se ejemplifica el uso de las redes sociales en dispositivos móviles, por parte de sus usuarios. En la M2P se puede mostrar la interacción de las personas con servidores, electrodomésticos e incluso casas (*SmartHouse*). En la M2M se ilustran los sistemas de climatización en grandes centros comerciales, en estos los termostatos se coordinan con los extractores y ventiladores para mantener las condiciones ambientales agradables para los compradores.

Múltiples fabricantes han optado por este nuevo mercado y han lanzado sus propios Hardware IoT. Para enumerar solo algunos, tenemos a los *ESP8266* y *ESP32*, ambos de la empresa Espressif [25]. También a *Digistump Oak* de la empresa DigiStump [24], *LightBlue* de Bean [59], *Linklt One* de MediaTek Labs [47], *Omega* de Onion [58] y *Galileo* de Intel [37], la cual ya ha sido descontinuada [7]. Por otro lado, hay tradicionales plataformas o placas de desarrollo, que han incorporado la capacidad de conectarse a Internet, para unirse a la revolución IoT, algunos ejemplos son: *Arduino/Genuino* [8], *Raspberry Pi*[44], *BBC*

Micro:bit [48], *ChipKIT Uno 32* [19], *Banana Pro* [42], entre otros.

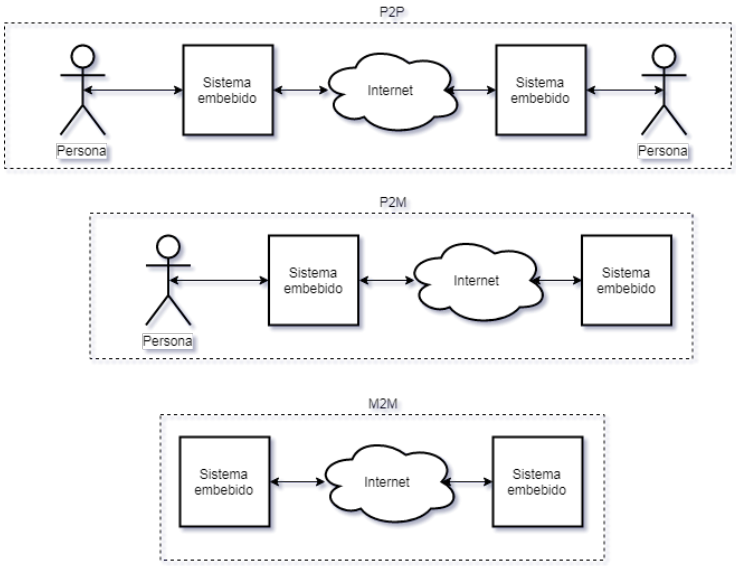

Figura 1-1.: Tipos de interacciones presentes en el Internet de las Cosas. Fuente: Propia.

El desarrollo de software para el Internet de las Cosas ha evolucionado desde las primeras lineas de código a grandes proyectos libres y propietarios. Entre los más influyentes están *ThingSpeak* de MathWorks [66], *Blynk* [15], *Mosquitto* [55], Cisco IoT [20], *Azure* de Microsoft [50], *Ubuntu Iot*, *Fiware* de la Unión Europea [29] y los proyectos Espressif Systems; por mencionar solo algunos. Los firmware desarrollados por la compañía Espressif están orientados a los dispositivos *ESP8266* y *ESP32*. En el desarrollo de software especializado para el IoT se presentan similares desafíos a los grandes proyectos de software. La implementación de herramientas modernas de ingeniería de software es esencial.

Los sistemas embebidos propuestos para el Internet de las Cosas cumplen funciones de monitoreo por medio de adquisición de datos. Otra habilidad de estos sistemas es el manejo de procesos, por medio del Control Automático, quien ha venido dotando a las máquinas con la capacidad de tomar decisiones en procesos industriales y de consumo. El desarrollo del control automático es bastante grande y cuenta con teorías robustas. Una de estas teorías es el control adaptativo, quien dota a un controlador con una flexibilidad, tal que pueda adaptarse al cambio en la dinámica de cualquier ambiente posible y sobreponerse de la mejor manera a las perturbaciones no previstas. Gracias al avance de los microprocesadores

el control digital embebido está en auge en la industria [60]. La inclusión del IoT en esta corriente es inevitable e impone nuevos retos.

Este trabajo busca proponer al ciclo MAPE-K de la Computación Autónoma como herramienta para aportar a la solución de los retos del desarrollo de software para el Internet de las Cosas, en particular, para la función de control adaptativo, pertinente para el desarrollo de la industria 4.0. Con el objetivo de realizar de esta tesis Maestría, se utilizó la metodología de desarrollo tecnológico. Con ella se buscó como objetivo principal el *Desarrollar un Control automático adaptativo por medio del ciclo MAPE-K, en el Internet de las Cosas, usando el System On Chip ESP32*. En sus objetivos secundarios se emprendió el: *Desarrollar un Objeto de Internet para realizar control automático, usando el System On Chip ESP32. Desarrollar una estrategia de control automático adaptativo por medio del ciclo MAPE-K* y finalmente *Probar el funcionamiento del Control automático adaptativo por medio del ciclo MAPE-K, en el Internet de las Cosas, usando el System On Chip ESP32.*

El documento esta dividido en capítulos y en cada uno de ellos se abordará la temática desde su particular perspectiva. En el apartado *Antecedentes* se explora el nacimiento y crecimiento del Internet de las Cosas, la aparición del ciclo MAPE-K. En el capitulo *Planteamiento del problema* se ahondará en los retos actuales del IoT y se justificará la necesidad del presente trabajo. En el *Referente teórico* se introducirán las principales teorías empleadas en la investigación. Para el quinto capítulo se declarará el objetivo principal y los secundarios. En la *Metodología* se mostrarán las directrices usadas en la investigación. En el séptimo apartado se presentaran los *Resultados* encontrados a lo largo del camino recorrido, posterior a esto, en el capitulo 8 se analizaran y discutirán estos hallazgos. Finalmente, en el capitulo 9 se recolectarán las *Conclusiones* finales de la presente tesis de maestría. Al final del documento se anexan elementos complementarios importantes para el entendimiento de la labor realizada.

2. Antecedentes

La aparición del Internet de las Cosas no es un fenómeno aislado, pertenece al desarrollo de grandes tendencias en el avance tecnológico y el crecimiento económico. En este apartado se estudiará la expansión del IoT, desde sus precedentes filosóficos, pasando por las tecnologías que aportaron a su creación y finalizando en su estado actual. De igual forma, se abordará los antecedentes del ciclo MAPE-K, que es el principal aporte de la presente tesis.

2.1. Internet de las Cosas (IoT)

Al inicio de la década de los 90s, *Mark Weiser* planteo la futura existencia de una inmersión de las computadoras en la vida cotidiana. Esta relación poseería las mismas características de otras *tecnologías profundas*. Según su articulo *The computer for the 21st Century* [73], las tecnologías más profundas son las que desaparecen, entrelazándose con el diario vivir hasta que no se logren distinguir de él. La escritura como técnica de comunicación, es un ejemplo de este concepto. En cada aspecto de las interacciones humanas se encuentra el texto, desde los libros, anuncios publicitarios, mensajes en redes sociales y otros. La lectura en el humano moderno no requiere de una atención activa para captar la información contenida en ella. El proceso pasa desapercibido, casi como si no existiera. De forma similar, Weiser plantea que los ordenadores afectaran nuestras vidas al fundirse en el trasfondo de la habitualidad. El numero de equipos de computo relacionadas con los procesos diarios superaran a la intuición. La instalación, mantenimiento y operación de los computadores no requerirán de gran concentración. En palabras del escritor: *el verdadero potencial de las tecnologías de la información, es que las computadoras alcancen al mundo real y se desvanezcan en el fondo*. A este modelo lo llamo *Computación Ubicua*. Igualmente, advierte Weiser la desaparición de las tecnologías es un fenómeno psicológico, en razón de *cuando las personas aprenden algo lo suficientemente bien, dejan de ser conscientes de ello*. Por tal

motivo, si requiere de algún tipo de atención, no es ubicua.

A finales de la década de los 90s, el Massachusetts Institute of Technology (MIT) creo el *Auto-ID Center* con el objetivo de investigar y desarrollar tecnologías y aplicaciones de identificación automatizadas. El centro desarrollo infraestructura y recomendó estándares para la identificación automatizada un mundo físico en red, por medio de Radio Frecuencia (RFID) [53]. El RFID se caracteriza por una etiqueta que contiene un circuito que al ser excitado por una señal electromagnética, retorna inalámbricamente un código único de identificación [70]. Con este número se puede monitorear su ubicación y otros datos asignados a la etiqueta. Esta tecnología fue adoptada rápidamente por el área de logística y gestión de órdenes, en empresas de mercancías [11]. Grandes superficies como Wall-mart, Tesco y organismos gubernamentales, entre ellos el Departamento de Defensa de los Estados Unidos, fueron pioneras en la adopción de esta herramienta. La incorporación del RFID proporciono grandes ventajas en el manejo de sus actividades críticas, como inventarios y en la identificación de amigos o enemigos, durante un combate [70].

En 1999, *Kevin Ashton* acuño el termino Internet de las Cosas [2] tras estudiar la tecnología RFID y plantear la conexión de objetos o cosas reales a otros objetos, por medio del ciberespacio. *Ashton* como ejecutivo de Procter & Gamble y director del *Auto -ID Center* del Massachusetts Institute of Technology (*MIT*), vio el potencial de poseer una red de cosas o *Objetos de Internet* interconectadas. Este tipo de interconexión seria máquina a máquina (*M2M*) y sin intervención humana, estableciéndose un símil con el sistema nervioso humano [11]. A partir de este punto el *MIT* lideró este sendero con su libro blanco de la conexión del mundo físico [53]. Ya para el 2002, la revista Forbes realizo un reportaje sobre el Internet de las Cosas, en el cual, se promovió la necesidad de estandarizar la forma como los computadores entienden el mundo real [64].

Para el 2006, ya se han desarrollado dispositivos electrónicos con la capacidad de conectarse a Internet, planteando el interrogante de como conectar millones de dispositivos entre sí [21]. Este tiempo se caracteriza por el diseño de arquitecturas para sistemas embebidos [74]. Estos nuevos dispositivos soportaron las aplicaciones del Internet de las Cosas. En el mundo académico se da inicio a un proceso de reflexión alrededor del Internet de las Cosas. Esto genera la publicación de una serie de artículos científicos con *Estados del Arte* que puedan dar sentido a las teorías y tendencias existentes. Entre estos trabajos encontramos a

Atzori [9], quien realiza un análisis de un paradigma del Internet de las Cosas convergiendo las distintas visiones en tan solo unas pocas. En el 2011, Bandyopadhayay [10] estudia las variadas definiciones de Internet de las Cosas aportadas hasta el momento. Concluye que el Internet de las Cosas posee una serie de desafíos, entre los que se destaca la inteligencia de las *Objetos de Internet* y su portabilidad.

Durante el 2012, Miorandi [52] presento su *Survey* sobre el Internet de las Cosas, con el cual se mostró una teoría unificada de las distintas tecnologías llamadas *IoT*. Esta suposición fundamenta las investigaciones posteriores y abre campos nuevos de investigación. Para el 2013, Gubbi [33] presento su *visión, arquitectura y futuras direcciones del Internet de las Cosas*. Este trabajo sentó las bases de la actual arquitectura del Internet de las Cosas y con este trabajo se separan las actividades del Internet de las Cosas para realizar investigaciones más a fondo en los distintos elementos del IoT. En 2016, Minuraud [51] realizó un análisis de brechas de las Aplicaciones Web o plataformas de Internet de las Cosas existentes. Este estudio se realizó para los usuarios y desarrolladores de Finlandia. De este trabajo se extrajeron los deseos e intereses actuales de los actores del Internet de las Cosas.

En el ámbito económico, grandes compañías muestran interés por el Internet de las Cosas, entre ellas, la afamada *Google* [14]. Los libros de Internet de las Cosas aparecen para el 2011 con la fundación Bankinter [11] y otras editoriales con aportes de Weber [72] y Bandyopadhyay [10]. Otro aspecto relevante hace aparición en el 2010 con la discusión de los aspectos legales y éticos del Internet de las Cosas, estudiado por Weber en [71] y [72]. Actualmente se desarrollan soluciones a distintos problemas usando tecnologías de prototipado provistas por grandes empresas como Google [32], Mathworks [66] y Espressif[26], entre otros.

2.2. Ciclo MAPE-K

Antes de la aparición del concepto del ciclo *MAPE-K*, se originó el proyecto Computación Autónoma. Este nació de la mano de Paul Horm, director de operaciones de la empresa IBM, quien en 2001 publica el libro *Autonomic Computing: IBM's Perspective on the State of Information Technology* [34]. La publicación se convirtió en el manifiesto de una nueva comunidad naciente. El documento muestra la preocupación de Horm por el aumento de la complejidad en los sistemas informáticos, la cual dificulta el desarrollo y administración

de estos. Los sistemas han evolucionado hacia la automatización de tareas repetitivas o no deseadas para un ser humano, pero alerta Paul, -*La evolución a través de la automatización también produce complejidad, como un producto inevitable*-. Para contrarrestar esta dificultad se propone un nuevo enfoque llamado Computación Autónoma la cual busca construir sistemas informáticos similares al sistema nervioso autónomo (SNA), con la premisa -*No lo pienses, no es necesario, lo tengo todo cubierto*- y tal como el SNA que mantiene en equilibrio a todas las funciones orgánicas tales como respirar y el latir del corazón; este paradigma se encargara de simplificar el desarrollo y la administración de los componentes informáticos.

En el manifiesto [34] se afirma que es tiempo para diseñar y construir sistemas informáticos capaces de ejecutarse a si mismos, ajustarse a diferentes circunstancias y preparar sus recursos para manejar de la manera más eficiente las cargas de trabajo que se les asigne. Se requiere de una infraestructura confiable que pueda proporcionar un rápido crecimiento del sistema y oculte su complejidad a los usuarios, clientes y proveedores. La computación autónoma es una visión holística que permitirá a toda la computación entregar mucha más automatización que la suma de sus partes autogestionadas individualmente.
En 2003, se lanzó el articulo *The vision of Autonomic Computing* [39], quien recogió las ideas de IBM[34] y aportó una arquitectura de software, que busca cumplir con las expectativas de Horm. Esta publicación impulsó el desarrollo de la Computación Autónoma, en los años venideros. En ella el autor muestra sin profundizar, una infraestructura distribuida y orientada al servicio, para apoyar a los elementos autónomos y sus interacciones. En la figura **2-1** se observa como los distintos elementos autónomos interactúan entre sí, logrando juntos los objetivos de alto nivel de administración. Según el artículo, la unión de estos logran un mejor desempeño en conjunto, que individualmente. El acronimo de las secciones Monitor (Monitor), Analyze (Análisis), Plan, Execute y Knowlegde; dan origen al nombre ciclo *MAPE-K*. En el 2005, IBM público el libro blanco de la Computación Autónoma[35]. En este libro se presenta el ciclo MAPE-K con las partes *Monitor*, *Analyze*, *Plan* y *Execute*.

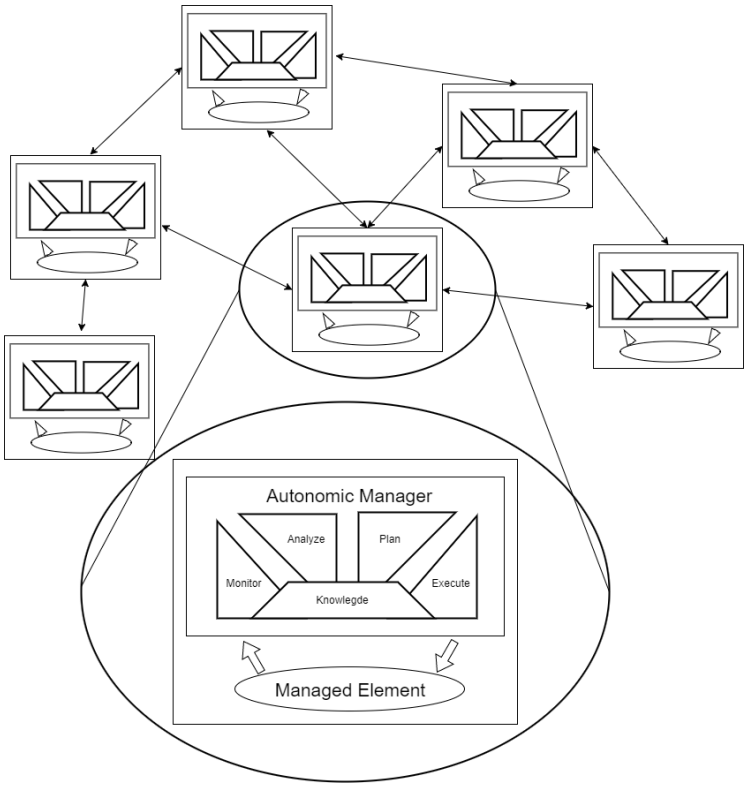

Figura 2-1.: Elementos autónomos interactuando con otros. Fuente: modificado de [39], figura 2.

En lo sucesivo las investigaciones buscaron resolver los desafíos planteados en estos libros. En este sentido se pueden mostrar los trabajos [13], [67], [54], [46], [17], [40], [6]. Otros trabajos buscaron usar la Computación Autónoma en otros campos de acción, como [3], [18]. Al interior de estos trabajos surgió la idea de usar el ciclo MAPE-K para realizar control automático, en este sentido encontramos los aportes [23], [31], [30], [61], [57], [38], [45].

3. Planteamiento del problema de investigación y su justificación

3.1. Descripción del área problemática

Durante los últimos años, el IoT se ha posicionado como un tópico de interés en la investigación en electrónica y sistemas computacionales. Empresas como Cisco [20], Microsoft [49], IBM [36] y Google [32], entre otros, han creado departamentos de investigación en el tema. Cada una de estas entidades ha presentado un aporte al estado del arte. Según Cisco, en el 2020 habrá 50,000 Millones de *Objetos de Internet* conectados [27]. Estos estarán enlazados a Aplicaciones Web, generando miles de nuevas oportunidades de negocio. Hoy en día, las principales aplicaciones del IoT se centran en la adquisición de datos, quienes una vez recogidos se analizan por medio de técnicas del campo de estudio *Big Data* para extraer información relevante para las empresas y entidades gubernamentales. Los datos usados en los procesos terminan alimentando artefactos, gobernados por estrategias de control automático.

En función de los recientes usos del Internet de las Cosas se han adoptado distintas arquitecturas y protocolos de comunicación. Un ejemplo de protocolo es el *Message Queue Telemetry Transport* [43], abreviado en *MQTT*. Igualmente se han desarrollado circuitos integrados y plataformas de prototipado para los sistemas embebidos IoT. El objetivo de estos adelantos es proveer herramientas de generación de nuevas empresas que desarrollen soluciones basadas en el IoT. Entre los circuitos integrados desarrollados para este tipo de trabajo se resalta el SoC *ESP8266* y su hermano el SoC *ESP32* [25]. En las plataformas de prototipado se puede mencionar a la *Raspberry Pi* [44] y en las Aplicaciones web, se puede referenciar a *ThingSpeak* de *MathWorks* [66].

Cuando un investigador y/o emprendedor desea realizar una solución basada en el Internet

de las Cosas puede optar por las herramientas antes descritas y un gran sin número de opciones más. Las Aplicaciones Web como *ThingSpeek* ofrecen diagramas típicos de conexión y Firmware para varias plataformas de prototipado. La primera dificultad que se presenta ante ese panorama es la no estandarización en los procesos y arquitecturas del Internet de las Cosas. Igualmente, las Aplicaciones Web operan de forma distintas entre ellas, generando incompatibilidades y sesgos en la apropiación de esta tecnología. Por otro lado, los desarrollos basados en estas aplicaciones y plataformas ofrecen una falsa sensación de tranquilidad al convencer a los desarrolladores que ya todo está resuelto. Muy pronto, estos mismos advierten la necesidad de un conocimiento avanzado en los temas de telemática, sistemas embebidos, desarrollo Web y otros campos de conocimientos especializados en Ingeniería, como el control automático. Día tras día, los proyectos de IoT aumentan en su complejidad causando más problemas a resolver por parte de los desarrolladores y administradores del sistema. Los retos son tan grandes y complejos que requieren de una gran experticia en múltiples disciplinas, la cual es muy difícil que posea una sola persona. Esto produce múltiples problemas que se hacen necesarios resolver. Uno de estos es la implementación y operación de estrategias de Control Automático para la automatización de procesos por medios de sistemas embebidos IoT.

Cuando se crea una red *Objetos de Internet*, se dice que se generó un entorno *Smart*. Actualmente muchos de estos entornos son utilizados por el público en general. Algunos de los entornos más conocidos son los *SmartPhone* y *SmartTv*, otros no tanto, pero muy importantes en investigación son las *SmartHouse*, *SmartGrid*, *SmartEnergy* y *SmartMetering*. Los entornos como las *SmartHouse*s y las *SmartGrid*, entre otros, usan procesos susceptibles de ser automatizados. En su mayoría estos contienen dinámicas complejas que requieren de tipos de estrategias de control automático desarrollados por expertos en el área.

Posteriormente, una vez propuesta la estrategia de control automático esta se debe sintonizar y verificar su correcto funcionamiento. Este trabajo requiere de mano de obra altamente calificada formada por ingenieros, físicos y matemáticos, en los casos más complejos. En las estimaciones de *Cisco* para el 2020 la cantidad de *Objetos de Internet* superan al número de humanos [27] y por tanto de ingenieros calificados en control automático y otros campos. La configuración de la estrategia de control usada depende enormemente de la situación particular de uso. Adicionalmente, las condiciones de operación con la que se da inicio al proyecto pueden cambiar y ocasionar malos funcionamientos durante el tiempo de la ejecución de la

solución. Este tipo de problemas son comunes y generan perdidas técnicas, económicas y financieras. Otro fenómeno derivado del escaso personal especializado, es la aplicación de controles automáticos pobres, insuficientes o con poco mantenimiento, lo cual aumenta las probabilidades de accidentes, daños y bajas en la productividad de las personas y empresas.

3.1.1. Pregunta de investigación

¿Cómo desarrollar un Firmware de control automático basado en el ciclo MAPE-K para Objetos de Internet, en el System On Chip ESP32?

3.1.2. Hipótesis de la investigación

El ciclo MAPE-K puede realizar control automático adaptativo en un sistema de Internet de las Cosas

3.2. Justificación

La elección y sintonización inicial de una estrategia de control automático, es un paso esencial para el éxito de cualquier control de procesos. En general estas configuraciones requieren de personal calificado en el área de estudio. Cuanto más complejo es un sistema, mayor es la necesidad de análisis y diseño de la estrategia de control, para disminuir la probabilidad de falla. Una avería por un inadecuado control automático genera pérdidas de tiempo, dinero y confiabilidad a investigadores y emprendedores.

Al contar con una alternativa que se encargue de la elección y sintonización de la estrategia de control automático se disminuyen los tiempos de diseño, desarrollo y lanzamiento de nuevos productos basados en IoT. Basándose en las predicciones de *Cisco* existirán más *Objetos de Internet* que personas, por lo cual, se requerirán de equipos enormes de desarrollo compuestos por profesionales multidisciplinarios. Estos grupos numerosos ejercen una carga económica enorme que pueden hacer muy costosas algunas aplicaciones del IoT.

Las aplicaciones del Internet de las Cosas cada día se integran más a la vida cotidiana de las personas y estas poco se detienen a pensar en cómo funcionan o cómo deben configurarlas para hacerlas funcionar. En la medida en que para los usuarios sea mayormente

transparente la implementación de las estrategias de control automático se mejora el grado de aceptación de las mismas, aumentando la probabilidad de supervivencia y crecimiento de las empresas basadas en el Internet de las Cosas.

El IoT como revolución tecnológica cambiará el mundo de maneras insospechadas. Esta cambiará la sociedad y la economía mundial generando nuevos focos de desarrollo, con lo cual se impulsará el crecimiento de los países involucrados. Colombia, en calidad de nación en vía de desarrollo, encuentra en el IoT una oportunidad única para mejorar sus condiciones de competitividad y bienestar. Las universidades y en particular sus investigadores están llamados a generar nuevos conocimientos, tales que impulsen el avance del país.

4. Referente Teórico

En este capítulo se estudiará las teorías y herramientas usadas en la realización del presente trabajo, los campos de estudio aquí presentados son muy extensos y exceden la capacidad de este informe final, por tal motivo, en cada caso solo se incluirán los elementos más relevantes para alcanzar los objetivos propuestos en el capítulo 5. Se iniciará explorando la noción de IoT, su arquitectura y sus interrelaciones. Con el propósito explorar claramente el campo del IoT se dará por sentado un conocimiento básico en el funcionamiento de Internet. Posteriormente se estudiará la teoría detrás del ciclo MAPE-K. A continuación, se revisará el concepto de control automático y en particular el área del Control Adaptativo. Finalmente, se abordarán las herramientas utilizadas en el desarrollo de aplicaciones Web.

4.1. Internet de las Cosas

El Internet de las Cosas es la siguiente evolución en las redes de comunicación basadas en la Web. La primer evolución Web permitió conectar personas entre sí (P2P). La segunda, conecto a personas con dispositivos electrónicos (P2M) y finalmente la tercera, permite conectar dispositivos electrónicos entre sí (M2M). Esta última se realiza sin intervención humana en el proceso [33]. Para lograr esta comunicación se utilizan sistemas embebidos basados en microprocesadores, capaces de interactuar con el mundo físico. Estos sistemas electrónicos se han integrado a los elementos cotidianos tales como televisores, teléfonos celulares, neveras y otros. A estos dispositivos con capacidad de comunicaciones *Máquina a Máquina* (M2M) por medio de la Web los designaremos como *Objeto de Internet*, *thing* en inglés. Al unirse estos *Objetos de Internet* obtenemos el Internet de las Cosas, cuyo acrónimo es IoT.

El IoT ha tenido un desarrollo impresionante en los últimos años a causa del sin fin de aplicaciones en la vida cotidiana. Entre estas utilidades encontramos el monitoreo de varia-

bles físicas, la realización de acciones específicas en sistemas y la generación de información útil [51]. Cuando se crea una interconexión de *Objetos de Internet* en el Internet de las Cosas se puede decir que se ha creado un ambiente *Smart*[33]. Entre los ambientes *Smart* más conocidos tenemos el *SmartPhone*y el *SmartTv*. Otros no tan conocidos son: *SmartHouse*, *SmartMetering*, *SmartEnergy* y más. El *SmartHouse* es el vínculo de las *Objetos de Internet* que integran un hogar como pueden ser: luces, grifos de agua, calefacciones, tomas de corriente, puertas, por nombrar algunos. El *SmartMetering* se refiere a la alianza de distintos elementos de medida como contadores eléctricos, de agua y gas entre otros. El *SmartEnergy* es el enlace de los elementos de producción y distribución de energía, quienes aún se pueden subdividir en las *SmartGrid*.

El IoT se divide en una serie de acciones descritas en [33], como *Networking*, *Computation*, *Storage* y *Visualization*. Cada acción se relaciona con enormes campos de la ingeniería y el conocimiento. Estas nuevas relaciones generan campos enteros de investigación. La acción de *Networking* involucra todos los sistemas de comunicación M2M, incluyendo cualquier medio de conexión a Internet como el *Wifi*, el *LTE*, *LoRa* y otros. La acción de *Computation* incluye los cálculos, algoritmos y harware que sean requeridos por la *Objeto de Internet*, parte del cómputo puede ocurrir en el mismo dispositivo o en la nube (*Cloud Computing*). Esta acción además incluye la interacción con el mundo físico por medio de sensores y actuadores. La acción de *Storage* almacena los datos más relevantes generados por las *Objetos de Internet* y al igual que el *Computing* puede usarse un servicio en la nube o guardarse en el mismo sistema. Finalmente la *Visualization* ocurre por diversos medios como pueden ser una página Web o una aplicación en un dispositivo móvil, como un *SmartPhone* tableta o en el mismo sistema embebido.

4.1.1. Networking

La principal característica de un *Objeto de Internet* es su capacidad de conectarse a otras *Objetos de Internet* por medio de Internet. Para esto, se requiere una infraestructura de comunicación digital que soporte la conexión entre ellas. El Internet es una red de redes que usa los protocolos *TCP/IP* para dar compatibilidad entre todas. Estos enmallados pueden tener su propia arquitectura y protocolos, más cuando se conectan al ciberespacio se debe respetar el protocolo *TCP/IP*. Las redes de computación se clasifican según se observa en

las tabla **4-1**. En esta tabla se hace especial énfasis en aquellas usadas en el IoT.

Tabla 4-1.: Clasificación de las redes de Internet. Fuente: propia.

Criterio	**Tipos**
Medio de Transmisión	Alámbrica e inalámbricas
Alcance	PAN, LAN, CAN, MAN y WAN
Relación funcional	Cliente - Servidor, Peer-to-peer
Tecnología	Punto a Punto, Difusión, Multipunto
Topología física	Estrella, Anillo, Bus, Malla, Árbol e Hibrida
Direccionalidad de datos	Simplex, Half-duplex y Full-duplex
Grado de autenticación	Privada y Acceso público
Grado de difusión	Intranet e Internet
Servicio o función	Comercial, Educativa y Proceso de datos

Los dos tipos de medios de transmisión se clasifica en alámbrico e inalámbrico. El alámbrico está caracterizado por usar medios guiados de electricidad como cables, en tanto, el inalámbrico usa el espectro electrómagnetico para transmitir sus mensajes. Según el alcance las redes son: *Personal Area Network* (PAN) es una red de alcance personal y no supera unos pocos metros. La *Local Area Network* (LAN) se limita a un espacio de utilización especifico, como un cuarto, piso, oficina o vehículo. El *Campus Area Network* (CAN) posee un alcance local limitado por un Área geográfica particular. El *Metropolitan Area Network* (MAN) amplía su cobertura hasta el nivel de un edificio o incluso una ciudad. La *Wide Area Network* (WAN) posee un alcance más allá de lo geográfico llegando incluso a nivel mundial. El criterio de relación funcional establece la forma como se comportan los distintos elementos de la red. En la relación Cliente - Servidor uno de los individuos opera como cliente solicitando una petición y un único elemento llamado Servidor contesta las peticiones de los clientes. La tecnología hace referencia a la emisión y recepción de los datos. Punto a punto implica un mensaje enviado de un único emisor a un único receptor.

Difusión o *Broadcast* involucra a un único emisor con múltiples receptores. Multi punto se compone de múltiples emisores enviando mensaje a múltiples receptores.

La interconexión física de los nodos de una red se llama topología. Las usadas en el Internet son la de Estrella, Anillo, Bus, Árbol e Hibrida. En la topología en estrella existe un único nodo que conecta a todos los demás nodos. Para la topología en anillo cada nodo se conecta a un nodo anterior y al siguiente, formando un lazo cerrado. En la topología en Bus todos los nodos se conectan a un canal común a los demás nodos. En la topología en malla cada nodo está conectado a todos los otros nodos. En el Árbol los nodos poseen una jerarquía entre ellos por medio de múltiples estrellas. Finalmente la híbrida o mixta es la combinación de cualquiera de las anteriores, en la Figura **4-1** se observan un descripción gráfica de estas topologías.

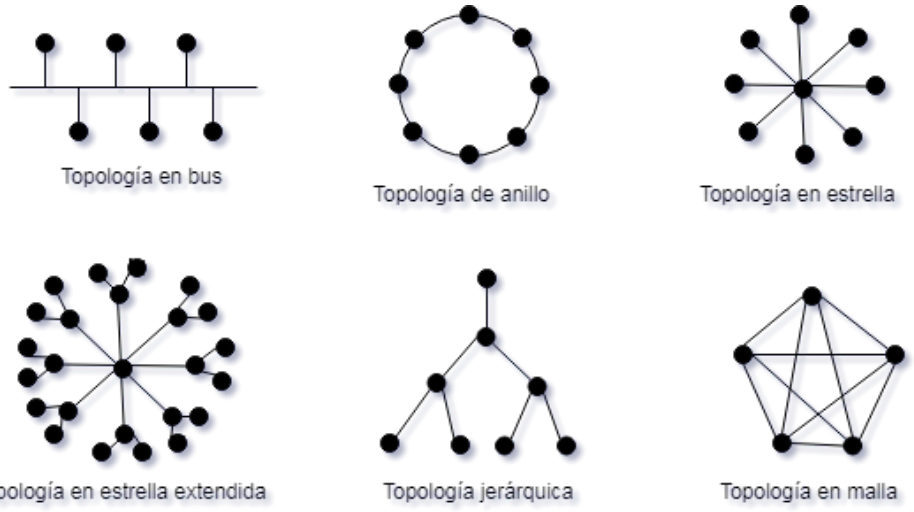

Figura 4-1.: Topologias de Red usadas en Internet. Fuente: tomado de [22]

La direccionalidad de los datos pueden ser Simplex, Half-duplex y Full-duplex. La Simplex o también llamada unidireccional se compone de un único emisor y un único receptor. En el Half- duplex ambos nodos pueden ser emisores o receptores, más esto no ocurre en simultaneo. Finalmente, en la Full- duplex ambos nodos son transmisores y destinatarios al mismo tiempo. En la figura **4-2** se observa el comportamiento de las direccionalidades entre dos nodos.

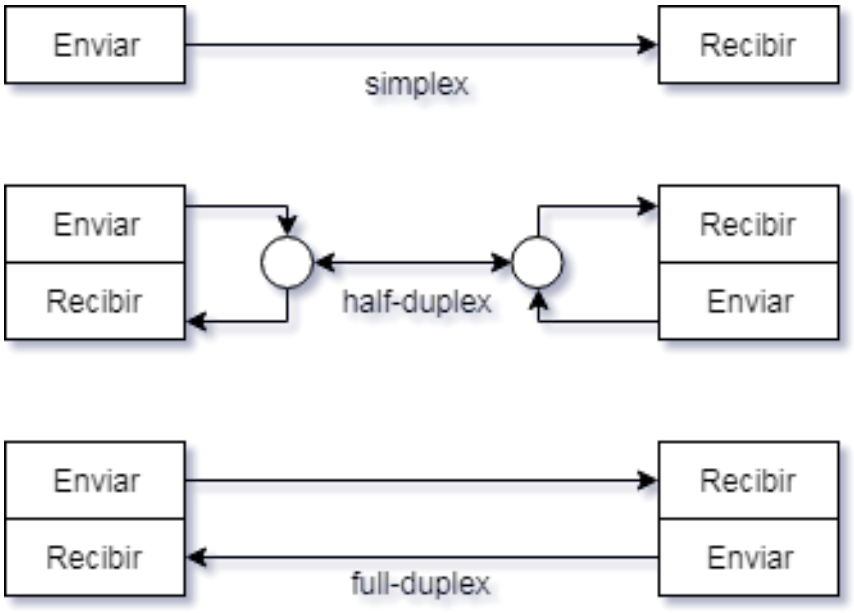

Figura 4-2.: Direccionalidad de los datos en una comunicación entre dos nodos. Fuente: tomado de [56]

El grado de autenticación administra el acceso a una red al restringirlo en función de su razón de ser. En ocasiones es normal contar con una red en la cual solo unos pocos nodos o terminales pueden acceder a ella y en otras ocasiones no hay restricción en la terminal de acceso, ya que sus funciones son públicas. El servicio o función de las redes define una finalidad de operación y su carácter. Entre las finalidades más comunes están la comercial, la educativa y más recientemente la de procesamiento de datos.

Para lograr que la Internet como la red WAN más grande del mundo funcione se requiere poner de acuerdo a fabricantes y usuarios en la forma de producir la comunicación. Los procesos que se llevan a cabo en Internet son tan complejos que abordarlos en su conjunto es inviable. Para disminuir la dificultad en la comprensión de la Internet se desarrolló el concepto de modelo de Red TCP/IP. Este modelo categoriza todos los procesos de la comunicación y permiten desarrollar tecnologías más específicas a cada categoría llamada capa. Los fabricantes y desarrolladores se pueden especializar en una capa del modelo para impulsar sus aportes al Internet. El modelo TCP/IP posee 4 capas llamadas *Enlace*, *Internet*, *Transporte* y *Aplicación*. En la figura **4-3** se observa las relaciones entre las capas.

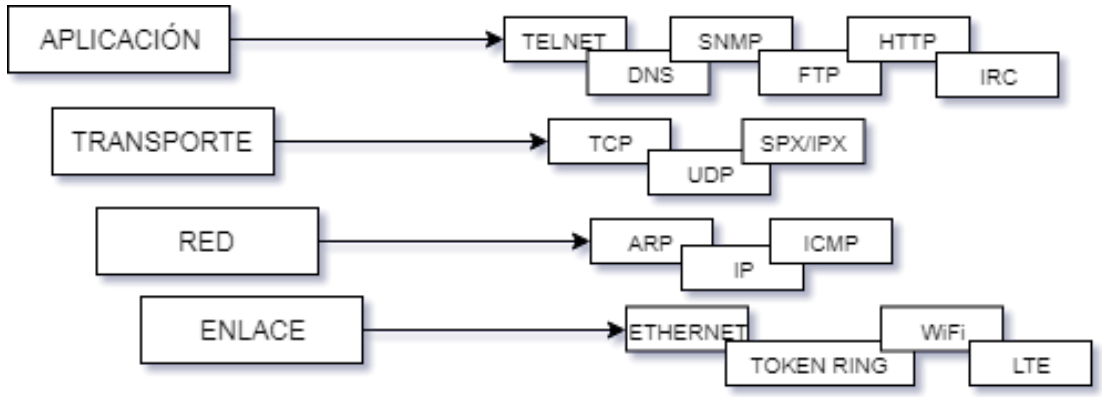

Figura 4-3.: Modelo TCP/IP implementado en Internet. Fuente: tomado de [28]

La capa de enlace provee la conexión a la red, está relacionada con los dispositivos físicos que permiten el enlace de datos. La capa de Red es la responsable de organizar los datos para su envió. En esta capa los datos se dividen en segmentos con información del lugar a enviar, el remitente y otros datos más. Este segmento de los datos y la información recibe el nombre de paquete de Internet. La capa de transporte se encarga como su nombre lo indica del transporte de los datos. Esta capa actúa de tal forma que permite a los programas comunicarse. La capa de aplicación es la encargada de usar los datos enviados o recibidos, es en esta capa que se desarrollan los servicios de Internet como http, ftp y otros.

Como se observó en el modelo TCP/IP la conexión física a la red pertenece a la capa de enlace y en esta se han desarrollado múltiples tecnologías de transmisión. Las más significativas entre las inalámbricas para el Internet de las Cosas son el WiFi y el LTE. El WiFi se desarrolló para la comunicación inalámbrica de equipos en una red LAN y opera bajo el estándar IEEE 802.11, con lo cual se garantiza la compatibilidad de los dispositivos desarrollados por distintos fabricantes. La necesidad de conexión a Internet se propagó a los teléfonos móviles, para los que se desarrollaron varias tecnologías entre las que se desataca el Long Term Evolution (LTE).

4.1.2. Computation

La acción de *Computation* (computación) en el IoT involucra al Hardware y Software usados para comunicarse y relacionarse con el entorno físico. Esta acción comparte elementos en común con la teoría de sistemas embebidos. El dispositivo principal en los sistemas

embebidos son los microprocesadores, los cuales se componen de una Unidad Central de Procesamiento (CPU), memoria digital y puertos de comunicación. Aquellos sistemas de cómputo que poseen un solo circuito integrado con la mayoría de los anteriores elementos se conocen como microcontrolador (μC). En la figura **4-4** se observa un esquema general de un μC.

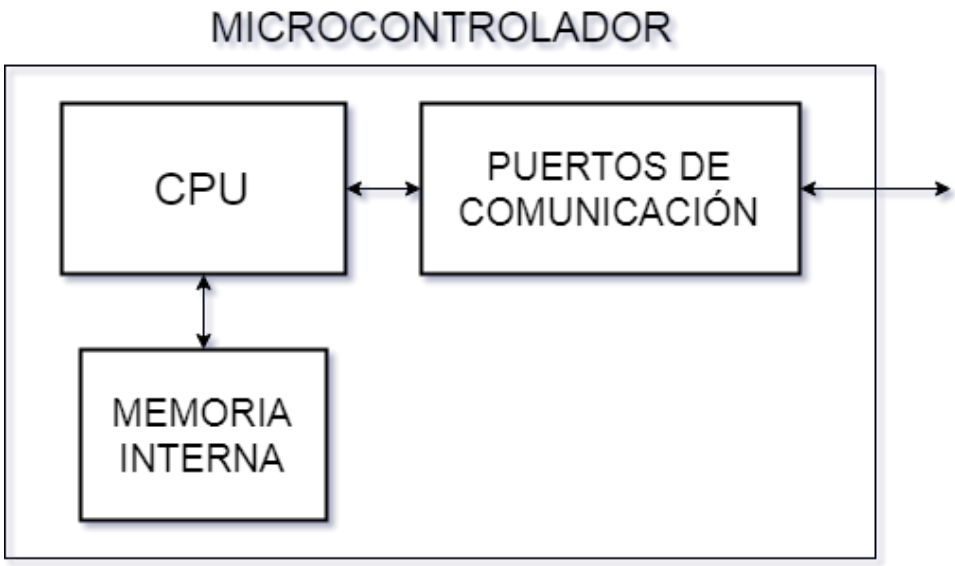

Figura 4-4.: Esquema general de un Micro Controlador. Fuente: propia.

En el mercado se encuentran microcontroladores (μC) de diferentes fabricantes y características. En la tabla **4-2** se observa las clasificaciones de los principales (μC). Las dos principales arquitecturas usadas en el desarrollo de un sistema embebido son la *Von Neumann* y la *Harvard*, su principal diferencia se encuentra en la organización de su memoria interna. En los últimos años la arquitectura *Harvard* ha dominado el mercado. Las Unidades Centrales de Procesamiento (CPU) se ha construido por tamaños de palabra de 8, 16, 32 y 64 bits. Cuanto mayor es el número de bits en el procesador, mayor es la capacidad de cómputo y más complejos los algoritmos que este es capaz de ejecutar. La memoria interna se ha divido en Memoria de Acceso Aleatoria (RAM) y Memoria de Solo Lectura (ROM). La ROM podemos dividirla en *Flash* y *EEPROM* según su forma de almacenar sus datos. Por lo general los programas se almacenan en la *Flash*.

Tabla 4-2.: Clasificación de Microcontroladores. Fuente: propia.

Criterio	Tipos
Arquitectura	Von Neumann y Harvard
CPU	8 bits, 16bits, 32bits y 64bits
Memoria flash	1kB, 2kB, 4kB, 8kB, 16kB, 32kB, 64kB, 128kB, 256kB, 512kB, 1MB y superior
Memoria RAM	128bits, 256bits, 512bits, 1kB, 2kB, 4kB, 8kB y superior
Memoria EEPROM	1kB, 2kB, 4kB, 8kB, 16kB, 32kB, 64kB, 128kB, 256kB, 512kB, 1MB y superior
Conversor Analógico / Digital	8bits y 10bits
Puertos	GPIO, UART, SPI, I^2C, USB, Ethernet, Can y otros
Modulador de Ancho de Pulso	8bits y 10bits
Fabricante	Atmel, Freescale, Intel, National Semiconductor, Microchip, NXP, Texas Instruments, Zilog y otros

La comunicación del microcontrolador con el exterior es de vital importancia y por tal motivo se han desarrollado múltiples tecnologías para tal fin. Las principales comunicaciones se realizan en formato digital, otras comunicaciones se realizan de forma analógica. Entre las digitales encontramos los puertos de propósito general (GPIO), Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I^2C), Universal Serial Bus (USB), Ethernet, can y otros. En la interfaz analógica se emplean los Conversores Analógico - Digital (ADC), los Conversores Digital - Analógico (DAC) y los puertos de Modulación por Ancho de Pulso (PWM).

Se han desarrollado microcontroladores y sistemas embebidos encapsulados en un solo circuito integrado llamados *System on Chip* (SoC). Estos μC poseen puertos de comunicación

compatibles con Internet, como pueden ser Ethernet, WiFi, Bluetooth, LoRa y otros. Múltiples fabricantes han desarrollado sistemas embebidos para IoT y para el caso particular de la presente tesis se estudiarán los presentados en la tabla **4-3**.

Tabla 4-3.: Características de sistemas embebidos para el Internet de las Cosas, usados en la presente tesis. Fuente: propia.

Sistema	**ESP 8266**	**ESP32**
Microprocesador	Tensilica L106 32-bit RISC processor	Xtensa single-/dual-core 32-bit LX6 microprocessor
CPU	32bits	2 x 32bits
Memoria flash	512kB y superior	512kB y superior
Memoria RAM	50 kB	520 kB
Conversor Digital / Análogo	0	2 x 8bits
Conversor Análogo / Digital	1x 10bits	18 x 12bits
Puertos	GPIO, UART, SPI, I^2C, USB, Ethernet, Can y otros	GPIO, 4 x SPI, 2 x I^2C, 3x UART, CAN y otros
Modulador de Ancho de Pulso	10 x 10bits	16 x 24bits
Puerto para IoT	WiFi	WiFi, Bluetooth y Ethernet
Fabricante	Espressif	Espressif

En el área del software se han usado lenguajes de programación como C, C++ y Lua por mencionar algunos. Igualmente, se han lanzado múltiples Entornos de Desarrollo Integrados (IDEs). *Arduino* es un IDE y plataforma libre de prototipado para sistemas embebidos [68], basado en una biblioteca basada en C++. *Arduino* ha logrado unificar la sintaxis de muchos proyectos de Internet de las Cosas. Adicionalmente, los fabricantes de microcontroladores y otras plataformas proveen de soporte para trabajar con esta tecnología.

4.1.3. Storage

La acción *Storage* (almacenamiento) involucra los datos generados por el sistema IoT y la forma como se guardan. Los datos son fundamentales para el desarrollo de los proyectos de Internet de las Cosas. Gran parte de esta información se utiliza para realizar análisis de tendencia y otros procesos más complejos como el Big Data. El almacenamiento de estas cifras se realiza en servidores en la nube o en el mismo sistema embebido. Para el ciberespacio se utiliza comúnmente las bases de datos relacionales. A estas bases se acceden por medio de lenguajes de consulta, como el *Structured Query Languaje* (SQL). A nivel mundial se presentan múltiples proveedores de bases de datos y estos realizan sus propias variaciones a SQL, motivo por el cual se representa la estructura de la información allí contenida por medio diagramas y otros tipos de modelos, independientes de la versión de SQL soportada. Una de estas herramientas es el diagrama de relación - entidad, en el que se representa gráficamente los datos guardados y sus tipos. En cuanto a las relaciones indicadas por el esquema, las principales son 1:1 donde un solo dato se relaciona con otro dato único, 1:n donde un dato se relaciona con múltiples datos y n:m donde varios datos se relaciona con otros datos en distintas cantidades. En la figura **4-5** se observa el ejemplo de un blog con usuarios, post y una relación 1:n.

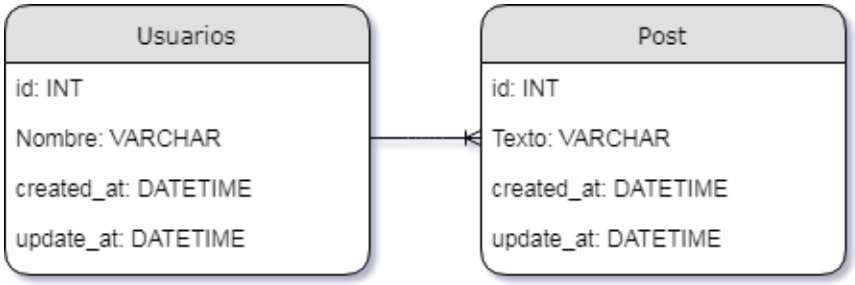

Figura 4-5.: Ejemplo de un diagrama relación - entidad. Fuente: propia.

4.1.4. Visualization

La acción *Visualization* (Visualización) incluye los elementos que permiten conocer el estado interno de un *Objeto de Internet* por lo general se realiza en el mismo sistema embebido o en otros dispositivos conectados a una plataforma Web de Internet de las Cosas. Si la visualización se realiza al interior del sistema embebido se pueden emplear *leds*, *displays* de 7 segmentos, Pantallas de Cristal Líquido (LCD) o pantallas de Leds Orgánicos (OLEDs), en-

tre otros. En la plataforma Web se emplean páginas de internet desarrolladas por medio del Lenguaje de marcas de hiper texto (HTML), Hojas de Estilo en Cascada (CSS) y lenguajes de programación como el Preprocesador de Hipertext (PHP) y JavaScript. Las plataformas Web cumplen una doble función en las acciones *Networking* y en la *Visualization*, motivo que las convierten en elementos complejos de diseñar y desarrollar. Algunas empresas han desarrollado sus propias plataformas web, entre las más famosas estan ThingSpeak, Blynk. Su diseño está orientado a tipos genéricos de proyectos de Internet de las Cosasen los cuales se reciben, muestran y en algunos casos almacenan datos. Algunas de estas plataformas permiten el envío de ordenes sencillas desde el usuario hasta el sistema embebido.

Una aplicación Web es un programa ubicado en un servidor al que los usuarios pueden acceder para interactuar con él. A diferencia de las paginas Web estáticas estas cuentan con la capacidad de guardar información y tomar decisiones en función de un algoritmo previamente implementado. Para realizar una Aplicación Web se requiere de usar herramientas de desarrollo. Entre las principales encontramos el lenguaje de marcas HTML, los lenguajes de programación PHP, JavaScript, Python, un motor de base de datos y otros. El manejo en conjunto de todos estos elementos puede ser una labor desafiante, se han creado aplicaciones llamadas *frameworks* con el objetivo de facilitar esta tarea.

Laravel es un *framework* de desarrollo Web, de código abierto. Desarrollado por Taylor Otwell en 2011, para el lenguaje de programación PHP. Su elaboración se baso en las herramientas *Ruby On Rails* y *Symfony*. Es utilizado con el objetivo de facilitar el trabajo en áreas comunes como autenticación, enrutamiento, gestión de secciones (de usuario) y almacenamiento en caché. [63]. Entre sus características más relevante están:

Diseño: Originalmente basado en el Modelo Vista Controlador (MVC), ha creado su propia arquitectura, la cual facilita su utilización. En la figura **4-6** se observa su organización. Esta estructura permite una correcta separación y modularización del código.

Elonqent: Sistema de mapeo de datos relacional (ORM), el cual convierte la información entre un lenguaje de programación orientado a objetos y una base de datos relacional como motor de persistencia.

Base de datos: Gestión y manipulación de tablas desde el código. Mantiene el control de versiones mediante el sistema de migraciones.

Pantillas Blade: Sistema de plantillas para la elaboración de las vistas, el cual emplea una memoria cache con el objetivo de mejorar su velocidad. Facilita la creación de las vistas mediante el uso de layouts, herencia y secciones.

Bibliotecas externas Laravel extiende su funcionalidad al permitir el uso de paquetes externos, creados por terceros. Estos facilitan la programación de las aplicaciones web y disminuye el tiempo del desarrollo.

Artisan La mayor fortaleza de Laravel esta en su capacidad de producir automáticamente componentes de código, estos han pasado por un proceso de depuración y son bastante estables. Entre los módulos a crear se encuentran aquellos que facilitan el manejo de las bases de datos, la gestión de rutas, las tareas programadas y el uso de cachés y colas. El desarrollador accede a la administración por medio de comandos en consola e interpretados por el programa Artisan.

La arquitectura de las aplicaciones Web realizadas en Laravel se puede observar en la figura **4-6**. Para describir su funcionamiento se utilizará un caso, donde un usuario solicitará cierta información a la aplicación Web.

El usuario ingresa en el navegador para realizar una petición a la aplicación Web, indicando ruta o URL donde se encuentra dato a buscar. El servidor inicia el programa al ingresar la página *public/index.php*, esta verifica la existencia de la ruta por medio de *routes.php*. Si la URL es válida, se procede a comprobar que se cumplan las políticas de privacidad. En caso de cumplirse los requisitos de acceso la aplicación llama al *Controlador* encargado de preparar dicha información, en él se realizan las consultas necesarias usando al *Modelo* para acceder a la base de datos. Cuando los estos están listos para ser usados se entregan a la *Vista*, quien la presenta empleando una página Web facilitando la comprensión del cibernauta.

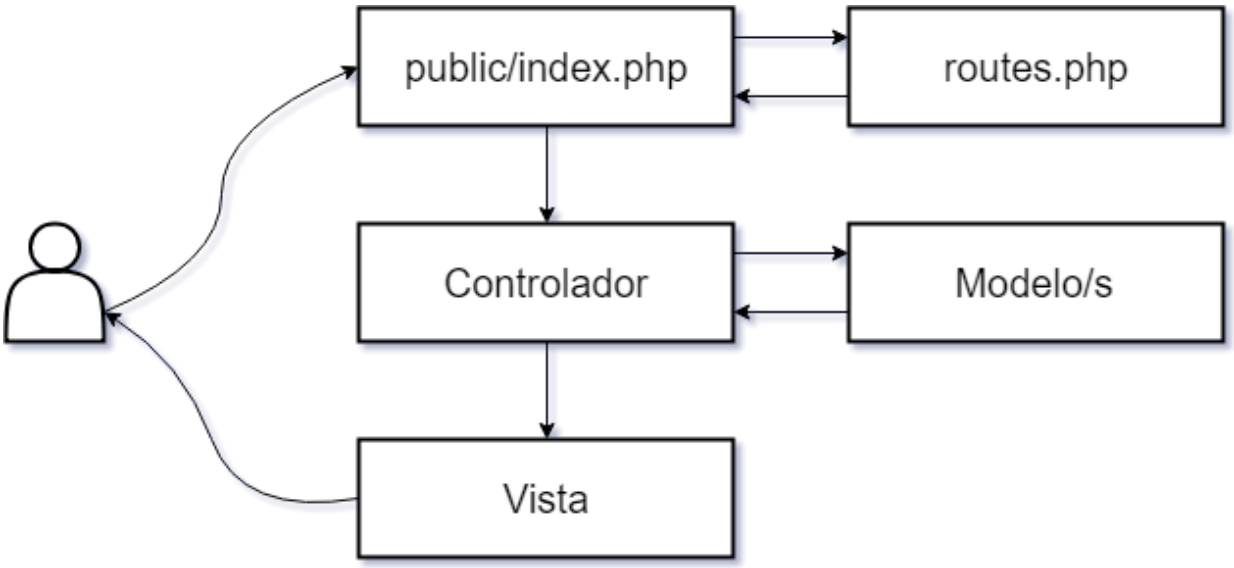

Figura 4-6.: Arquitectura de los proyectos desarrollados en Laravel 5. Fuente: tomado de [63].

4.2. MAPE-K

La búsqueda por desarrollar unos *Objetos de Internet* autónomos pasa por el desarrollo de software que les proporcione tal capacidad. Son muchos los paradigmas de autonomía presentes en el campo de la informática. Entre ellos se pueden nombrar a las máquinas de aprendizaje, los sistemas basados en agentes o multiagentes, la computación orgánica y autónoma. El presente trabajo se centrará en el caso del ciclo MAPE-K, proveniente de la Computación Autónoma.

En el capítulo 2 se presentó el origen del ciclo *MAPE-K*, a partir del trabajo de la empresa IBM [35]. Este paradigma plantea que cada tarea, módulo o funcionalidad implementada en un sistema informático debe ser administrado independiente y autónomamente por otro programa, tal como se aprecia en las figura **4-7**. Esta administración del sistema provee capacidades nuevas por la interacción de las distintas tareas y sus otras contrapartes, tal como se observa en la figura **2-1**. En la Computación Autónoma se llama *elemento autónomo* a la simbiosis de módulo y administración. Las principales características de estos sistemas autónomos se presentan en las tablas **4-4** y **4-5**. En la tabla **4-6** se presenta una descripción de las partes del *elemento autónomo*.

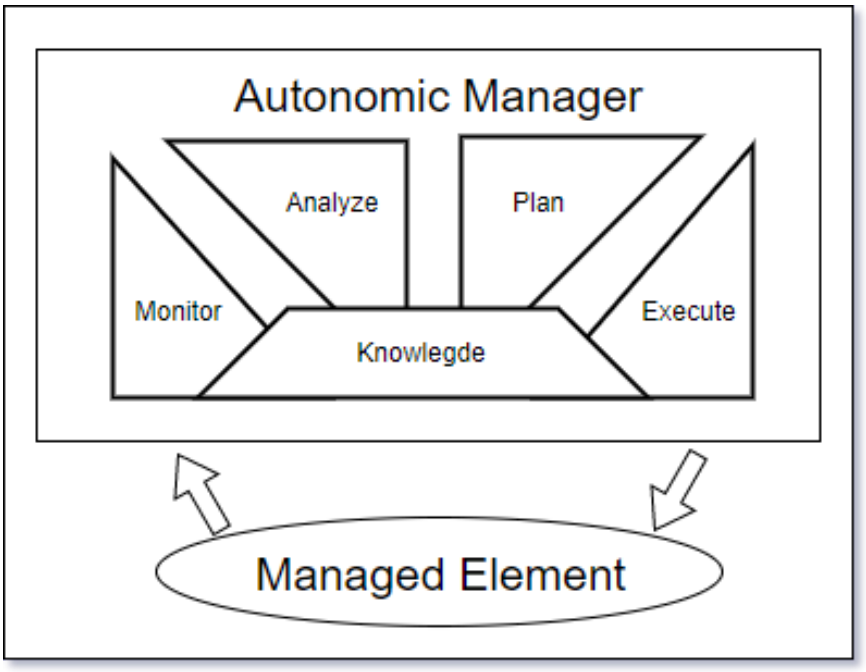

Figura 4-7.: Estructura de un elemento autónomo. Fuente: modificado de [39].

Tabla 4-4.: Características de un sistema informático autónomo. Primera parte. Fuente: adaptada de [34].

Característica	**Descripción**
Conocerse a si mismo	Para ser autónomo un sistema informático necesita conocerse a si mismo y comprender que sus componentes poseen una identidad. Requiere conocer el alcance de sus propios recursos, los que puede tomar prestado, los que pueden compartir y los que se deben aislar.
Configurarse y reconfigurarse	La configuración del sistema debe realizarse automáticamente, al igual que los ajustes dinámicos para lograr mejorar el rendimiento ante entornos cambiantes. Es posible que en la configuración automática un sistema deba crear varias imágenes de su software crítico y reasignar sus recursos cuando sean necesario. Algoritmos adaptativos correrán en tales sistemas y deberán aprender las mejores configuraciones que alcancen los niveles de rendimiento obligatorios.

Tabla 4-5.: Características de un sistema informático autónomo. Segunda parte. Fuente: adaptada de [34].

Característica	**Descripción**
Optimizar su funcionamiento	Un sistema de computación autónoma nunca se conforma con el status quo, siempre busca formas de optimizar su funcionamiento. Él supervisa sus partes constituyentes y ajusta el flujo de trabajo para lograr sus objetivos predeterminados. El esfuerzo constante por optimizarse es la única forma en que un sistema informático podrá satisfacer las complejas y frecuentes demandas de los usuarios. La auto optimización también es la clave para habilitar la disponibilidad ubicua de los servicios. Se necesitan mecanismos avanzados de control realimentado para monitorear sus métricas y tomar las decisiones adecuadas, igualmente, se requieren nuevos enfoques para aplicar la teoría de control en los sistemas computacionales. Las innovaciones en el control deben ocurrir a la par con los nuevos enfoques de arquitectura general de los sistemas. Los componentes de un sistema autónomo, sin importar cuan diversos sean, deben ser controlables de manera unificada.
Curarse a si mismo	Un sistema autónomo debe realizar algo similar a la recuperación: debe poder reponerse de eventos rutinarios o extraordinarios que podrían causar que algunas de sus partes funcionaran mal. Debe descubrir problemas potenciales y luego encontrar una forma alternativa de usar sus recursos o reconfigurar el sistema para que funcione según lo esperado. En lugar de aumentar las piezas de reabastecimiento, como se hace actualmente en la recuperación en un sistema informático, este debe poner en acción elementos redundantes o subutilizados para que actúen como piezas de reemplazo, en cierto sentido, similar a lo que hace nuestro cerebro cuando se dañan partes de él. La identificación de las causas de falla requiere de un análisis de la causa raíz, un intento por examinar sistemáticamente, *que hizo qué a quién*, identificando el origen del problema. Un enfoque orientado a la acción debe tener prioridad en una solución autónoma, a medida que la inteligencia artificial se incorpore, estos aprenderán a repararse por si mismos.

Tabla 4-6.: Partes de un elemento autónomo. Fuente: adaptada de [39]

Elemento	**Descripción**
Managed Element	El elemento gestionado es el equivalente a un sistema no autónomo. Este puede ser un recurso de Hardware como una memoria, una CPU, una impresora o un recurso de Software como una base de datos, un servicio o un sistema heredado.
Autonomic Manager	El gestor autónomo monitorea al elemento gestionado y su entorno, analiza esta información, construye planes que ejecuta en función de los objetivos de alto nivel.

Tabla 4-7.: Partes del ciclo MAPE-K. Fuente: adaptada de [35]

Elemento	**Descripción**
Monitor	Mecanismo para recopilar, agregar y filtrar información. Se usan métricas y topologías.
Analyze	Correlaciona y modela las situaciones complejas, usando técnicas como: series de tiempo, modelos de espera entre otros. Este mecanismo aprende sobre el entorno y predice situaciones futuras.
Plan	Mecanismo para construir las acciones necesarias para alcanzar las metas y objetivos. Esta parte del ciclo usa información de políticas para guiar su trabajo.
Execute	Proporciona los mecanismos que controlan la ejecución de un plan con las consideraciones para la actualización dinámica.

En la tabla **4-7** se detalla sus componentes. Igualmente, el libro resume la autogestión en la tabla **4-8**.

Tabla 4-8.: Aspectos de la autogestión en la Computación Autónoma. Fuente: modificado de [39]

Concepto	**Informatica actual**	**Computación Autónoma**
Auto - Configuración	Centros de datos con múltiples proveedores y plataformas, las cuales requieren ser configuradas manualmente.	Configuración automatizada de componentes siguiendo políticas de alto nivel. El resto del sistema se ajusta automáticamente.
Auto - Optimización	Cientos de parámetros ajuste no lineales cambiados manualmente. El rendimiento y numero de parámetros cambia por cada versión nueva.	Los sistemas buscan continuamente oportunidades para mejorar su propia eficiencia.
Auto - Curación	La determinación de problemas en los sistemas grandes y complejos pueden llevar semanas a equipos de programadores	El sistema automáticamente detecta, diagnostica y localiza problemas en el Software y Hardware.
Auto - Protección	La detección de ataques y fallas en cascada es manual.	El sistema se defiende automáticamente contra ataques o fallas en cascada. Usa alertas tempranas para anticipar y prevenir fallas en todo el sistema.

4.3. Control Automático

El control automático es la rama de la ingeniería que busca garantizar que los sistemas cumplan su cometido, desde las primeras máquinas de vapor hasta el transbordador espacial. El control automático ha sufrido múltiples evoluciones a lo largo de su historia. Estas inician en el desarrollo de mecanismos gobernados por complejas teorías matemáticas. Continúan,

gracias al avance de la electrónica, con la implementación de circuitos eléctricos analógicos y posteriormente, con la aparición del computador, se desarrollan equipos de cómputo para control digital.

Dado que el objetivo principal del control automático es lograr el cumplimiento de los objetivos planteados [41], se ha desarrollado una teoría que la organice. En esta, un sistema de control se constituye de objetivos, componentes del sistema de control y resultados o salidas. En la figura **4-8** se observa cómo se interrelacionan estos elementos.

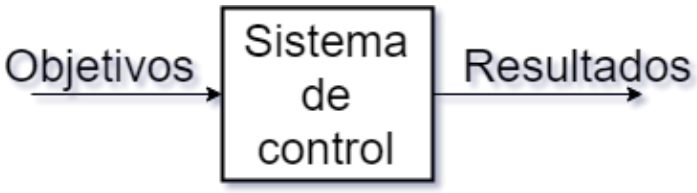

Figura 4-8.: Esquema de un sistema de control. Fuente: tomado de [41]

En el desarrollo del control automático, las matemáticas han ocupado un lugar preferencial. Las principales herramientas usadas son la *transformada de Laplace*, la *transformada Z*, el *álgebra lineal* y el *análisis matemático*, entre otros. Con ayuda de la física, la química y otras ciencias; se posibilita modelar el comportamiento de casi cualquier máquina y/o proceso y a partir de esta abstracción matemática, se busca diseñar un sistema de control que garantice el desempeño deseado para la máquina y/o proceso ante distintas circunstancias de operación no deseadas. Generalmente, este sistema de control (controlador) se conecta a la entrada de la máquina y/o proceso, los objetivos del sistema de control se modelan por medio de una entrada de referencia, la cual puede ser estática o dinámica. En la figura **4-9** se observa la conexión entre el controlador y el proceso controlado.

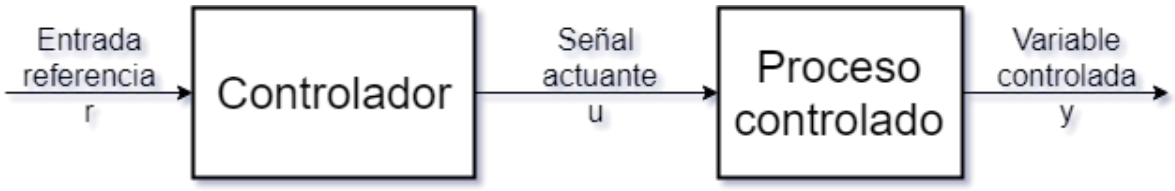

Figura 4-9.: Esquema de conexión entre el controlador y el sistema controlado. Fuente: tomado de [41]

La principal estrategia dentro de la teoría de control es la retroalimentación, tal y como se

muestra en la figura **4-10**. La estrategia se compone de un lazo cerrado de relaciones causa - efecto entre la maquina y/o proceso a controlar, la variable controlada (y), un sensor para monitorear la salida (y) una comparación (+/-) de la salida observada en la planta con el comportamiento esperado (r), un sistema de control automático (Controlador) y finalmente, una perturbación que modela las circunstancias de operación no deseadas.

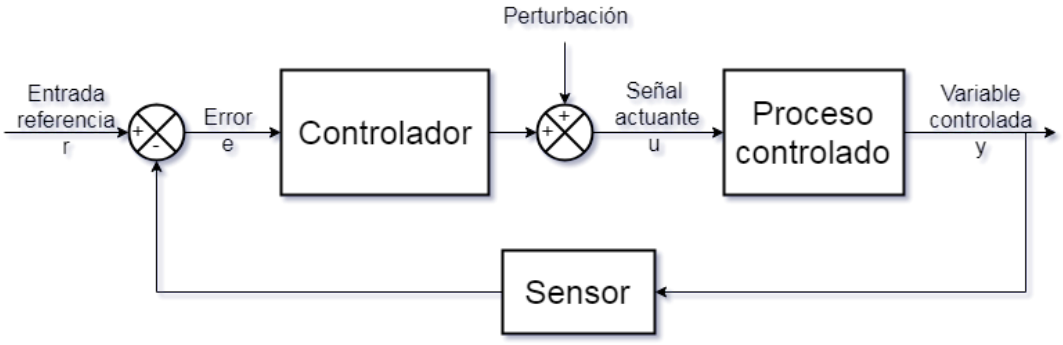

Figura 4-10.: Esquema de un sistema controlado por medio de un lazo cerrado. Fuente: tomado de [41]

4.3.1. Tipos de sistemas de control

Los sistemas de control se catalogan según su método de análisis, el tipo de señal y su propósito principal. Los métodos de análisis se pueden separar en lineales o no lineales, variantes e invariantes en el tiempo. Los tipos de señales se separan en tiempo continuo o discreto y moduladas o no moduladas. El propósito principal de un sistema de control busca el establecimiento de la posición, o la velocidad de la señal de salida [41].

Sistemas no lineales: En la mayoría de los casos todos los sistemas son no lineales y se modelan como lineales por su facilidad del análisis y diseño. Los sistemas en ciertos intervalos exhiben características lineales por principio de superposición y no existen métodos generales para resolver una amplia clase de sistemas.

Sistemas Lineales Invariantes en el Tiempo: Los sistemas Lineales Invariantes en el Tiempo (LTI) poseen parámetros estacionarios, es decir contienen elementos que se no degradan o cambian en el tiempo.

Sistemas de control en tiempo continuo: El control se produce por medio de señales continuas en función del tiempo, es decir sin discontinuidades o cambios bruscos.

Sistema de control en tiempo discreto: El control se compone de señales en forma de pulsos o en código digital. Este se dividen en control muestreados y digital. En la figura **4-11** se observa un esquema de control muestreado. El control digital, plantea múltiples ventajas, al trabajar en instantes de tiempos. Lo que posibilita el uso de un sistema de computo que realice multiples calculos en varios canales al mismo tiempo. De igual manera, los cambios en los sistemas de control son más fáciles de implementar y es menos sensible a los ruidos.

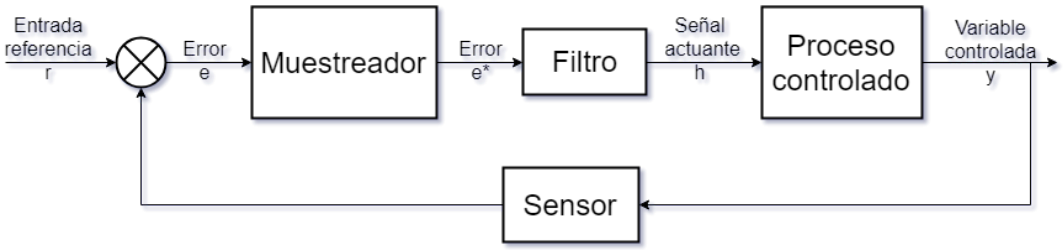

Figura 4-11.: Esquema de un sistema de control muestreado. Fuente: tomado de [41]

4.3.2. Diseño del sistema de control

Para realizar un adecuado control se debe conocer con la mayor precisión posible el comportamiento del sistema a controlar ante diferentes entradas. Este conocimiento se puede obtener mediante la experimentación, la cual es una tarea engorrosa y difícil; o por medio del modelado matemático de sus leyes físicas, químicas u otras. Para facilitar el diseño del sistema de control se emplean herramientas analíticas que permita un diseño razonablemente predictivo y confiable, sin necesidad de realizar extensas simulaciones, que verifiquen su éxito. El diseño del controlador se divide principalmente en teoría clásica y moderna. El control clásico emplea las herramientas matemáticas de variable compleja, ecuaciones diferenciales y en diferencias, transformadas de *Laplace* y *Z*. El control moderno usa la teoría de matrices, de conjuntos, el álgebra y transformación lineal, el calculo variacional, programación matemática y la teoría de probabilidad [41]. En sistemas que no se conoce con precisión su comportamiento se aplica el control adaptativo.

4.3.3. Control Adaptativo

En los sistemas controlados la realimentación se utiliza para compensar los faltantes o inexactitudes de los modelos de las plantas. Si se conociera completamente el comportamiento del sistema y además este, no contara con perturbaciones el control realimentado no se requeriría. En casos donde es pobre el modelo o se cuentan con fuentes de perturbaciones es necesario contar con control realimentado. En la figura **4-10** se observa la perturbación a la entrada del proceso a controlar o planta, esta perturbación tambien puede ocurrir a la salida, tal y como se observa en la figura **4-12**.

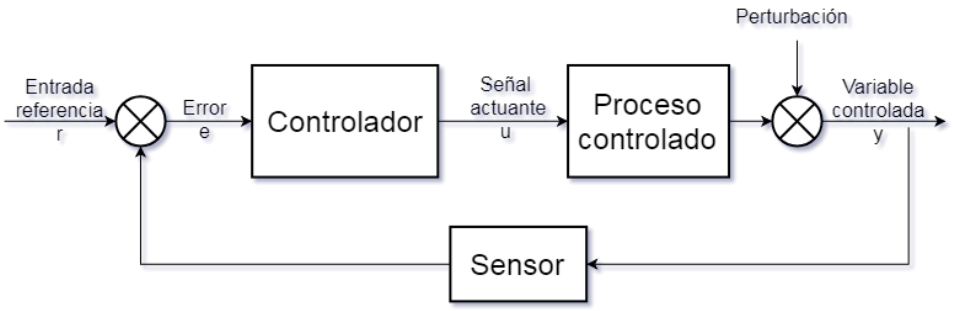

Figura 4-12.: Esquema de controlado por medio de realimentación con perturbación. Fuente: tomado de [60]

En la figura **4-13** se presenta el lazo cerrado adaptativo. En este nuevo esquema se observan los siguientes elementos:

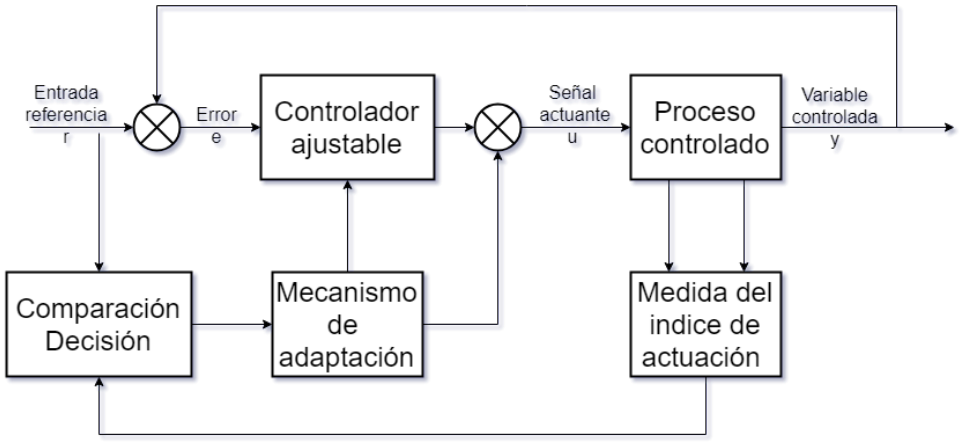

Figura 4-13.: Configuración básica de control adaptativo. Fuente: tomado de [60]

Controlador ajustable: La configuración del controlador puede variar durante la ejecución del lazo del control. Esté es manipulado por un elemento externo.

Comparación de decisión: Para determinar la nueva configuración del controlador se compara su desempeño contra el estado deseado, representado en la entrada de referencia r. En caso de ser pobre o insuficiente se procede a cambiar su configuración.

Mecanismo de adaptación: La forma como se cambia la configuración del controlador es por medio del sistema *Mecanismo de adaptación* y este es particular para cada uno.

Medidad del indice de actuación: El indice de actuación es un valor escalar que representa el grado de efectividad del controlador en el proceso controlado. Su medida es particular a cada proceso controlado.

A continuación, se presenta un pequeño listado de las teorías de control, esté no pretende ser un compendio de todas las teorías de control, solo busca mostrar la gran cantidad de opciones presentes a la hora de diseñar un control.

- Control Proporcional (P)
- Control Derivativo (D)
- Control Integral (I)
- Control Proporcional, Derivativo e Integral (PID)
- Control por adelanto y atraso de fase
- Control por retroalimentación de estados
- Control Difuso
- Control por redes neuronales
- Control Optimo
- Control Adaptativo
- Control Robusto

4.3.4. Control PID

El control PID es un tipo de control que tiene en cuenta el error, la integral del error y la derivada del error. En la figura **4-14** se observa el diagrama de bloques del control.

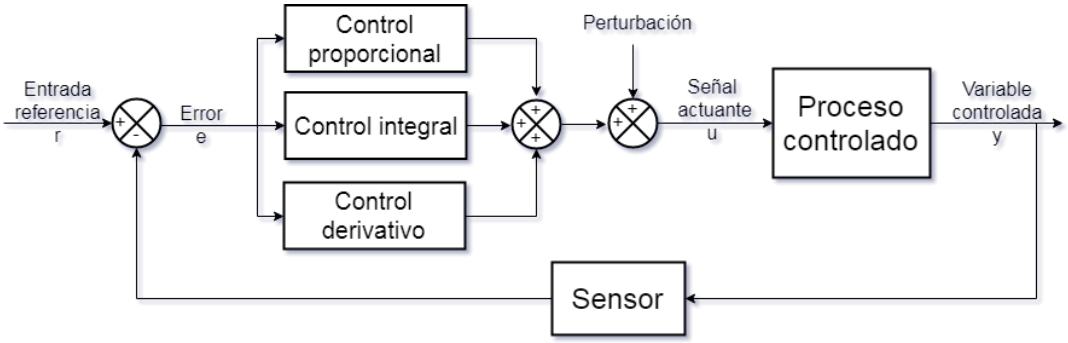

Figura 4-14.: Esquema de conexión entre el controlador y el sistema controlado. Fuente: modificado de [41]

5. Objetivos

5.1. Objetivo general

Desarrollar un Control automático adaptativo por medio del ciclo MAPE-K, en el Internet de las Cosas, usando el System On Chip *ESP32*

5.2. Objetivos específicos

- Desarrollar un *Objeto de Internet* para realizar control automático, usando el System On Chip *ESP32*.
- Desarrollar una estrategia de control automático adaptativo por medio del ciclo MAPE-K.
- Probar el funcionamiento del Control automático adaptativo por medio del ciclo MAPE-K, en el Internet de las Cosas, usando el System On Chip *ESP32*.

6. Metodología

El presente trabajo de grado se realizó en cuatro fases llamadas: Conceptual, Planificación, Empírica e Interpretativa. En la fase Conceptual se formuló la pregunta de investigación y se refinó constantemente por medio de una reflexión concienzuda, por más de un año. Durante este tiempo se recopiló material bibliográfico de alta fiabilidad, usando técnicas de búsqueda crítica. Estas cualidades se garantizaron por el uso de fuentes de información primarias y secundarias en bases de datos indexadas por Colciencias y otros entes internacionales oficialmente reconocidos. Posteriormente se estudiaron los sistemas embebidos usados por el Internet de las Cosas. Se analizaron sus características en función de las necesidades del proyecto. Se realizó una comparación entre ellos y se eligió a uno, empleando una tabla de toma de decisión. De las múltiples opciones del mercado se seleccionó el SoC *ESP32* de la empresa *Espresssif*. El ESP32 posee dos procesadores de 32bits, conexiones WiFi y BlueTooth, adicionalmente cuenta con 18 entradas ADC y 2 salidas DAC, necesarias para ejercer control automático.

Posteriormente, en la fase de Planificación se planteó la hipótesis de investigación y los objetivos principal y secundarios, enumerados en el capítulo 5. A continuación, se propuso el diseño metodológico del presente trabajo, descrito más abajo. En la fase Empírica se desarrollaron los prototipos de dispositivos electrónicos y software embebido y en la nube. A estos, se le sumaron el desarrollo de sistemas LTI de prueba. Ulteriormente, se diseñó un conjunto de pruebas para verificar el alcance de los objetivos, cuyos resultados se muestran en el capítulo 7. En la culminación de esta tesis se realizó la etapa Interpretativa con el análisis de resultados y las consiguientes conclusiones, las cuales se encuentran en los capítulos 8 y 9, respectivamente.

6.1. Enfoque metodológico

Esta investigación parte de la siguiente hipótesis: *El ciclo MAPE-K puede realizar control automático adaptativo en un sistema de Internet de las Cosas*. Con base en ella, se propone un desarrollo tecnológico que usa al ciclo MAPE - K, para realizar control automático.

6.2. Tipo de estudio

Este trabajo es un desarrollo tecnológico en la cual se busca la administración del control automático adaptativo por medio del ciclo MAPE-K al Internet de las Cosas, en un *Objeto de Internet*.

6.3. Diseño de la investigación

Para dar respuesta al objetivo de *Desarrollar un Control automático adaptativo por medio del ciclo MAPE-K, en el Internet de las Cosas, usando el System On Chip ESP32* se plantearon tres objetivos específicos que serán tratados a continuación.

Desarrollar un Objeto de Internet para realizar control automático, usando el System On Chip ESP32: Se diseñó un sistema embebido basado en el SoC *ESP32* para la implementación de un control automático en sistemas electrónicos. Posteriormente se desarrolló una Aplicación Web, que conectó al controlador con el usuario. Esta plataforma es programable y permite la configuración del *Objeto de Internet*.

Desarrollar una estrategia de control automático adaptativo por medio del ciclo MAPE-K: Para el desarrollo del control automático adaptativo se procedió a mapear los elementos del ciclo de control clásico en un Ciclo MAPE - K. La estrategia de control implementada en este trabajo es el controlador PID (Proporcional, Integrativo y Derivativo) y constituyó el Managed Element del ciclo MAPE-K. En la Aplicación Web se tomo los componentes restantes del ciclo MAPE-K, que no fueron implementados en el sistema embebido. Los desarrollos de software se administraron por medio de la herramienta *Git* y se publicaron por medio de repositorios públicos en *GitHub*.

Probar el funcionamiento del Control automático adaptativo por medio del ciclo MAPE-K, en el Internet de las Cosas, usando el System On Chip ESP32: Para

la comprobación el sistema embebido, la estrategia de control PID (Managed Element) se aplicó a varias plantas LTI electrónicas de prueba. Se estudió su comportamiento como controlador automático ante condiciones de desconocimiento de la dinámica de la planta y a perturbaciones externas. Finalmente, se realizó un análisis de los resultados y se presentaron las conclusiones. Las plantas de prueba se construyeron a partir de amplificadores operacionales, resistencias, capacitores y bobinas. Los circuitos electrónicos a controlar funcionaron en el rango de $0V$ a $3,3V$ y presentaron tiempos de establecimiento del orden de segundos. Las plantas de control propuestas son:

- Sistema Lineal Invariante en el Tiempo (LTI) de primer orden.
- Sistema Lineal Invariante en el Tiempo (LTI) de segundo orden.

En la figura **6-1** se observa el diagrama de bloques del sistema propuesto. En ella se presentan cada uno de los actores del proceso. En la sección inferior de la imagen se destaca los elementos que constituyen el control automático, en la porción central se especifica los participantes que pertenecen al Internet de las Cosas.

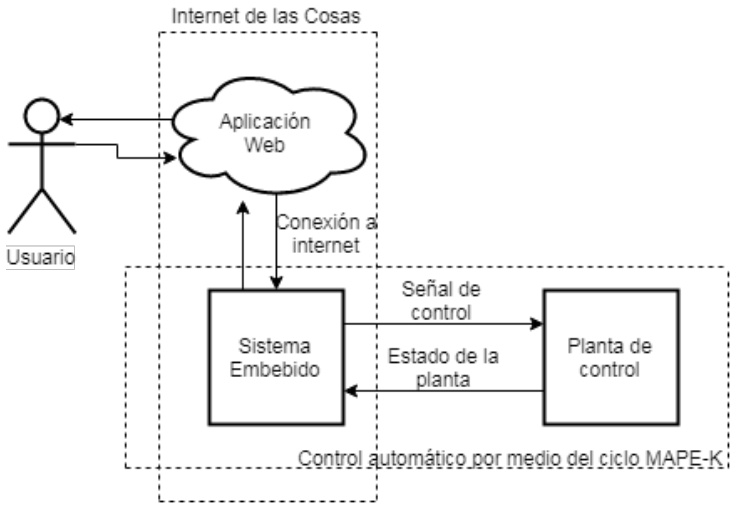

Figura 6-1.: Diagrama de bloques de la propuesta de tesis. Fuente: propia.

Para la realización de este trabajo se partió de la premisa de que el *Objeto de Internet* posee una conexión a Internet estable y por tanto solo se debe configurar sus acciones de control automático. Por tal motivo, se excluyen las fallas derivadas por la configuración de la comunicación al ciberespacio por parte del sistema embebido.

7. Resultados

7.1. Sistema embebido

Para elaborar al sistema embebido se seleccionó al *ESP32* entre las opciones presentadas en la tabla **4-3**, por contar con salidas basadas en DACs y ADCs de mayor precisión. Estas características lo hacen ideal para el tipo de control automático planteado en la presente tesis. En la figura **7-1** se presenta el módulo ESP32 tal y como es entregado por su fabricante. Esta presentación requiere de circuitería adicional para poder operarla y ser programada. Algunos otros fabricantes han tomado la presentación anterior y le han proporcionado la circuitería requerida, la cual permite apalancar proyectos como este. En la figura **7-2** se presenta el diagrama esquemático de la tarjeta ESPWROOM seleccionada. La tarjeta usada incluye puerto USB y es compatible con el IDE ARDUINO.

Figura 7-1.: Vista lateral del modulo ESP32 fabricado por la compañía Espressif. Fuente: tomado de [4]

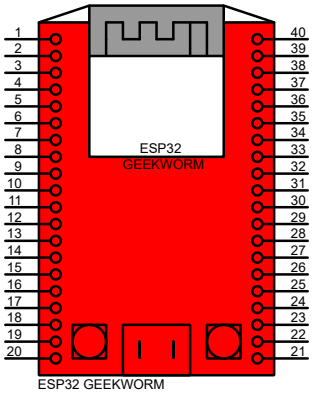

Figura 7-2.: Diagrama esquemático del *ESP32 GeekWorm*. Fuente: propia.

Si bien la tarjeta de la figura **7-2** es útil, no es suficiente para alcanzar los fines de realizar control automático. Por tal motivo se desarrolló una segunda tarjeta que reciba la anterior y provea los elementos necesarios para el control electrónico. En la figura **7-3** se presenta el diagrama esquemático de esta tarjeta, inspirada en los shields para ARDUINO NANO.

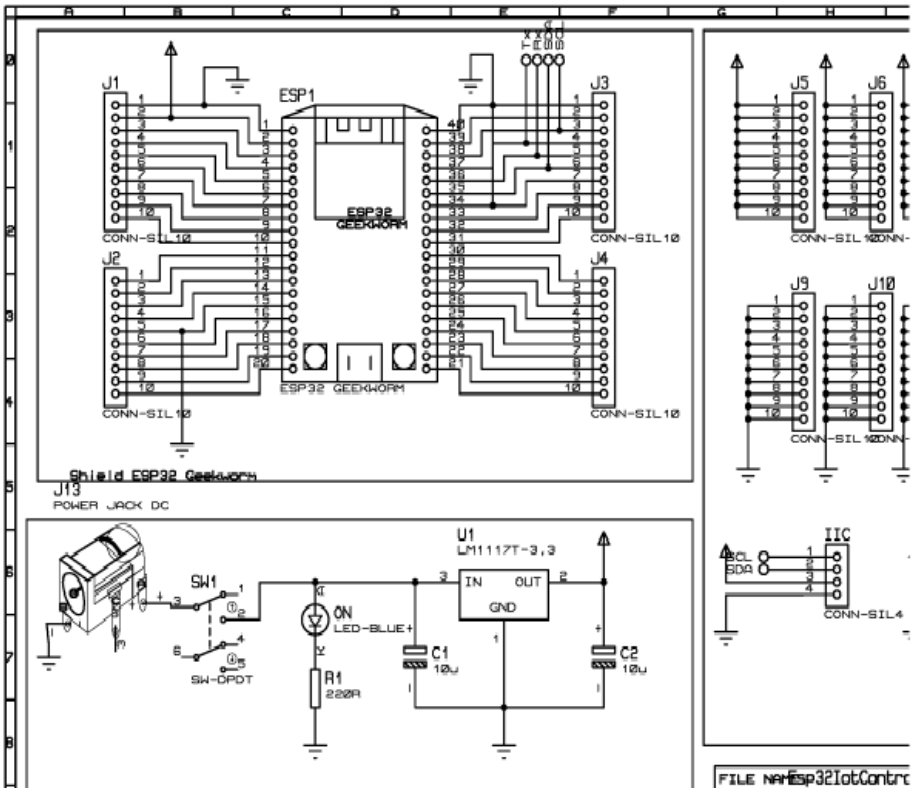

Figura 7-3.: Shield para ESP32 weekworm. Detalle del circuito esquemático. Fuente: propia.

Posteriormente, se empleó el *ESP32 DEVKITV1* para generar un sistema embebido más acorde al control automático. En las figuras **7-4**, **7-5**, **7-6** y **7-7** se observan los diagramas esquemáticos de todos elementos usados en la nueva tarjeta.

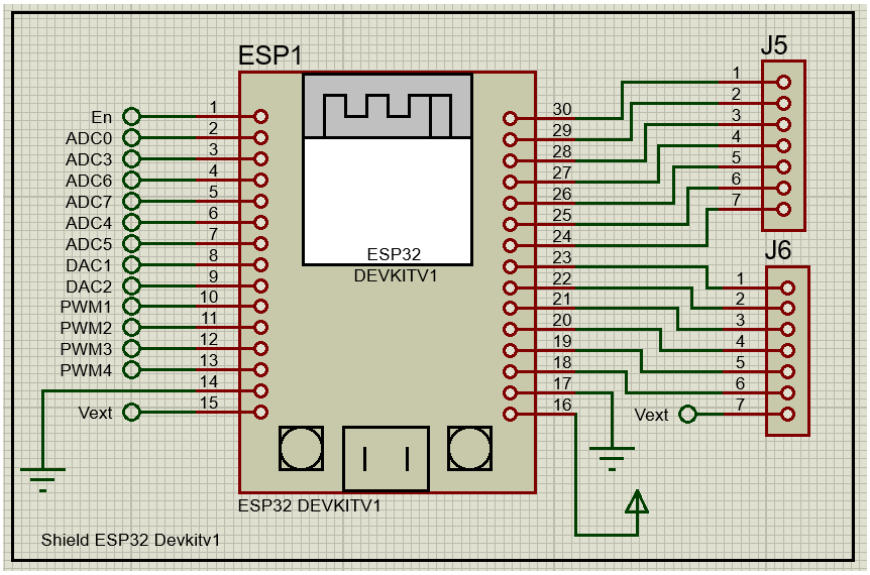

Figura 7-4.: Diagrama esquemático del Shield ESP32 Devkitv1. Fuente: propia

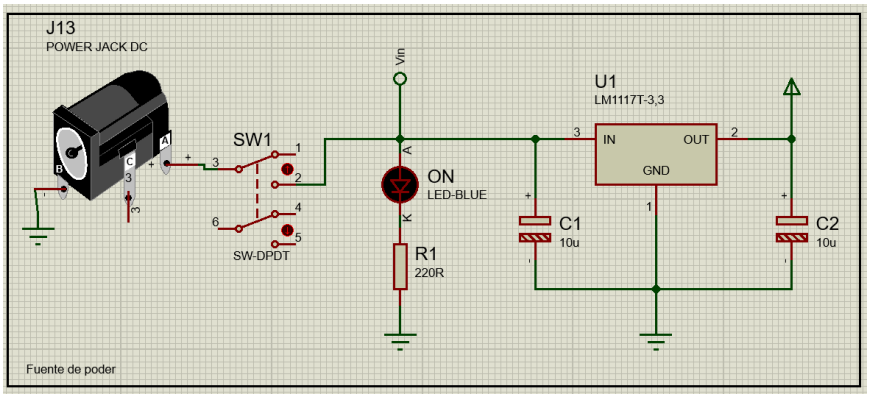

Figura 7-5.: Diagrama esquemático de la fuente interna de alimentación. Fuente: propia.

Entre los cambios más relevantes se encuentran la inclusión del amplificador operacional LM324 en su configuración seguidor de voltaje no inversor. La finalidad de este es evitar daños en el *ESP32* por cuentas de sobrecorrientes o sobrevoltajes. Otro aspecto a señalar es la conexión interna del ADC al DAC, esto es con el objetivo de contar con una medida

de mayor resolución de la salida de control automático. El ADC del ESP32 posee con una resolución de 12$bits$, en tanto que, su DAC cuenta solo con una resolución de 8$bits$. En la figura **7-8** se muestra su Printed Circuit Board (PCB) de la nueva tarjeta.

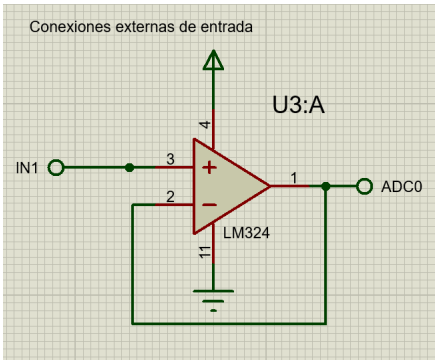

Figura 7-6.: Diagrama esquemático de la entrada analógica. Fuente: propia

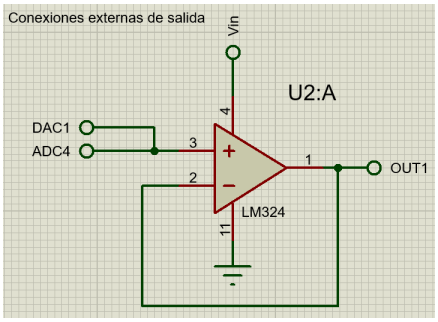

Figura 7-7.: Diagrama esquemático de la salida analógica. Fuente: propia.

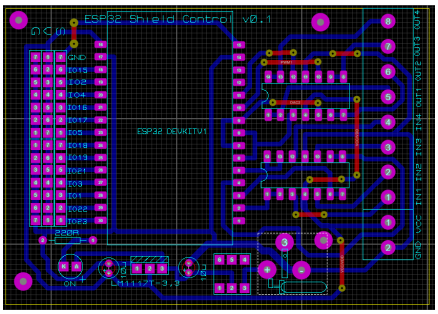

Figura 7-8.: Diseño del PCB de la tarjeta de control

Una vez creado el sistema embebido se plantea un esquema de control electrónico que reciba las plantas a controlar. En la figura **7-9** se modela la conexión entre estos elementos. Como

los sistemas a controlar poseen una entrada y salida de voltaje, el sensor es el mismo ADC.

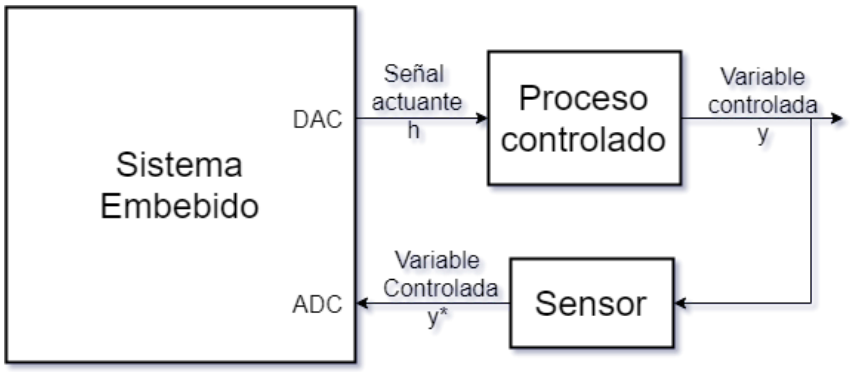

Figura 7-9.: Esquema de control en un sistema embebido. Fuente: propia

En la figura **7-10** se muestra por medio de un diagrama de secuencia la comunicación entre el sistema embebido y la plataforma Web desarrollada.

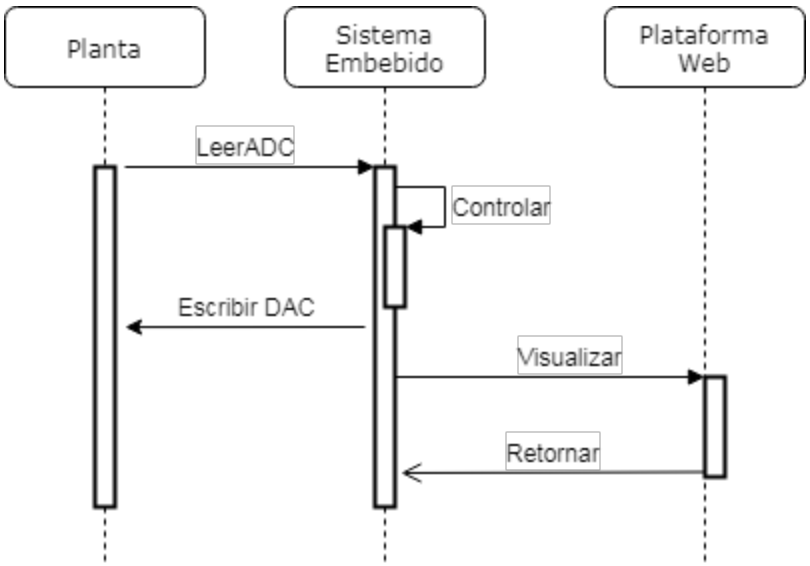

Figura 7-10.: Diagrama de secuencia del control realizado por un sistema embebido usando monitoreo IoT. Fuente: propia.

La tarjeta desarrollada como parte del sistema embebido recibe el nombre de *IotControl*.

7.2. Aplicación Web

Como parte del desarrollo de un *Objeto de Internet* se creó una aplicación web para la comunicación entre objetos y personas. Esta aplicación Web se desarrolló a partir del mapa del sitio presentado en la figura **7-11**. Los datos almacenados en la plataforma Web se diseñaron por medio del *Modelo Entidad - Relación* de la figura **7-12**.

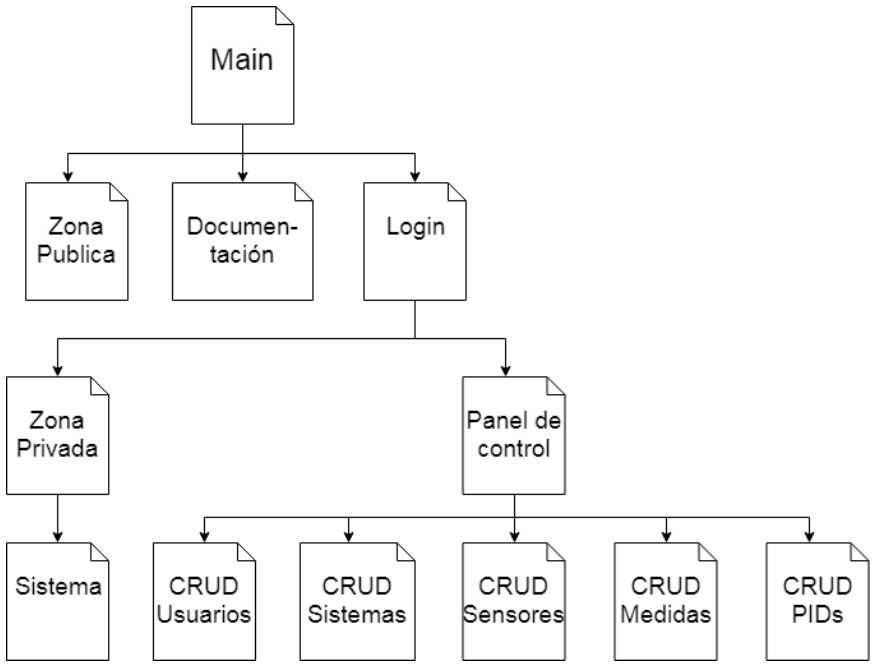

Figura 7-11.: Mapa del sitio para la Aplicación Web. Fuente: propia.

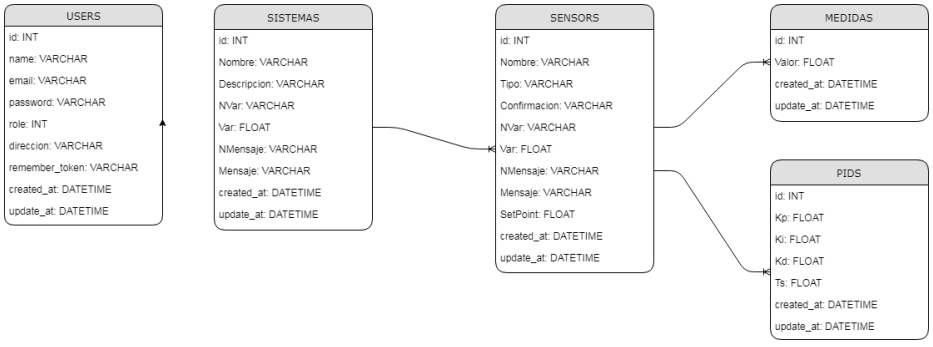

Figura 7-12.: Modelo Entidad - Relación de los datos en la Plataforma Web. Fuente: propia.

La interfaz gráfica de las distintas páginas Web se observan en las figuras **7-13**, **7-14**, **7-15**,

7-16 y **7-17**. En el anexo D se observa el manual de usuario que incluye el resto de la interfaz gráfica.

Figura 7-13.: Interfaz Gráfica Main de la plataforma Web. Fuente: propia.

Figura 7-14.: Interfaz gráfica de Login en la plataforma web. Fuente: propia.

Figura 7-15.: Interfaz gráfica para registro en la plataforma web. Fuente: propia.

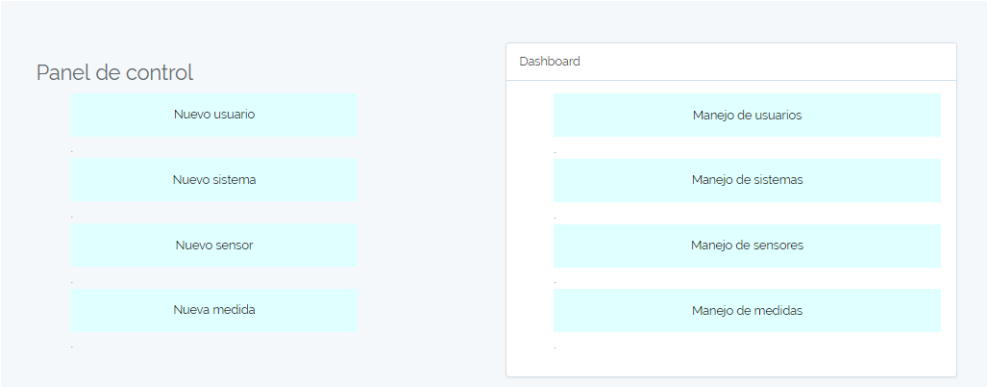

Figura 7-16.: Interfaz gráfica del panel de control. Fuente: propia.

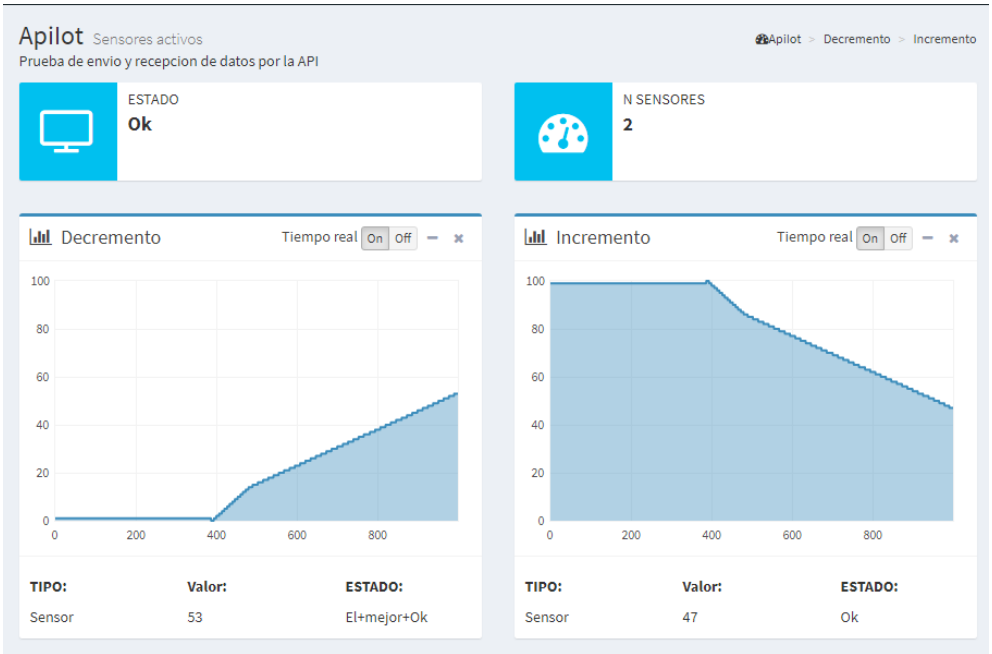

Figura 7-17.: Interfaz gráfica de un sistema. Fuente: propia.

Se creó una Application Programming Interface (API) para la comunicación entre la aplicación Web y los sistemas embebidos conectados a ella. Los datos intercambiados se entregan por medio de una notación JavaScript Object Notation (JSON). En la tabla **7-1** se muestra el esquema JSON usado. Para el intercambio de los sensores se elaboró otra estructura que se muestra en la tabla **7-2**. Los datos de intercambio en ambos casos son RESTful y enviados por la URL, con lo cual se busca disminuir la carga del servidor.

Tabla 7-1.: Estructura de notación Json usada para intercambiar datos entre la aplicación Web y el sistema embebido. Fuente: propia.

```
{
        "id": 3,
        "Nombre": "Planta de prueba",
        "Descripcion": "Planta de prueba usada en la tesis de
           maestria de Cesar Alvarez",
        "NVar": "Experimento",
        "Var": "0",
        "NMensaje": "Estado actual",
        "Mensaje": "En prueba",
        "sensors": [
                {
                        "id": 3,
                        "Nombre": "Esp32 ADC",
                        "Tipo": "Sensor",
                        "NVar": "Voltaje [V]",
                        "Var": "0",
                        "NMensaje": "Estado",
                        "Mensaje": "Controlado"
                        "SetPoint": "1.0",
                }
                {
                        "id": 4,
                        "Nombre": "ESP32 DAC",
                        "Tipo": "Actuador",
                        "NVar": "Voltaje [V]",
                        "Var": "0",
                        "NMensaje": "Estado actual",
                        "Mensaje": "control"
                        "SetPoint": "1.0",
                },
        ]
}
```

Tabla 7-2.: Estructura de notación Json usada para intercambiar datos del sensor entre la aplicación Web y el sistema embebido. Fuente: propia.

```
{
        "id": 3,
        "Nombre": "Esp32 ADC",
        "Tipo": "Sensor",
        "NVar": "Voltaje [V]",
        "Var": "0",
        "NMensaje": "Estado",
        "Mensaje": "Controlado",
        "SetPoint": "1.0",
        "pids": [
                {
                        "id": 2,
                        "Kp": "2.05",
                        "Ki": "186.16",
                        "Kd": "0",
                        "Ts": "0.01",
                        "sensor_id": 3,
                        "created_at": "2019-07-26 12:23:34",
                        "updated_at": "2019-07-26 12:23:34"
                }
        ]
}
```

Se usó el framework Laravel para la construcción de la plataforma. Laravel implementa una adaptación del *Modelo Vista Controlador* tradicional y presenta la arquitectura de la figura **7-18**. Basado en el anterior esquema se diseñó la aplicación Web y se diseñaron controladores para la lógica de la página. Las vistas se realizaron usando HTML5, CSS y JavaScript. Los Modelos permitieron realizar lo propuesto en la figura **7-12**. Para el manejo de la base de datos se utilizó *Eloquent ORM* de Laravel. El OMR enmascara las solicitudes SQL y brinda una interfaz basada en Objetos de software, motivo por el cual no se depende un tipo particular de SQL. Los datos a graficar se obtuvieron por medio de solicitudes *AJAX*.

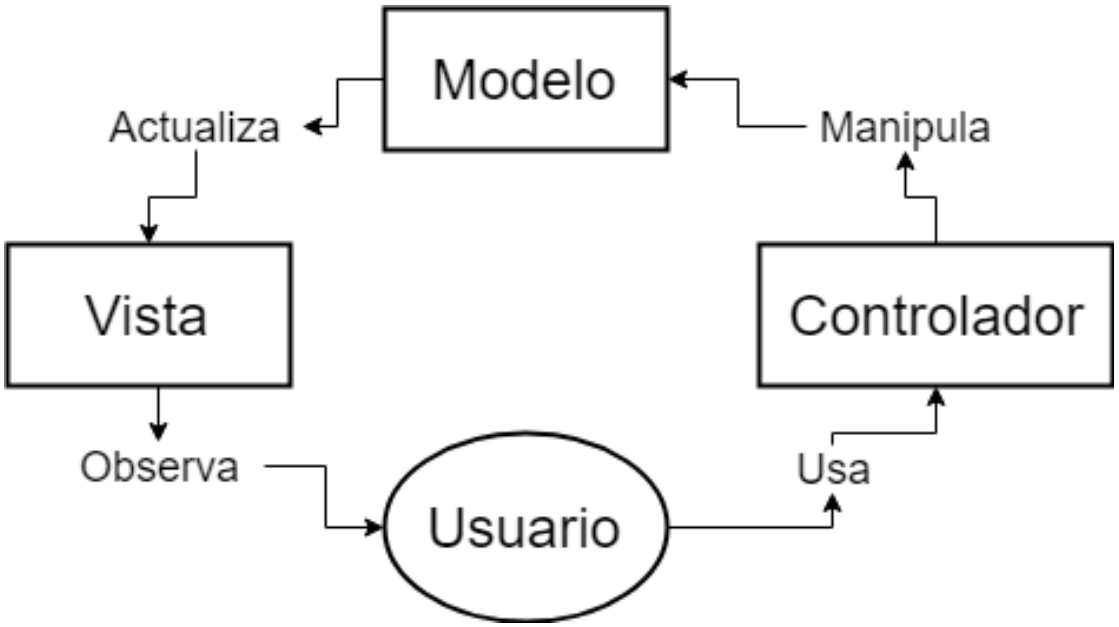

Figura 7-18.: Modelo Vista Controlador usado en el diseño de aplicaciones web. Fuente: tomado de [63]

Para subir y administrar la aplicación Web en la nube se utilizó la Plataforma como Servicio (PaaS) *Heroku*. La aplicación Web se puede encontrar en la url: `http://iotview.herokuapp.com`. Se documentó el proceso por medio del repositorio *GitHub* y se puede encontrar el proyecto en el link: `https://github.com/CesarAlvarezG/IoTView`. Se le aplico la plantilla *AdminLTE*, disponible en `https://github.com/almasaeed2010/AdminLTE/`.

7.3. Firmware MAPE-K

Se desarrolló un firmware, para el sistema embebido presentado en el apartado anterior, el cual implementa un control adaptativo por medio del ciclo MAPE-K. Este se desarrolló por medio del IDE ARDUINO y las bibliotecas especializadas provistas por la compañía *Espressif*. Se versionó su avance por medio de la plataforma GitHub y se encuentra en el link `https://github.com/CesarAlvarezG/IoTEsp`. La primera parte del desarrollo se basó en implementar un espejo de la información contenida en la plataforma Web. En la figura **7-19** se observa el diagrama de clases realizado.

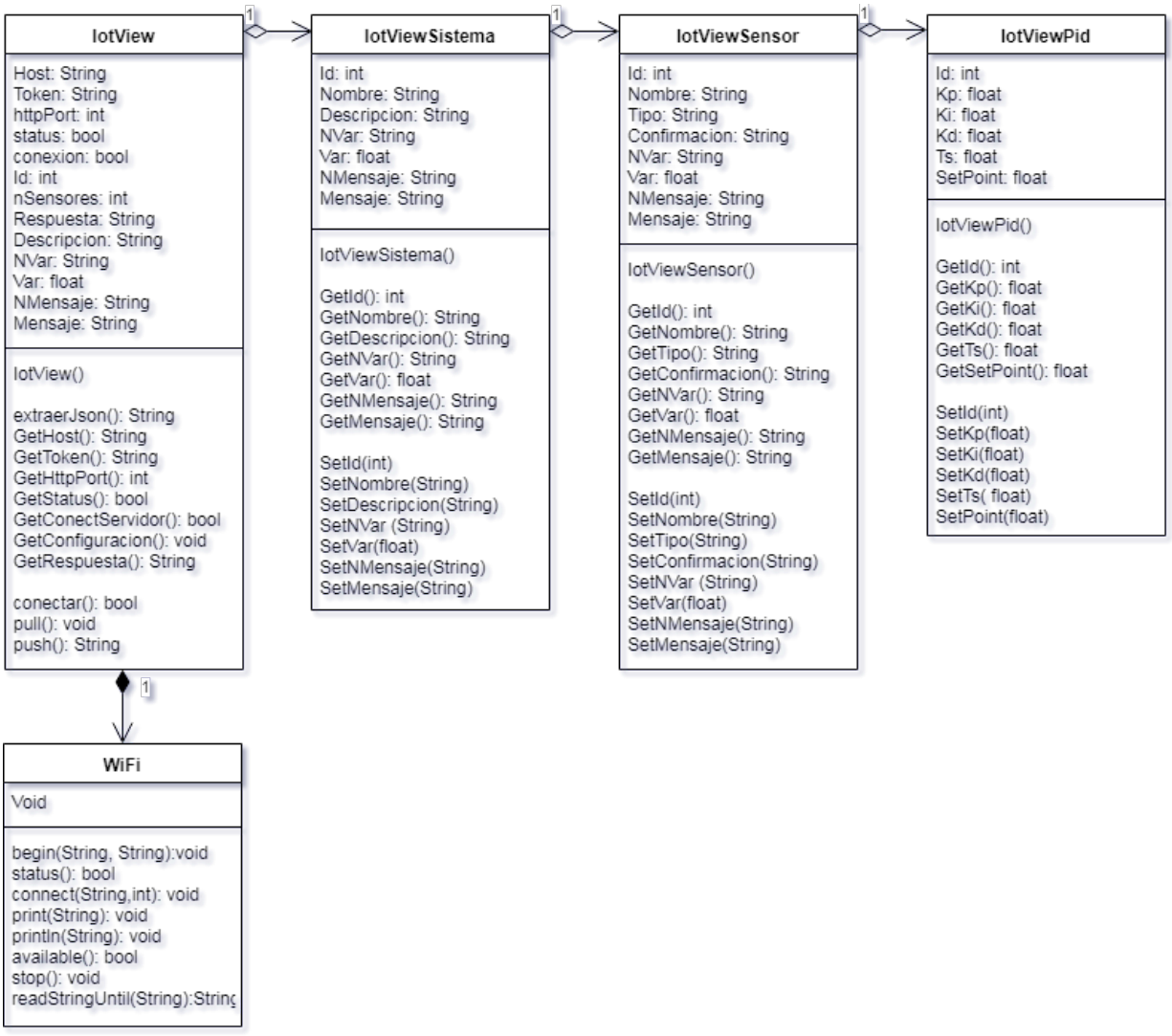

Figura 7-19.: Diagrama de clases en sistema embebido. Fuente: propia.

En la segunda parte, con la finalidad de implementar el ciclo MAPE-K al anterior firmware se elaboró un Objeto de Software en el lenguaje C++, compatible con ARDUINO. El funcionamiento del ciclo MAPE-K se puede modelar en la figura **7-20**. Este primer ciclo MAPE-K representa el lazo de control cerrado tradicional. En la figura **7-21** se observa la representación del objeto.

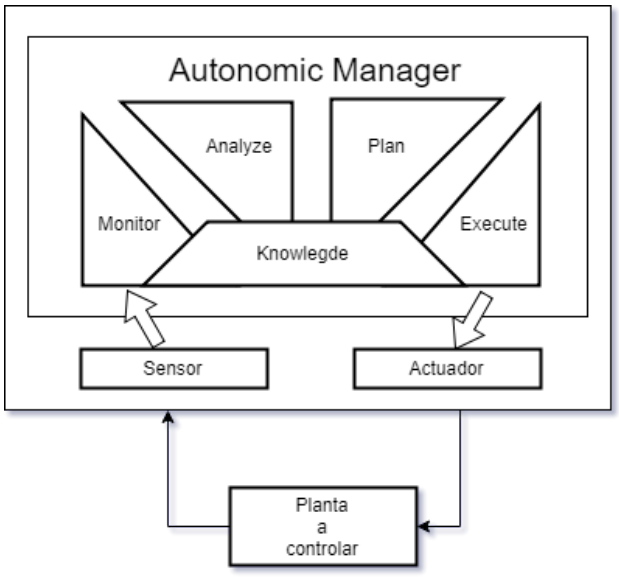

Figura 7-20.: Esquema básico de control por medio del ciclo MAPE-K. Fuente: propia.

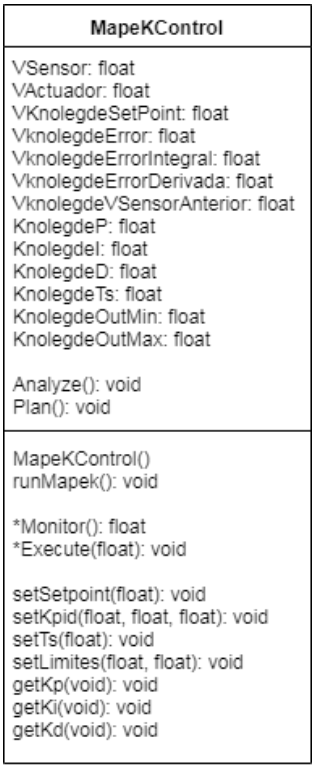

Figura 7-21.: Diagrama de clases de ciclo MAPE-K para hacer control automático. Fuente: propia.

El mapeo de las partes del ciclo MAPE-K llevó a los siguientes elementos. El puntero a

función *monitor()* se enlaza a una rutina externa que toma la lectura del ADC proveniente de la salida del sistema a controlar y la convierte al valor de voltaje correspondiente a la variable a controlar *VSensor*. En la función *analyze()* se calcula, por medio de la ecuación 7-1, la desviación (error) de la señal de salida del sistema respecto al valor deseado o *set-Point*, igualmente se procedió a calcular la derivada y la integral del error, por medio de las ecuaciones 7-2 y 7-3.

$$VknoloegdeError = VknoloegdeSetPoint - VSensor \tag{7-1}$$

$$VknoloegdeErrorIntegral+ = VknoloegdeError * VknoloegdeTs \tag{7-2}$$

$$VknoloegdeErrorDerivada = \frac{VknoloegdeError - VknoloegdeVSensorAnterior}{VknoloegdeTs} \tag{7-3}$$

Donde:

VknoloegdeError: Valor del error en la acción del controlador.

VknoloegdeSetPoint: Setpoint del sistema a controlar.

VSensor: Valor obtenido del sensor.

VknoloegdeError: Error de la acción de control.

VknoloegdeTs: Periodo de muestreo.

VknoloegdeErrorIntegral: Integral del error.

VknoloegdeErrorDerivada: Derivada del error.

VknoloegdeVSensorAnterior: Valor anterior obtenido del sensor.

El elemento *plan()* calcula la acción del controlador con base al error encontrado por medio de la ecuación 7-4, para el presente caso se corresponde a un control PID. El puntero a función *execute()* se enlaza a otra rutina que recibe el valor en voltaje para aplicar la acción de control por medio del DAC, usando la variable *VActuador*. Se emplearon punteros a

función para implementar las acciones *monitor()* y *execute()*, con la intención de dotarlas de cierta flexibilidad que permitiera cambiar el uso de ADCs por sensores digitales o el de los DACs por PWMs.

$$VActuador = + \begin{matrix} VknoloegdeP * VknoloegdeError \\ VknoloegdeI * VknoloegdeErrorIntegral \\ VknoloegdeD * VknoloegdeErrorDerivada \end{matrix} \quad (7\text{-}4)$$

Donde:

VActuador: Valor enviado al actuador.

VknoloegdeP: Constante P del control PID.

VknoloegdeI: Constante I del control PID.

VknoloegdeD: Constante D del control PID.

Posteriormente, se implementó un segundo ciclo MAPE-K al controlador PID con la finalidad de verificar su correcto funcionamiento y hacerlo adaptativo. En la figura **7-22** se observa al ciclo MAPE-K actuar sobre el controlador. Finalmente en la figura **7-23** se observa el panorama completo de los ciclos MAPE-K en conjunto.

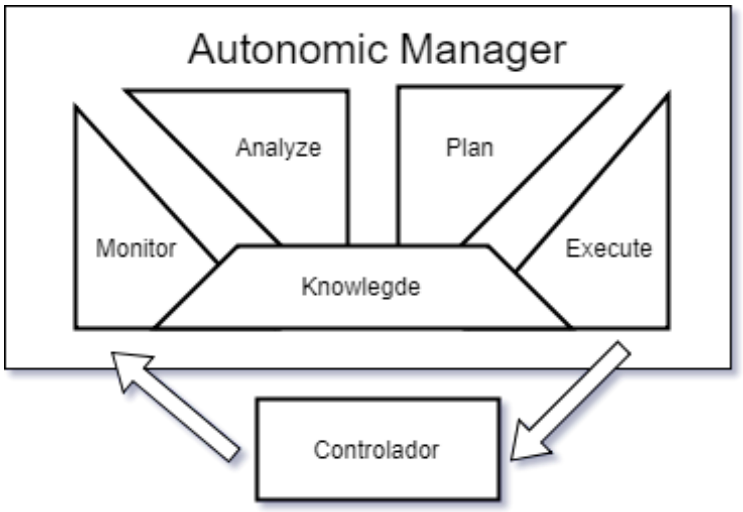

Figura 7-22.: Esquema básico de la administración del controlador. Fuente: propia.

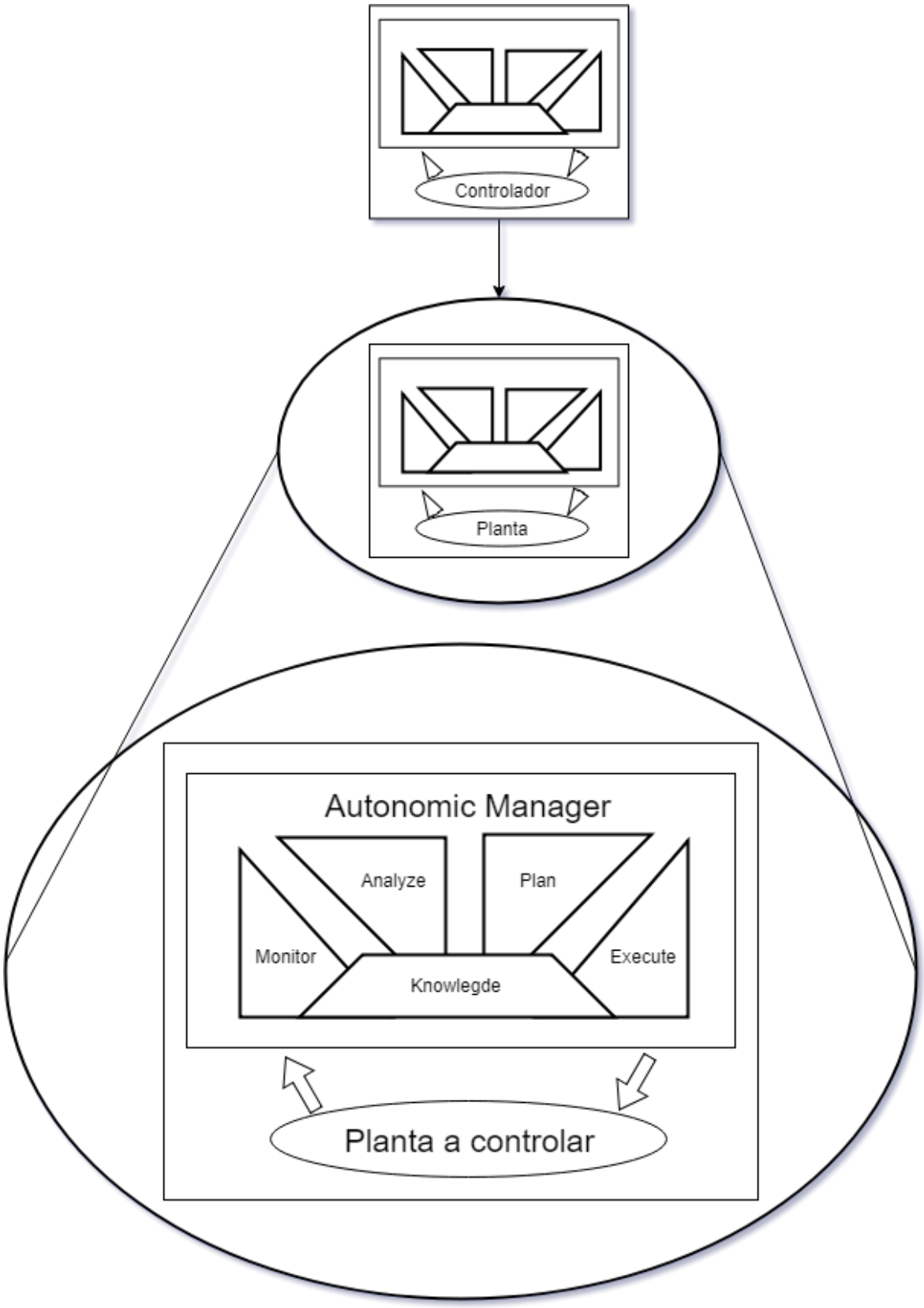

Figura 7-23.: Esquema básico de la administración de la planta a controlar y el controlador. Fuente: propia.

El segundo ciclo MAPE-K busca establecer el valor adecuado de las constantes proporcional ($VknoloegdeP$), integral ($VknoloegdeI$) y derivada ($VknoloegdeD$). En la figura **7-24** se observa el diagrama de clases de este nuevo objeto de software.

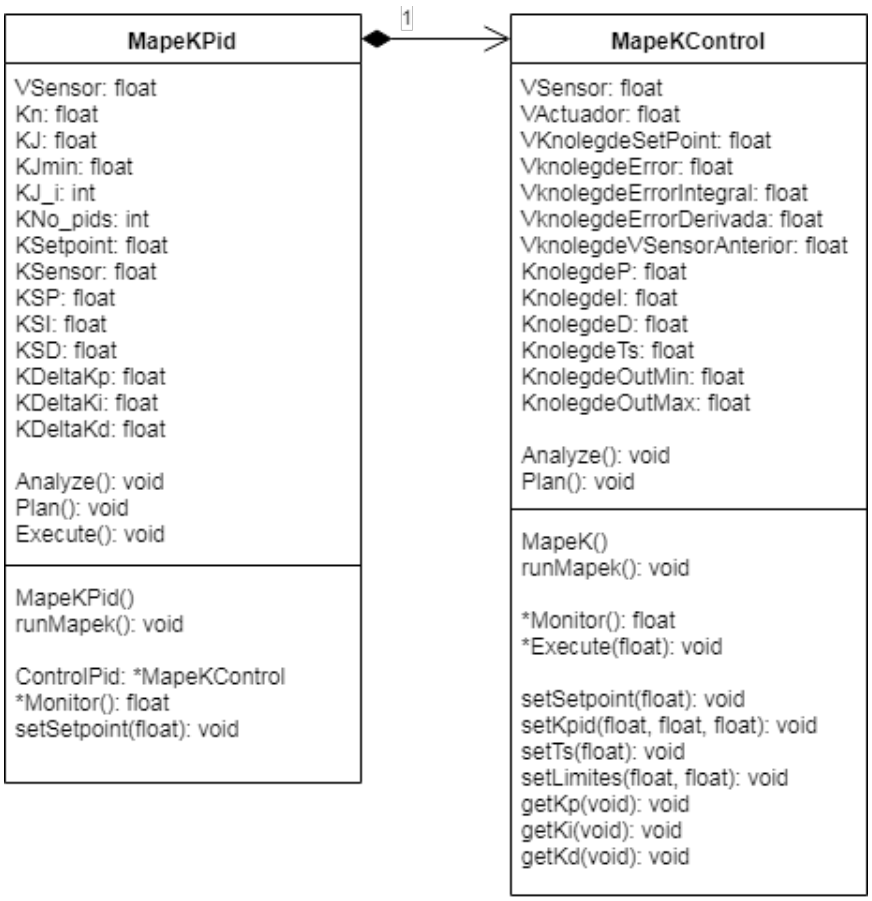

Figura 7-24.: Diagrama de clases de ciclo MAPE-K para hacer control automático. Fuente: propia.

El mapeo de las partes del segundo ciclo MAPE-K llevo a los siguientes elementos. El puntero a función *monitor()* se enlaza a una rutina externa que toma la lectura del ADC proveniente de la salida del sistema a controlar y la convierte al valor de voltaje correspondiente a la variable a controlar *VSensor*. Esta es la misma rutina del primer ciclo MAPE-K, lo cual garantiza la calidad de las medidas. En la función *analyze()* se calcula, por medio de la ecuación 7-5, el índice de actuación del control PID con respecto al escalón del setpoint.

$$KJ = \frac{1}{2}\int_0^t (Kerror)^2 dt \tag{7-5}$$

Donde:

KJ: Indice de desempeño del controlador, función costo.

Kerror: Error del controlador.

t: Tiempo.

En la etapa *plan()* se aplica una función de adaptación simple a cada constante. En las ecuaciones 7-6, 7-7 y 7-8 se observa el cálculo de la variación de cada constante.

$$\Delta Kp = -KS_p * \nabla KJ \tag{7-6}$$

$$\Delta Ki = -KS_i * \nabla KJ \tag{7-7}$$

$$\Delta Kd = -KS_d * \nabla KJ \tag{7-8}$$

Donde:

KS_p**:** Ganancia de adaptación de la función Proporcional.

KS_i**:** Ganancia de adaptación de la función Integral.

KS_d**:** Ganancia de adaptación de la función Derivativa.

*grad***:** Gradiente de la función costo.

ΔKp**:** Cambio a aplicar en la constante Proporcional.

ΔKi**:** Cambio a aplicar en la constante Integral.

ΔKd**:** Cambio a aplicar en la constante Derivativa.

El método *execute()* aplica la variación de cada constante del controlador PID al primer ciclo MAPE-K.

7.4. Pruebas de funcionamiento

Para verificar el funcionamiento del sistema se planteó una primera prueba para el objeto de software *MAPEKControl* y una segunda para el otro objeto de software *MAPEKPid*. En la primera prueba para comprobar el adecuado comportamiento del ciclo de control MAPE-K se diseñó un sistema dinámico de primer orden. Al cual es relativamente fácil realizar un control automático. En la ecuación 7-9 se observa la función de transferencia, la cual sirvió para afinar el ciclo MAPE-K de control.

$$\frac{V_o(s)}{Vi(s)} = \frac{1}{0{,}01s + 1} \tag{7-9}$$

Donde:

$V_o(s)$**:** Voltaje de salida.

$V_o(s)$**:** Voltaje de entrada.

s**:** Variable compleja de la transformada de Laplace.

El sistema dinámico se implementó por medio de un circuito de resistencia (R) y capacitancia (C) eléctrica en serie, como se muestra en la figura 7.4. Los valores del circuito RC son $R = 10k\Omega$ y $C = 1\mu F$. En la ecuación 7-10 se presenta su función de transferencia. Acto seguido, se implementó y probó por medio del sistema embebido *IotControl*, resultado del presente trabajo de grado. En la figura **7-26** se observa su comportamiento ante un escalón, la ilustración incluye su dinámica simulada y su desempeño real medido.

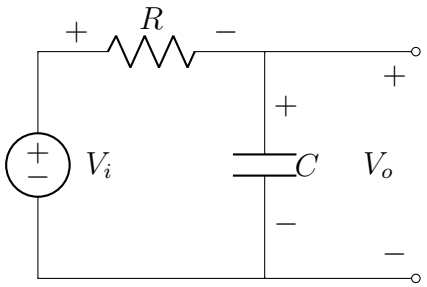

Figura 7-25.: Circuito eléctrico de primer orden por medio de una red RC. Fuente: propia.

$$\frac{V_o(s)]}{Vi(s)} = \frac{1}{RCs + 1} \tag{7-10}$$

Donde:

R**:** Resistencia eléctrica.

C**:** Condensador eléctrico.

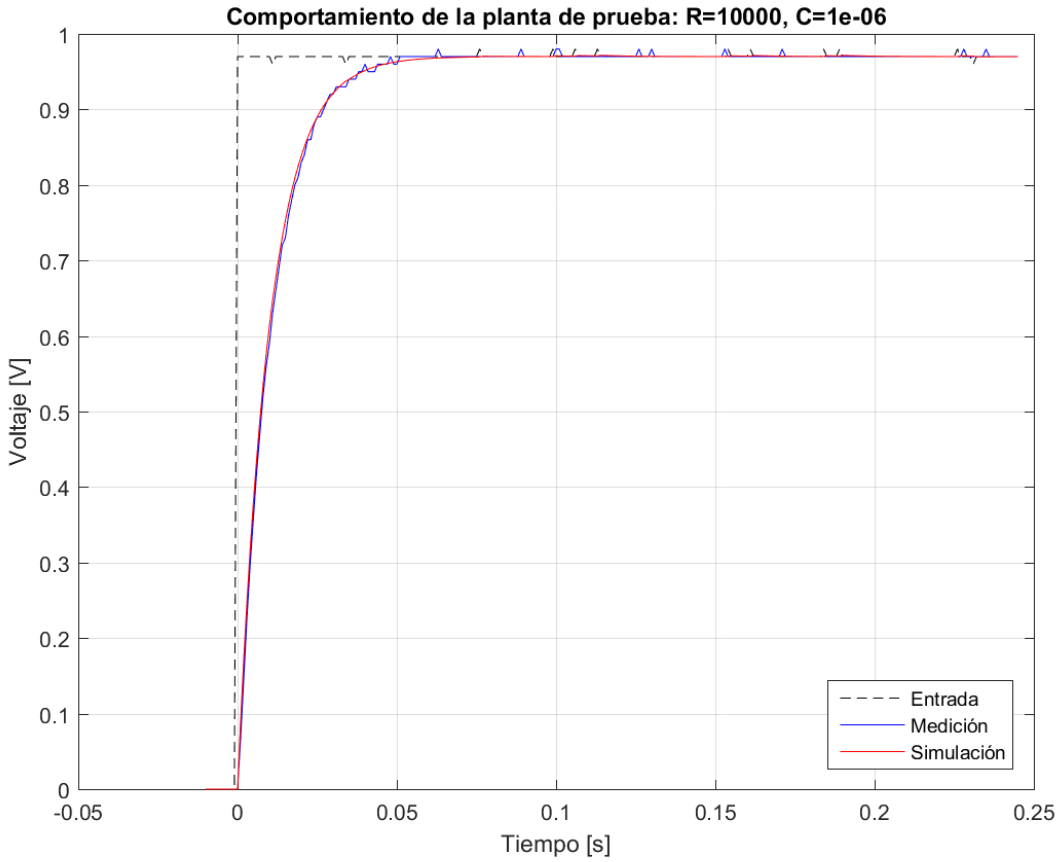

Figura 7-26.: Comportamiento de la planta de primer orden de prueba. Fuente: propia.

Se empleó la herramienta *pidTuner* de *MatLab* para calcular las constantes Kp, Ki y Kd de un control PID paralelo, obteniéndose los valores de la ecuación 7-11. Ya que el sistema se implementó en sistema embebido se empleó un control digital en tiempo discreto con un periodo de muestreo arbitrario de $1ms$. El periodo se eligió para cumplir con el teorema del muestreo. En la figura **7-27** se observa el resultado de usar estas constantes en la herramienta *PIDArduino* y el ciclo *MAPEKControl*.

$$Kp = 1{,}6884 \; Ki = 258{,}8284 \; Kd = 0{,}00077948 \tag{7-11}$$

Donde:

Kp: Constante Proporcional arroja por la herramienta pidTuner de MatLab.

Ki: Constante Integral arroja por la herramienta pidTuner de MatLab.

Kd: Constante Derivativa arroja por la herramienta pidTuner de MatLab.

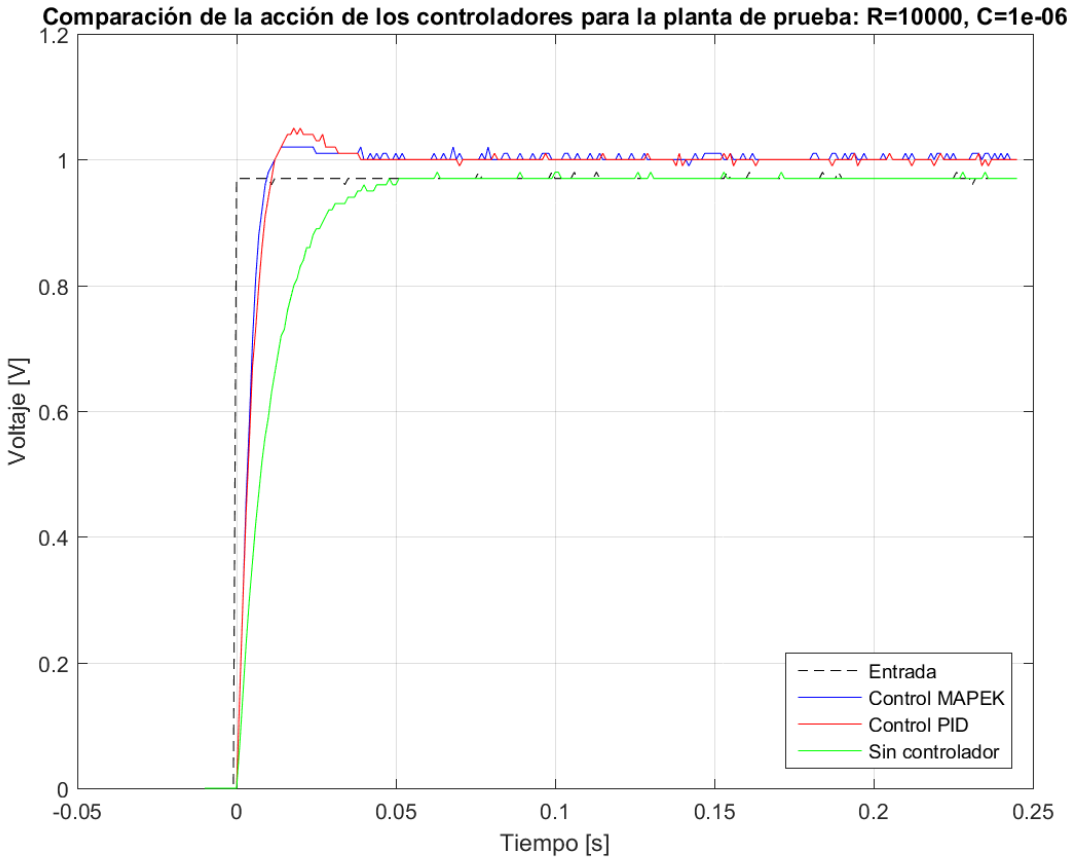

Figura 7-27.: Comparación de los controladores MAPE-K y Arduino PID. Fuente: propia.

En la segunda prueba de funcionamiento del sistema se diseñó un nuevo sistema LTI, esta vez de segundo orden. Para implementar el sistema LTI se seleccionó un circuito RLC en serie, descrito por la ecuación 7-12 y cuya conexión se observa en la figura 7.4. Los valores del circuito RLC son $R = 820\Omega$, $L = 1mH$ y $C = 1\mu F$. La diferencia entre el circuito de la figura **7-25** y el de la figura **7-28** es la introducción de una inductancia y el cambio en la resistencia del sistema, la cual cambia su dinámica interna.

$$\frac{V_o(s)]}{Vi(s)} = \frac{1}{LCs^2 + RCs + 1} \tag{7-12}$$

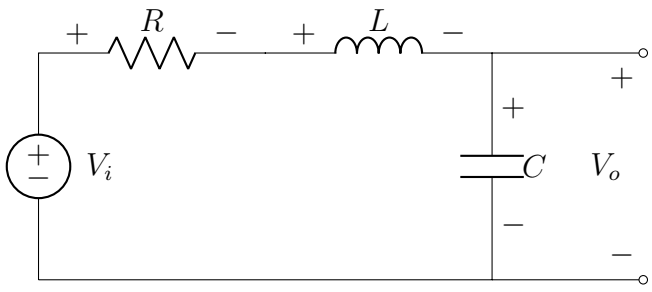

Figura 7-28.: Circuito eléctrico de segundo orden por medio de una red RLC. Fuente: propia.

El cambio en la dinámica de la planta de prueba por parte de la inclusión de la inductancia y el cambio en la resistencia se observa en la figura **7-29**, se puede evidenciar el cambio significativo en su comportamiento.

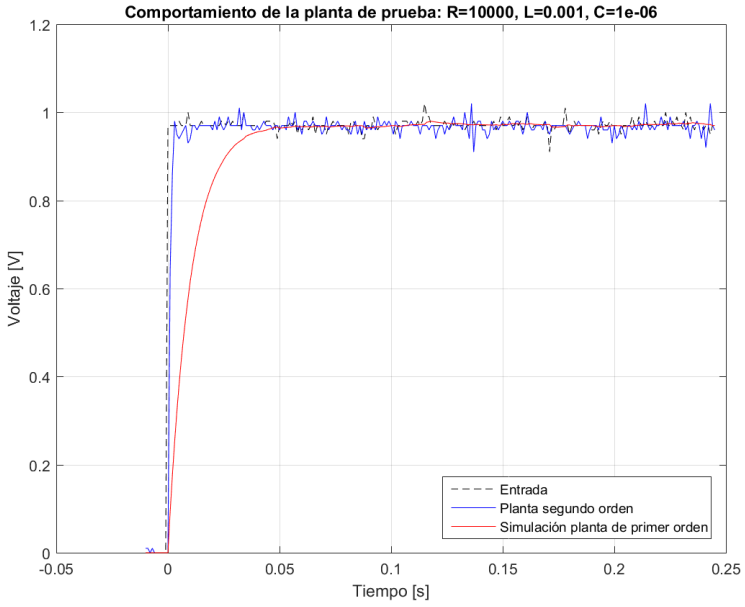

Figura 7-29.: Comportamiento de la planta de prueba de segundo orden compararada con la de primer orden. Fuente: propia.

En la figura **7-30** se observa la adaptación del nuevo sistema de control con el segundo ciclo MAPEK. En la figura **7-31** se observa un resumen de los logros del ciclo MAPEK.

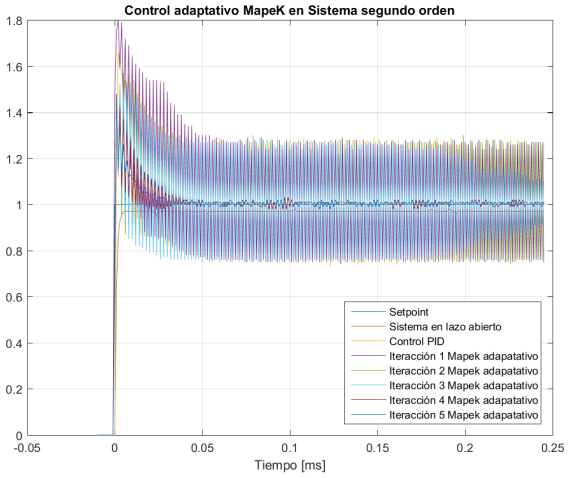

Figura 7-30.: Adaptación del control PID por parte del segundo ciclo MAPEK

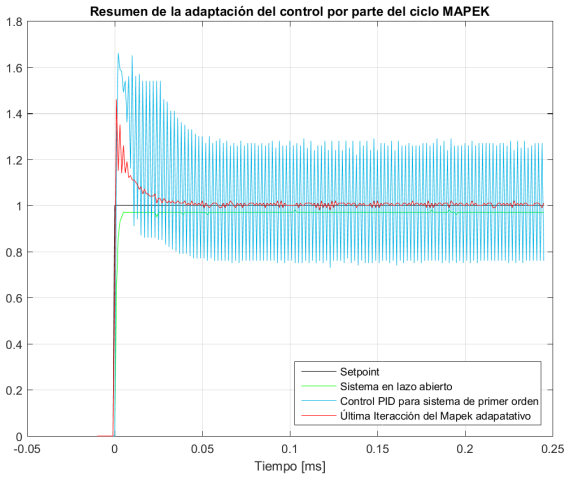

Figura 7-31.: Comparación del control PID para el sistema de primer orden con la adaptación del ciclo MAPEK

El control PID intenta ajustar al sistema y reacciona muy rápido a los cambios, lo cual produce oscilaciones al sistema. Por otra parte, el control por el objeto *MapeKControl* presenta un comportamiento similar, por lo cual se verifica la acción de control sobre la planta.

8. Discusión de los resultados

La elección del SoC ESP32 resultó satisfactoria en la creación de un sistema de control automático basado en el Internet de las Cosas. El uso de sus ADCs y DACs permitió realizar un control eléctrico en el rango de voltaje seleccionado. La programación basada en el *IDE ARDUINO* permitió una compaginación con el lenguaje C++ lo que amplía la compatibilidad de los desarrollos realizados para el ESP32 al ESP8266. El shield *ESP32 weekworm*, ver figura **7-3**, dotó a los primeros experimentos de la versatilidad necesaria para realizar cambios rápidos de conexión entre las plantas de control y los otros equipos electrónicos probados. El siguiente shield *ESP32 Devkitv1*, mostrado en figura **7-4**, se desarrolló para construir un tarjeta de control electrónico más acorde a las necesidades de un entorno industrial, bajo el rango de voltaje de $0V$ a $3{,}3V$. Las protecciones basadas en el amplificador operacional *Lm324*, revisar figura **7-6**, trabajaron según lo esperado y evitaron daños durante las experimentaciones. La toma de los datos de la salida del DAC, por medio de otro ADC, estudiar figura **7-7** permitió realizar las tareas del ciclo MAPE-K de forma más acertada. Otro aspecto a resaltar en la elección del SoC ESP32 es su reducido precio con respecto a otros sistemas con similares características, El System On Chip *ESP32* se puede conseguir por un valor de $40,000 pesos en el mercado colombiano, en tanto que otras soluciones como la Raspberry pi o similares en valores superiores a los $100,000 pesos.

La conexión planteada en la figura **7-9** funcionó según lo esperado y se conservó durante todo el desarrollo experimental. Igualmente, el diagrama de secuencia de la figura **7-10** se empleó a lo largo del trabajo. Se modificó la tarea de visualizar para ajustar a un tiempo de muestreo diferente, en función de las necesidades del usuario; esto sin perjudicar el proceso de control y evitando saturar la red. El desarrollo de una nueva aplicación Web obedeció a la necesidad de contar con la suficiente libertad para probar e implementar los algoritmos necesarios para realizar los objetivos de la tesis, otro factor que influyó en la decisión fueron los tiempos de muestreo previstos en plataformas IoT como ThingSpeak, Blynk y otros; en el orden de unos cuantos segundos. Estos están seleccionados por las restricciones propues-

tas por sus creadores y estas obedecen a la necesidad de no saturar la red por parte de los miles o millones de proyectos registrados en sus aplicaciones. El desarrollo de una plataforma del Internet de las Cosas es particularmente interesante por cuanto se puede acoplar otras herramientas especializadas como WEKA para poder realizar análisis de datos [69].

En la producción académica de la presente tesis se realizaron 4 tesis de pregrado en la universidad del Quindío. Estas tesis fueron dirigidas por el autor se nombran a continuación: *Desarrollo de un módulo de Smart Metering Para el servicio de energía Eléctrica* [5], *Reconocimiento de rostros para control de accesos* [16], *Tarjeta smart house para una habitación* [12] y *Laboratorio remoto de redes neuronales* [62].

El mapa del sitio en la figura **7-11** fue el resultado de múltiples mejoras a lo largo del periodo de desarrollo. La aplicación Web fue diseñada para realizar actividades de almacenamiento de información y para ello incorpora una base de datos relacional, en la figura **7-12** se muestra el modelo entidad-relación resultante. Por la forma como se estableció el depósito de la información de los sensores fue posible establecer un repositorio de controladores PIDs para poder llevar a cabo un control adaptativo. Este elemento es relevante porque así los sensores pueden almacenar PIDs probados y usar esta experiencia o conocimiento en el futuro en caso de necesitarlo. IotView cuenta con una interfaz gráfica que facilita su utilización, en las figuras **7-13**, **7-14**, **7-15**, **7-16** y **7-17** se muestra parte de esta. El diseño centrado en un administrador permite mantener la coherencia entre los distintos sistemas.

El intercambio de datos entre la aplicación Web y el sistema embebido se realizó por medio de una API diseñada para tal fin. Los datos se estructuraron usando JSON. En la tabla **7-1** se muestra un ejempló. Para la traducción del código se empleo el lenguaje de programación PHP. Se desarrolló el proyecto de tal forma que estén separadas las cargas de procesamiento. Para facilitar la codificación y ofrecer una mayor seguridad en todos los procesos se utilizó el framework *LARAVEL*. La introducción de esta herramienta permitió ahorrar enormes cantidades de tiempo en desarrollo de software. La arquitectura de *LARAVEL*, basado en el modelo vista-controlador, ver figura **4-6**, se acopló perfectamente a las necesidades del proyecto.

Todos los desarrollos se realizaron usando los principios del desarrollo ágil de software, fue-

ron versionados en Git y publicados GitHub. A la aplicación Web desarrollada se le llamo *IotView* y está disponible en `https://github.com/CesarAlvarezG/IoTView`. Al sistema embebido se le nombró IotESp y está disponible en `https://github.com/CesarAlvarezG/IoTEsp`. Las metodologías ágiles fueron protagonistas en el avance de la tesis y se recomiendan para siguientes proyectos de ingeniería.

Para poder llevar a la autonomía al objeto de Internet conformado por una aplicación Web y un sistema embebido se le agregó un Firmware basado en ciclo MAPE-K de la Computación Autónoma. El primer paso para aplicar el ciclo MAPE-K fue elegir el aspecto a convertir en autónomo. El Internet de las Cosas es bastante grande y complejo, de tal forma que, cualquier proceso es candidato para la Computación Autónoma. Se eligió una aplicación de control automático en circuitos eléctricos por ser claro en su desempeño y comportamiento deseado.

En el Firmware del sistema embebido se realizó una serie de Objetos de software, que replicaran la información almacenada por el usuario en la plataforma Web, ver figura **7-19**. Estos objetos se desarrollaron en C++. El ciclo MAPE-K se implementó en el Objeto *MapeKControl*, ver figura **7-21**, el cual se encargó de realizar un lazo cerrado de control en una planta de prueba. Posteriormente, se creó un segundo ciclo MAPE-K en Objeto llamado *MapekPid*, ver figura **7-24**, para administrar el control PID y sus constantes.

El objeto *MapeKControl* se probó en un sistema electrónico de primer orden, ver figura **7-25**. Su desempeño se contrastó con la biblioteca *PIDArduino*. Para permitir una comparación justa el diseño del control PID se realizó en *MATLAB* y los valores de las constantes se introdujeron en ambos controladores. El resultado se puede ver en la figura **7-27**. El sistema de primer orden de la ecuación 7-9 al ser implementado por medio de un circuito RC presenta un comportamiento con pequeñas variaciones en su salida, ver la figura **7-26**, esto a causa del ruido eléctrico presente en el ambiente en el cual se desarrolló la prueba. El controlador PID diseñado en la herramienta *pidTuner* de *MatLab* lleva al sistema al *setpoint* más rápido, según lo diseñado.

En la prueba del control adaptativo se empleó el mismo esquema de la primera prueba de control. Se diseñó el objeto *MapekPid* y aplicó al objeto *MapekControl*. Para verificar que el control fuera realmente adaptativo se cambio la dinámica de la planta de prueba,

agregándole una inductancia y disminuyendo su resistencia. Las constantes del control PID se mantuvieron igual y se le permitió al ciclo *MapekPid* adaptar el control original. Se observó al ciclo MAPE-K ajustarse a la nueva dinámica. Este comportamiento se comparó con el ejemplo adaptativo de la biblioteca *PIDArduino*. En ambos casos se partió de las constantes de control anteriores y se cambio la planta de prueba por la representada por la ecuación 7-12. En la gráfica **7-31** se observa el resumen de los resultados obtenidos durante la adaptación del ciclo MAPEK. Se observa que al usar el PID original es muy grande y aunque el sistema es estable no logra fijarse en un solo punto. En cambio, en la ultima interacción del segundo ciclo MAPEK el sistema logró fijarse en el punto deseado con una muy baja variación. El ciclo MAPEK logró demostrar su efectividad al logra realizar un control sobre un sistema con dinámica desconocida, a partir de un conocimiento previo. El enfoque del ciclo MAPEK es reducir la complejidad de la administración del sistema y esto se pudo verificar al observar como el sistema identifica cuando actuar y soluciona el problema de la configuración de las constantes del control PID. Otro elemento a destacar es la realización de las pruebas en entornos con un alto contenido de ruido eléctrico, que asemeja su funcionamiento a las condiciones industriales y las lleva más allá de las condiciones de laboratorio.

9. Conclusiones

Se realizó un objeto de Internet con la arquitectura propuesta por [33]. En el *networking* se realizó un objeto con conexión inalambrica, con alcance LAN, relación funcional cliente servidor, punto a punto, topología bus, direcionalidad full duplex y con grado de autenticación privada. En *computing* se utilizó el System on Chip ESP32 de la empresa Espressif. Se programó por medio del lenguaje e IDE ARDUINO. El objeto cuenta con 4 entradas análogicas de 12 bits de resolución y 4 salidas analógicas de 8 bits de resolución. Para la acción *Networking* se creó una base de datos en la nube. Para la visualización se desarrolló una plataforma Web por medio del framework Laravel 5.2. La programación del objeto de Internet usó los lenguajes C++, Arduino, PHP, JavaScript, adicionalmente, usó las herramientas HTML5, CSS, JSON, AJAX y Laravel.

Se desarrolló satisfactoriamente el ciclo MAPE-K por medio de Objetos de software en el lenguaje C++. El primer Objeto contó con dos punteros a función para hacerlo versátil en la captura y emisión de las señales eléctricas. El segundo Objeto interactuó con el primero administrándolo sin necesidad de intervención humana. Se cumplió el objetivo de la Computación Autónoma de disminuir la complejidad del administrador y se le traslado al desarrollador. Los dos ciclos MAPE-K se utilizaron para llevar a cabo una estrategia de control automático adaptativo. Un Objeto MAPE-K se empleó para controlar una planta de prueba eléctrica LTI por medio de una estrategia de control PID. El segundo objeto MAPE-K verificó la eficacia del control PID y lo ajustó para mejorar su desempeño. La plataforma Web sirvió como puente entre las órdenes de alto nivel por parte del usuario, e igualmente, la plataforma sirvió como repositorio del conocimiento del control realizado, en la forma de las constantes del PID.

Se aporto al área problemática al desarrollar un *Objeto de Internet* con una clara inclinación al control automático, igualmente, este sistema puede hacer tareas de monitoreo y/o otras típicas de las actuales tendencias en el Internet de las Cosas. Se dio respuesta a la

pregunta de investigación al mostrar el sendero de desarrollo de un Firmware de control automático basado en el ciclo MAPE-K para Objetos de Internet, en el System On Chip ESP32. Posteriormente se comprobó la hipótesis de poder realizar control automático adaptativo en el Internet de las Cosas. El objetivo general se cumplió a cabalidad al lograr los tres objetivos específicos. El primer objetivo especifico se alcanzó con el desarrollo de la aplicación Web *IotView*, el Firmware *IotEsp* y la tarjeta de control *IotControl*. El segundo objetivo específico se logró con el desarrollo de los dos Objetos de software *MapeKControl* y *MapeKPid*. El tercer objetivo específico se cumplió con las pruebas en circuitos eléctricos LTI de primer y segundo orden, respectivamente. Finalmente, se cumplieron con los resultados esperados.

10. Recomendaciones

Para mejorar el *Objeto de Internet* se recomienda emplear el sistema operativo en tiempo real propuesto por la empresa *Espressif* y especialmente diseñado para el ESP32. En cuanto a la plataforma web IotView se aconseja aplicar algoritmos de cifrado punto a punto para asegurar la privacidad de los datos enviados. Para la transmisión de la información se propone enviar los datos por la cabecera de los mensajes HTML y no por la URL, como actualmente se realiza. Igualmente para la aplicación Web se propone seguir mejorando la usabilidad para dar una mejor experiencia al usuario.

Se recomienda para mejorar el ciclo MAPE-K emplear Agentes de software en el lenguaje del sistema operativo nativo del ESP32. Se propone, de igual manera, aumentar el número de ciclos MAPE-K involucrados en la tarea de control automático, estos se pueden encargar de la frecuencia de muestreo, la lectura de voltajes y la escritura de voltajes. Para afinar el desempeño de ciclo MapeKPid se sugiere experimentar con otras técnicas de optimización que garanticen alcanzar el mejor controlador posible. Adicionalmente, se aconseja mejorar las estructuras de control con la aplicación de estrategias avanzadas, como lo son: *Control Adaptativo por modelo de referencia* (MRAC) y *Reguladores auto ajustables* (STR). Para verificar el gran potencial del control por medio del ciclo MAPE-K se invita a realizar pruebas con circuitos eléctricos no lineales y variantes en el tiempo.

Bibliografía

[1] Academy, Cisco N.: *Internet de las Cosas*. www.netacad.com/es/courses/intro-iot/, 2018

[2] Aciem: IoT: Grandes Oportunidades de Negocio. (2016)

[3] Alario Hoyos, Carlos. *Capacidades de Computación Autónoma en el entorno AWSE sobre el estándar WSDM para Servicios Web*. 2008

[4] Amazon: *ESP-WROOM, Módulo ESP32 WIFI + BT+BLE*. https://www.amazon.es/ESP-WROOM-32-M2019

[5] Arango, Laura. *Desarrollo de un módulo de Smart Metering Para el servicio de energía Eléctrica*. 2018

[6] Arcaini, Paolo ; Riccobene, Elvinia ; Scandurra, Patrizia: Formal Design and Verification of Self-Adaptive Systems with Decentralized Control. (2017)

[7] Arduino.cc: *Galileo Intel*. https://www.arduino.cc/en/ArduinoCertified/IntelGalileo, 2018

[8] Arduino.cc: *IoT Shield for Arduino*. https://create.arduino.cc/projecthub/mektrasys/iot-shield-for-arduino-e10ed9, 2018

[9] Atzori, Luigi ; Iera, Antonio ; Morabito, Giacomo. *The Internet of Things: A survey*. 2010

[10] Bandyopadhyay, Debasis ; Sen, Jaydip: Internet of Things - Applications and Challenges in Technology and Standardization. (2011)

[11] Bankinter: *El Internet de las Cosas*. 1. 2011

[12] Barahona Pineda, John Alejandro ; Tejada, César Andrés. *TARJETA SMART HOUSE PARA UNA HABITACIÓN*. 2018

[13] Barrett, Rob ; Maglio, Paul ; Kandogan, Eser ; Bailey, Jhon: Usable Autonomic Computing Systems: The System Administrators perspective. (2004)

[14] Baumann, Michael: Google Joins Movement for Internet of Things. (2010)

[15] Blynk: *The most popular mobile app for the IOT. Works with anything: ESP8266, Arduino, Raspberry Pi, SparkFun and others.* https://www.blynk.cc/, 2019

[16] Calderón Silva, Diego Fernando. *RECONOCIMIENTO DE ROSTROS PARA CONTROL DE ACCESOS*. 2018

[17] Castañeda Bueno, Lorena. *A Reference Architecture for Component-Based Self-Adaptive Software Systems*. 2012

[18] Chen, Yinong: Analyzing and Visual Programming Internet of Things and Autonomous Decentralized Systems. (2016)

[19] ChipKit: *ChipKit Uno32*. https://chipkit.net/wpcproduct/chipkit-uno32/, 2018

[20] Cisco: *Internet de las cosas (IoT)*. https://www.cisco.com/c/esco/solutions/internet-of-things/overview.html, 2018

[21] conti, Juan Pablo: The Internet of Things. (2006)

[22] Culturación: *Topologías de red*. http://culturacion.com/topologias-de-red/, 2017

[23] Dar, Kashif: *INF 5360 spring 2012*. 1. IBM, 2012

[24] Digistump: *Digistump Oak*. http://digistump.com/oak/, 2018

[25] Espressif: *Hardware*. https://www.espressif.com/en/products/hardware, 2018

[26] Espressif: *Software*. https://www.espressif.com/en/products/software, 2019

[27] Evans, Dave. *Internet de las Cosas: Cómo la Próxima Evolución de Internet lo cambia todo*. 2011

[28] Ezquerra, Josema: *Protocolos de red: TCP/IP*. https://laredinfinita.wordpress.com/2014/03/19/protocolos-de-red-tcpip/, 2017

[29] Fiware: *FIWARE: THE OPEN SOURCE PLATFORM FOR OUR SMART DIGITAL FUTURE*. https://www.fiware.org/, 2019

[30] Geisen, Silke. *MAPE-K4SEM - Selbst-adaptive Software - Engineering - Methoden.* 2015

[31] Gil de la Iglesia, Didac: MAPE-K Formal Templates for Self-Adaptive Systems: Specifications and Descriptions. (2014)

[32] Google: *GOOGLE CLOUD IOT.* https://cloud.google.com/solutions/iot/?hl=es, 2018

[33] Gubbi, Jayavardhana ; Buyya, Rajkumar ; Marusic, Slaven ; Palaniswami, Marimuthu: Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. (2013)

[34] Horm, Paul: *Autonomic Computing: IBM's Perspective on the State of Information Technology.* 1. 2001

[35] IBM: *An Architectural Blueprint for Autonomic Computing.* 1. IBM, 2005

[36] IBM: *Internet de las Cosas.* https://www.ibm.com/cloud-computing/bluemix/es/internet-of-things, 2018

[37] Intel: *Introducción a las Placas Intel® Galileo.* https://www.intel.la/content/www/xl/es/support/articles/000005912/boards-and-kits/intel-galileo-boards.html, 2018

[38] Keeley, Tom: IoT to IoAT: Internet of Autonomous Things devices provide solutions. (2016)

[39] Kephart, Jeffrey O. ; Chess, David M.: The Vision of Autonomic Computing. (2003)

[40] Klein, Cornel ; Schmid, Reiner ; Leuxner, Christian ; Sitou, Wassiou ; Spanfelner, Bernd: A Survey of Context Adaptation in Autonomic Computing. (2013)

[41] Kuo, Benjamin: *Sistemas de Control Autómatico.* 1. 1996

[42] Lemaker: *Banana Pro Specification.* http://www.lemaker.org/product-bananapro-specification.html, 2018

[43] Luzuriaga, Jorge ; Cano, Juan Carlos ; Calafate, Carlos ; Manzoni, Pietro ; Perez, Miguel ; Boronat, Pablo: Handling Mobility in IoT applications using the MQTT. (2015)

[44] Maksimović, Mirjana ; Vujović, Vladimir ; Davidović, Nikola ; Milošević, Vladimir ; Perišić, Branko: Raspberry Pi as Internet of Things hardware: Performances and Constraints. (2014)

[45] Martinez, Néstor Lucas. *Development of Fuzzy Control Based Self- Adaptive System for Energy Saving in Constraid Networks for Internet of Things*. 2016

[46] Martín, Alberto N. *Time Machine: Desarrollo de una Herramienta para la Gestión de Configuraciones en Sistemas de Computación Autónoma*. 2011

[47] MediaTek: *Linnklt One*. https://labs.mediatek.com/en/platform/linkit-one, 2018

[48] Micro:Bit: *BBC micro:bit*. http://microbit.org/es/, 2018

[49] Microsoft: *Internet de las Cosas*. https://www.microsoft.com/es-xl/internet-of-things/, 2018

[50] Microsoft: *Azure IoT Hub*. https://azure.microsoft.com/es-es/services/iot-hub/, 2019

[51] Mineraud, Julien ; Mazhelis, Oleksiy ; Su, Xiang ; Tarkoma, Sasu: A Gap Analysis of Internet-of-Things Platforms. (2016)

[52] Miorandi, Daniele ; Sicari, Sabrina ; Pellegrini, Francesco D. ; Chlamtac, Imrich: Internet of Things: Vision, Applications and Research Challenges. (2012)

[53] MIT: *white paper: The Networked Physical World. Proposals for Engineering the Next Generation of Computing, Commerce and Automatic-Identification*. 1. Massachusetts Institute of Technology, 2000

[54] Molines, Salvador I. *Desarrollo de una Herramienta para el Diseño de Reconfiguraciones Seguras en Sistemas de Computación Autónoma*. 2011

[55] Mosquitto: *Eclipse Mosquitt: An open source MQTT broker*. https://mosquitto.org/, 2019

[56] Muñoz Jiménez, José Antonio: *Planificación y Administración de Redes*. https://es.wikibooks.org/wiki/PlanificaciónyAdministracióndeRedes, 2017

[57] Nama, Suneth ; Gamaarachchi, Hasindu ; Lee, Gyu M. ; Um, Tai-Won: Autonomic Trust Management In Cloud-Based And Highly Dynamic Iot Applications. (2015)

[58] Onion: *Omega*. https://onion.io/, 2018

[59] Puonchthrough: *Bean*. https://punchthrough.com/bean, 2018

[60] Rodríguez Rubio, Francisco ; López Sánchez, Manuel Jesús: *Control Adaptativo y Robusto*. 1. 1996

[61] Rutten, Eric ; Marchand, Nicolas ; Simon, Daniel: Feedback Control as MAPE-K loop in Autonomic Computing. (2015)

[62] Sabogal Ardila, Maycol ; Durán Gutierrez, Jhon Aldemar. *LABORATORIO REMOTO DE REDES NEURONALES*. 2019

[63] Sanchez, Antonio Javier G.: *Laravel 5 The PHP Framework For Web Artisans*. 1. 2017

[64] Schoenberger, Chana ; Upbin, Bruce: The Internet of Things. (2002)

[65] Schwab, Klaus: *La Cuarta Revolución Industrial*. 2016

[66] The MathWorks, Inc: *ThingSpeak*. https://thingspeak.com, 2018

[67] Tianfield, Huaglory ; Unland, Rainer: Towards Autonomic Computing System. (2004)

[68] Óscar Torrente: *Arduino Curso práctico de formación*. 2013

[69] Waikato: *WEKA Machine Learning at Waikato University*. https://www.cs.waikato.ac.nz/ml/index.html, 2019

[70] Want, Roy: An Introduction to RFID Technology. (2006)

[71] Weber, Rolf H.: Internet of Things – New Security and Privacy challenges. (2010)

[72] Weber, Rolf H. ; Weber, Romana: *Internet-of-Things-Legal-Perspectives*. 1. 2010

[73] Weiser, Mark: The Computer for the 21st Century. (1991)

[74] wright, Mark ; richey, Rodger: Deeply Embedded Devices: The Internet Of Things. (2009)

A. Anexo: Costo Total de la Investigación

El Costo total para este trabajo se observa en las tablas **A-6**, **A-2**, **A-3**, **A-4**, **A-5**; en ellas se consignan los elementos necesarios para la ejecución de la investigación. La inversión final del proyecto se calcula en $46,880,000.

Tabla A-1.: Costo para el personal del proyecto de Investigación. Fuente: propia.

Rubros	**Valor**
Investigador	$6.000.000
Director del proyecto	$12.000.000
Co - Director	$6.000.000
Asesor	$3.000.000
Total	**$27.000.000**

Tabla A-2.: Costos de equipos del proyecto de Investigación. Fuente: propia.

Rubros	**Valor**
Computador personal	$1.500.000
Objeto de Internet basado en el ESP32	$500.000
Total	**$2.000.000**

Tabla A-3.: Costo para el software del proyecto de Investigación. Fuente: propia.

Rubros	**Valor**
Sistema Operativo	$580.000
Hosting Web (PaaS)	$1.000.000
Repositorio	$200.000
Total	**$3.680.000**

Tabla A-4.: Costo de materiales e insumos del proyecto de Investigación. Fuente: propia.

Rubros	**Valor**
Sistema Embebido basado en ESP32	$1.000.000
Equipos de laboratorio	$2.000.000
Montaje para plantas controlables	$500.000
Servicio de fabricación de PCB para Sistema Embebido	$500.000
Total	**$3.500.000**

Tabla A-5.: Costos de bibliografía en el proyecto de Investigación. Fuente: propia.

Rubros	**Valor**
Libros	$1.500.000
Bases de datos indexadas	$500.000
Cursos virtuales	$1.500.000
Total	**$3.500.000**

Tabla A-6.: Costo general del proyecto de Investigación. Fuente: propia.

Rubros	**Efectivo**	**Recurrente**	**Valor**
Personal		$27.000.000	$27.000.000
Equipos		$2.000.000	$2.000.000
Software		$3.880.000	$3.880.000
Materiales e Insumos	$3.500.000		$3.500.000
Servicios técnicos		$2.000.000	$2.000.000
Bibliografía		$3.500.000	$3.500.000
Viajes	$3.000.000		$3.000.000
Salidas de Campo	$2.000.000		$2.000.000
Total	**$8.500.000**	**$38.380.000**	**$46.880.000**

B. Anexo: Resultados Esperados

Tabla B-1.: Resultados esperados de la investigación. Fuente: propia.

Objetivos	**Resultados / Productos**	**Indicador**	**Beneficiarios**
Desarrollar un *Objeto de Internet* para realizar control automático, usando el System On Chip *ESP32*	Sistema embebido para control automático, usando el ESP32	Diagrama esquemático y archivos Gerber del sistema embebido	Investigador principal, Grupo de investigación Automática.
Desarrollar una estrategia de control automático adaptativo por medio del ciclo MAPE-K	Firmware de control automático por medio del ciclo MAPE-K	Firmware en el repositorio GitHub	Comunidad especializada en IoT
Desarrollar una estrategia de control automático adaptativo por medio del ciclo MAPE-K	Aplicación Web de IoT para monitorear y controlar	Aplicación Web en el repositorio GitHub	Comunidad especializada en IoT
Probar el funcionamiento del Control automático adaptativo por medio del ciclo MAPE-K, en el Internet de las Cosas, usando el System On Chip *ESP32*	Articulo Científico	Sometimiento en revista indexada	Comunidad científica, Grupos de investigación y desarrolladores en el área del IoT

C. Anexo: Impactos Esperados

La presente investigación generará los siguientes impactos, que se observan en la tabla **C-1**. Los plazos en años usados son: Corto (1-4), Mediano (5-9) y Largo (10 o más).

Tabla C-1.: Impactos esperados de la investigación.

Impacto esperado	**Plazo después de finalizado el proyecto**	**Indicador verificable**	**Supuestos**
Académico	Corto	Articulo en revista indexada	El Firmware y la Aplicación Web desarrolladas puede ser usadas en aplicaciones de control en el Internet de las Cosas y es de interés para la comunidad del IoT.
Desarrollo Tecnológico	Mediano	Firmware especializado que implementa el ciclo MAPE-K	El ciclo MAPE-K puede ser usado en el control automático adaptativo.
Desarrollo Tecnológico	Mediano	Aplicación Web especializada en monitoreo y control automático	La aplicación Web puede ser usada en proyectos de monitoreo y control automático.
Económico	Mediano	Empresa de base tecnológica	El IoT genera un mercado económico viable para el desarrollo del país.

D. Anexo: Manual de usuario IotView

IotView es una plataforma de Internet de las Cosas (IoT) para realizar control automático en sistemas dinámicos, redes de sensores, SmartMetering y SmartGrid, entre otros. Esta aplicación se ha desarrollado como resultado de la tesis de Maestría en Ingeniería de César Augusto Álvarez Gaspar, en la Universidad Autónoma de Manizales. Los desarrollos originales son para los System on Chip (SoC) ESP32 y ESP8266. En próximas entregas se espera ampliarla para otros dispositivos del IoT. Esta aplicación web esta creada para un solo proyecto compuesto por múltiples sistemas, cada uno representado por un solo SoC. La gestión de la plataforma recae en el administrador quien crea, actualiza, destruye los sistemas, sensores y otros elementos asociados al proyecto.

D.1. Introducción

El Internet de las Cosas (IoT) es un entorno donde los objetos inanimados se conectan a las personas, datos, procesos y entre sí mismos por intermedio de la Internet, con el objetivo de mejorar la calidad de vida de los hombres y mujeres, en conjunto con la efectividad de los desarrollos industriales [1]. A fin de llevar esto a cabo, se dota a cualquier objeto inanimado de un computador embebido capaz de conectarse al ciberespacio, creando un *Objeto de Internet*. Algunos de estos poseen sensores y actuadores que los dotan de la capacidad de interactuar con su medio ambiente. Son numerosos los aportes propuestos con base en el uso del IoT.

Los *Objetos de Internet* están formados por computadoras pequeñas diseñadas para un único fin, llamados sistemas embebidos. Estos sistemas se componen de una parte material llamada Hardware y una intangible llamada Firmware, equivalente al Software. Una característica distintiva del IoT es la conexión que realiza entre personas, procesos, datos y objetos inanimados [1]. Esta comunicación se puede clasificar como interacciones: Persona

a Persona (P2P), Persona a Maquina (P2M) y finalmente la Maquina a Maquina (M2M). En la figura **D-1** se observan los actores presentes en estos tipos de interacción. En el caso de la P2P se ejemplifica el uso de las redes sociales en dispositivos móviles, por parte de sus usuarios. En la M2P se puede mostrar la interacción de las personas con servidores, electrodomésticos e incluso casas (*SmartHouse*). En la M2M se ilustran los sistemas de climatización en grandes centros comerciales, en estos los termostatos se coordinan con los extractores y ventiladores para mantener las condiciones ambientales agradables para los compradores.

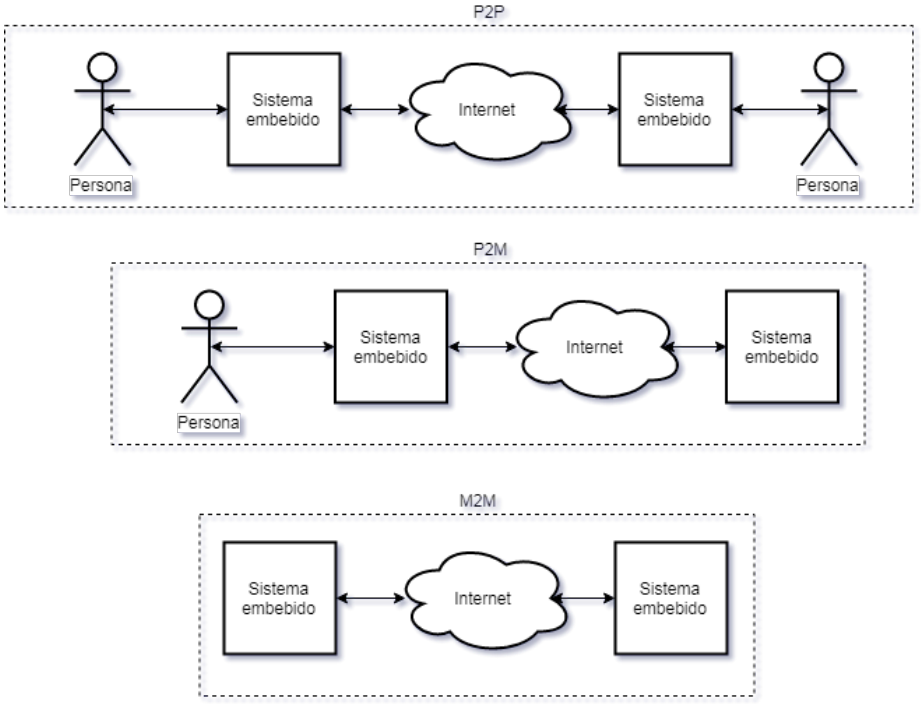

Figura D-1.: Tipos de interacciones presentes en el Internet de las Cosas.

Los sistemas embebidos propuestos para el Internet de las Cosas cumplen funciones de monitoreo por medio de adquisición de datos. Otra habilidad de estos sistemas es el manejo de procesos, por medio del Control Automático, quien ha venido dotando a las máquinas con la capacidad de tomar decisiones en procesos industriales y de consumo. El desarrollo del control automático es bastante grande y cuenta con teorías robustas. Está aplicación Web busca aportar al avance de la implementación del Control Automático en los nuevos desarrollos.

D.2. Descripción de la aplicación

IotView es una aplicación Web que funciona como plataforma del Internet de las Cosas. Esta permite realizar tareas de adquisición y almacenamiento de datos, como medidas de sensores o controladores PIDs usados en procesos automatizados. El sistema se comunica por medio de una Application Programming Interface (API) con los sistemas embebidos conectados a ella. El empaquetamiento de datos se realiza por medio de la notación JavaScript Object Notation (JSON). Para mejorar su seguridad se utiliza un token de autenticación para cada sistema. Los sistemas embebidos a conectar deberán contener el firmware IotEsp que contiene los protocolos de comunicación adecuados, este software pertenece a otro proyecto y podrá descargarse del link `https://github.com/CesarAlvarezG/IoTEsp.git`.

D.3. Guía Instalación

Antes de realizar la instalación de *IotView* deberá tener instalado en su computador las siguientes herramientas:

- PHP, disponible en `https://windows.php.net/download/`
- Composer, disponible en `https://getcomposer.org/download/`
- Git, disponible en `https://git-scm.com/downloads`
- Laravel, disponible en `https://laravel.com/docs/5.8/installation`

IotView debe ser instalado en un servidor para su funcionamiento. Se propone a Heroku para tal fin, al ser una Plataforma como Servicio (PaaS) ofrece grandes funcionalidades. En la figura **D-2** encuentra el link al sitio Web, allí podrá encontrar información adicional. Los siguientes pasos están preparados para usar heroku para instalar *IotView*.

Figura D-2.: Plataforma como Servicio (PaaS)utilizado para instalar *IotView* `https://www.heroku.com/`

El primer paso, en Heroku, es crear una cuenta de usuario. En la figura **D-3** se observa la configuración del registro de la cuenta. Es fundamental seleccionar en *lenguaje de programación* la opción *PHP*.

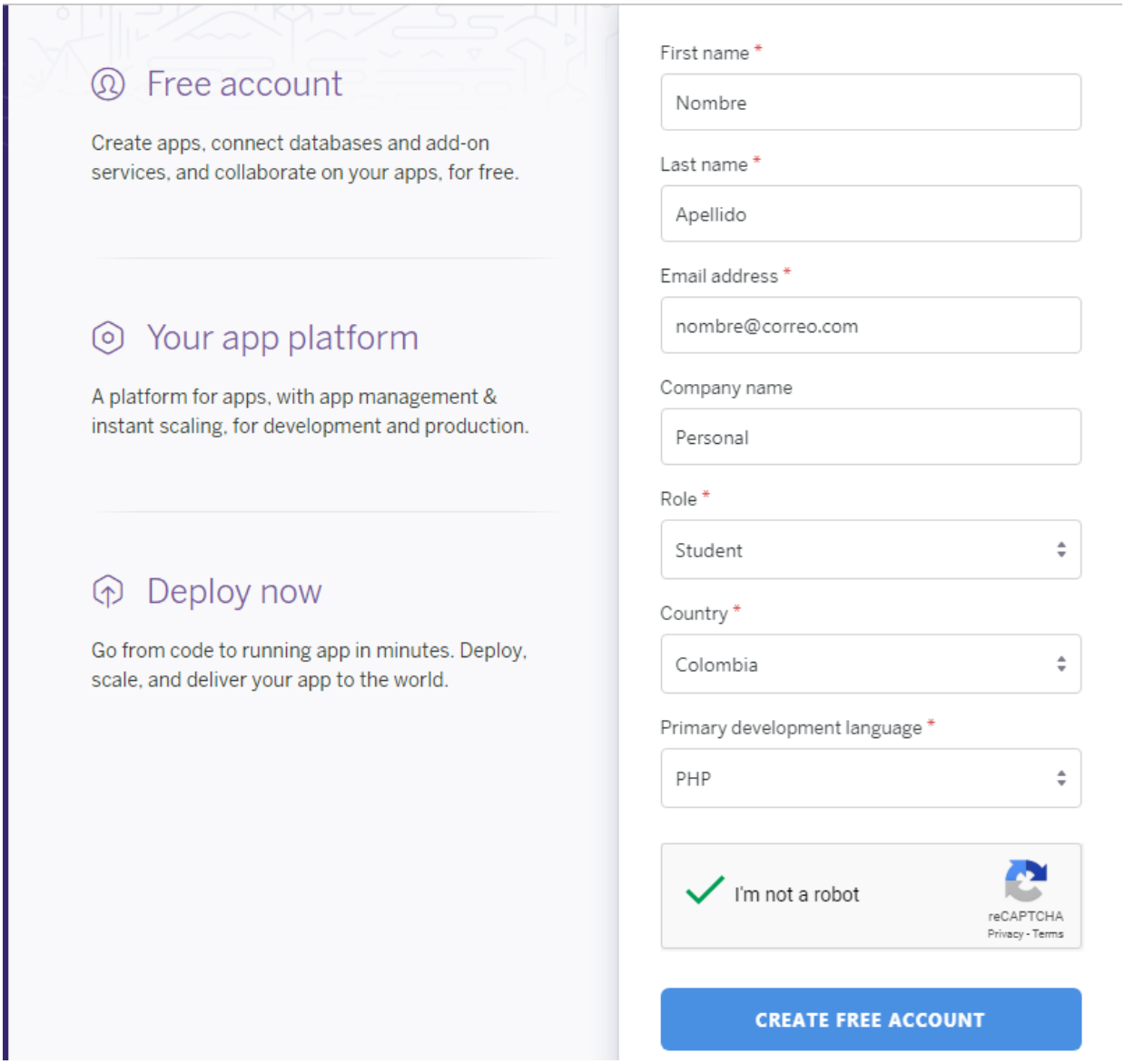

Figura D-3.: Configuración del registro de una nueva cuenta en Heroku

Al correo electrónico ingresado llegará una confirmación, que al ser aceptada lo guiará al ingreso de la contraseña, tal como se observa en la figura **D-4**. La contraseña deberá contener letras, números y símbolos.

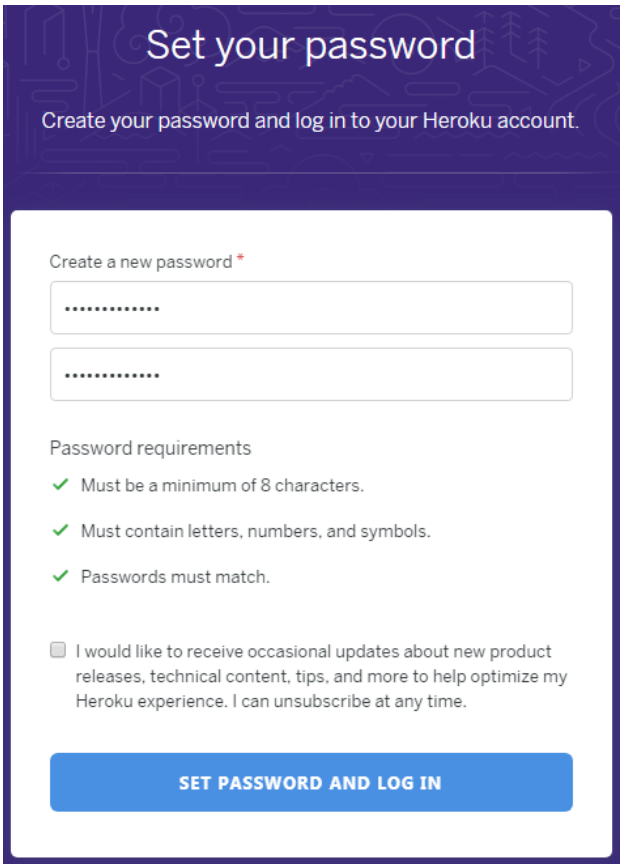

Figura D-4.: Ventana para el ingreso de la contraseña en Heroku

Una vez identificado se accede al panel de creación de nueva aplicación o equipo de trabajo. Seleccionamos la opción *new app*, tal como se observa en la figura **D-5**.

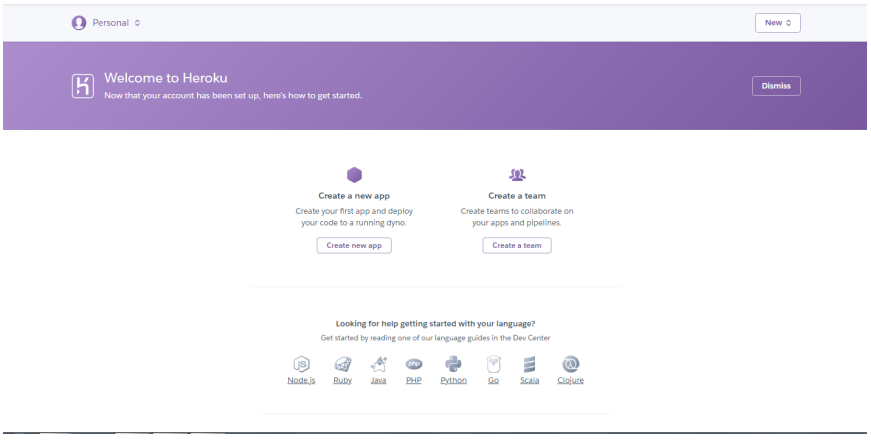

Figura D-5.: Panel de creación de nueva aplicación o equipo de trabajo en Heroku

El siguiente paso es ingresar el nombre del proyecto, como se observa en la figura **D-6**. Es importante seleccionar uno que sea fácil de entender, ya que este será usado como parte del url de la aplicación.

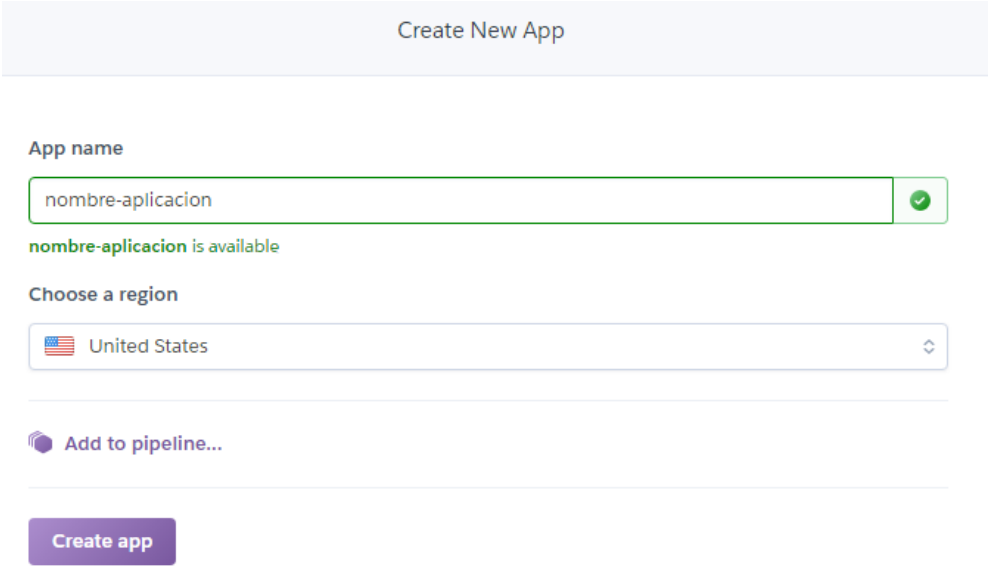

Figura D-6.: Panel de ingreso del nombre de la aplicación en Heroku

Instale la herramienta *Heroku-cli* (figura **D-7**) en su computador, en el siguiente enlace encontrará una guía de cómo hacerlo `https://devcenter.heroku.com/articles/heroku-cli`. Una vez instalados ingrese a la ventana de trabajo de Git, tal como se observa en la figura **D-8**.

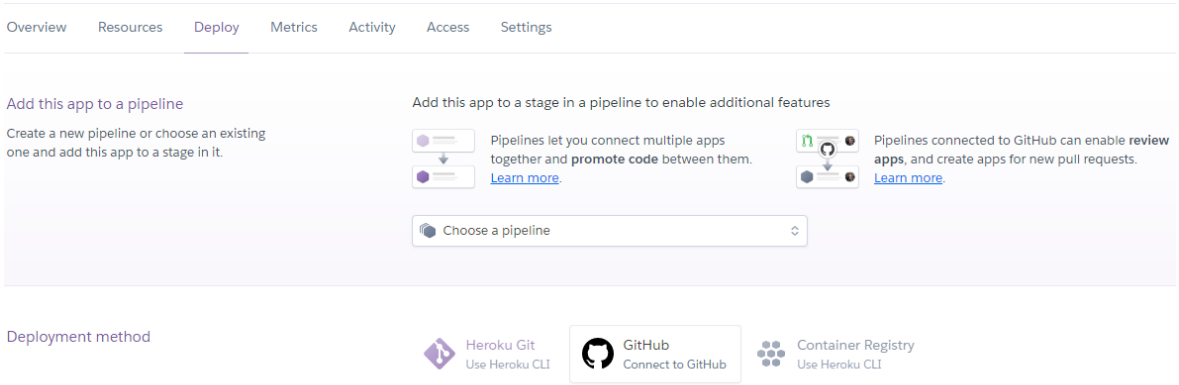

Figura D-7.: Herramientas para usar Heroku

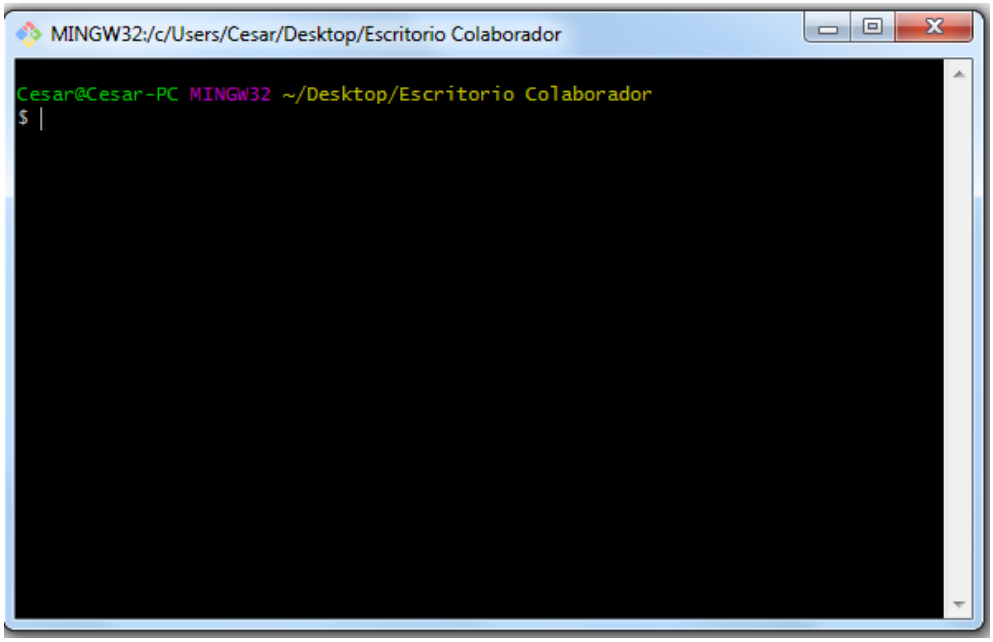

Figura D-8.: Ventana de trabajo de Git

En la ventana de trabajo de Git ingrese los siguientes códigos:

```
heroku login
git clone https://github.com/CesarAlvarezG/IoTView.git
```

En la figura **D-9** se observa el resultado de la descarga.

```
$ git clone https://github.com/CesarAlvarezG/IoTView.git
Cloning into 'IoTView'...
remote: Enumerating objects: 110, done.
remote: Counting objects: 100% (110/110), done.
remote: Compressing objects: 100% (68/68), done.
remote: Total 8811 (delta 61), reused 74 (delta 40), pack-reused 8701
Receiving objects: 100% (8811/8811), 15.73 MiB | 2.31 MiB/s, done.
Resolving deltas: 100% (2084/2084), done.
Checking out files: 100% (6969/6969), done.
```

Figura D-9.: Descarga de IotView en el computador local

Una vez descargado el repositorio de IotView, en el computador local se deben encontrar archivos de la figura **D-10**.

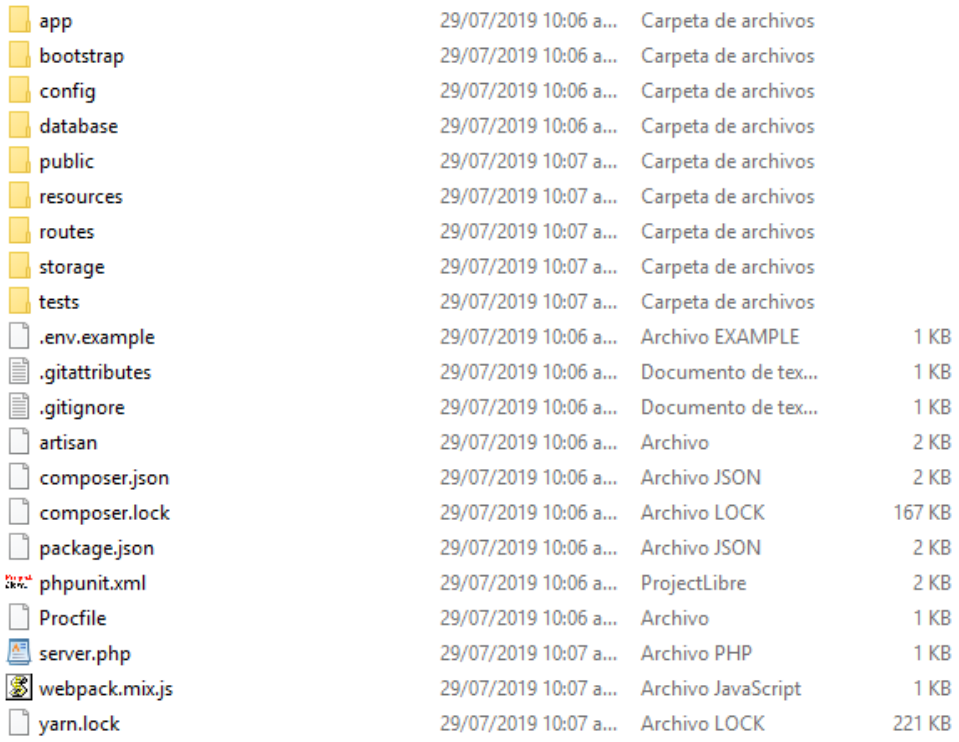

Figura D-10.: Archivos en la carpeta IotView recién creada

Ubíquese la ventana de trabajo de Git en la carpeta IotView recién creada e ingrese las siguientes líneas de código.

```
heroku git:remote -a <nombre-aplicacion>
heroku config:set BUILDPACK_URL=https://github.com/heroku/heroku-buildpack-php
heroku config:set APP_KEY=$(php artisan --no-ansi key:generate --show)
php artisan key:generate --show
```

Al ejecutar la última línea de código deberá aparecer el nuevo valor de la *Api Key*. Este valor debe ingresarlo en lugar de *API Key* en la siguiente línea de código en el espacio *API Key*. Una vez ingresado, continué con las otras.

```
heroku config:set APP_KEY=<API Key>
heroku config:set APP_LOG=errorlog
heroku config:set APP_DEBUG=true
heroku config:set APP_LOG_LEVEL=debug
git push heroku master
```

Si ha realizado correctamente, todos los pasos anteriores, el proyecto ha subido a *Heroku*. Solo resta crear y configurar la base de datos. Las siguientes líneas se deberán realizar paso a paso y prestando atención a cada una.

```
heroku addons:create heroku-postgresql:hobby-dev
```

En la pantalla aparecerá la respuesta de la creación de la base de datos Postgresql. Esta contiene los datos de configuración. La respuesta posee la siguiente estructura:

```
postgres://<username>:<password>@<host>:<port>/<database>
```

Con esta configuración reemplace los datos de *username*, *password*, *host*, *port* y *database* en las siguientes líneas de código:

```
heroku config:set DB_CONNECTION=pgsql
heroku config:set DB_USERNAME=<username>
heroku config:set DB_PASSWORD=<password>
heroku config:set DB_HOST=<host>
heroku config:set DB_PORT=<port>
heroku config:set DB_DATABASE=<database>
```

Ahora, ingrese las últimas líneas de código así:

```
heroku run php artisan migrate
heroku run php db:seed
```

Finalmente, con esto ha concluido el proceso de instalación.

D.4. Guía de uso

Una vez instalado IotView, se ha creado un usuario administrador con todos los permisos para crear, leer, actualizar y borrar; todos los elementos contenidos en la aplicación. Los datos del administrador son los siguientes:

Usuario: admin@coreo.com
Contraseña: 12345678

Estos datos son temporales y se aconseja cambiarlos inmediatamente.

D.4.1. Como iniciar

Para iniciar a usar *IotView* se debe ingresar en el navegador de Internet, de su elección, la dirección asignada durante la instalación. Al entrar en ella se encontrará con la página inicial, ver figura **D-11**.

Figura D-11.: Pantalla principal de la aplicación web IotView

Diríjase a la esquina superior derecha y presione el botón *LOGIN*, una vez dentro introduzca su *Usuario* y *Contraseña*, ver figura **D-12**.

Figura D-12.: Ventana de *LOGIN* en la aplicación web IotView

Una vez identificado podrá pasar la siguiente página. Si el usuario es administrador se redireccionado al *Panel de Control* y si no lo es será direccionado a la *Zona Privada*.

D.4.2. Como usar el panel de control

El *Panel de Control* es el corazón de *IotView*, en el podrá visualizar, crear, actualizar o borrar todos los elementos contenidos en la aplicación web. En la figura **D-13**, se observa su interfaz gráfica, en ella se despliega la información de la totalidad de los elementos contenidos en la aplicación.

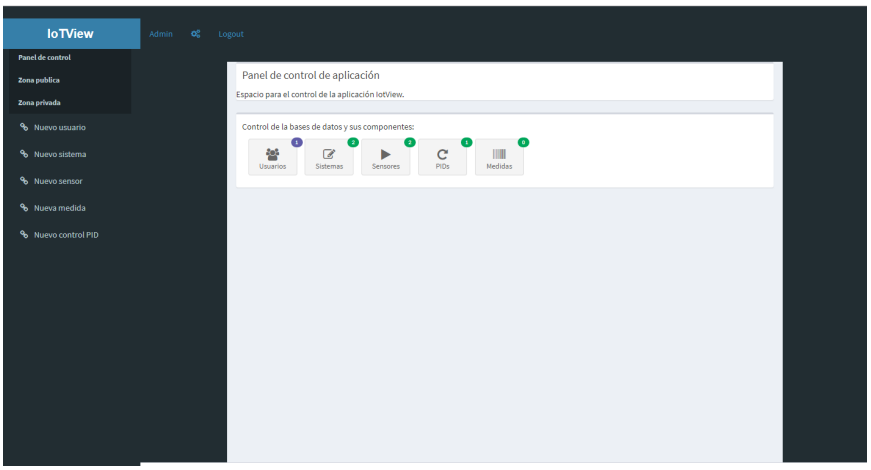

Figura D-13.: Panel de control de la aplicación web IotView

Los elementos bajo el mando del panel de control son *Usuarios*, *Sistemas*, *Sensores*, *Pids* y *Medidas*. Los *Usuarios* son aquellas personas registradas y que cuentan con uno de los tres roles de acceso a IotView. Los roles de acceso son: *Administrador*, *Persona* e *IoTSystem*. Los *Sistemas* son el conjunto de funcionalidades agrupadas bajo un mismo objetivo, generalmente estas se encuentran en un solo sistema embebido. Los *Sensores* son las fuentes generadoras de información y se dividen en dos tipos: el sensor propiamente dicho y el actuador. Su principal diferencia es que el actuador solo muestra su valor actual y no cuenta con *Pids* y/o *Medidas*. Los *Pids* son las configuraciones de los controladores que pueden ser usados por un sistema embebido, en caso de necesitarlo. Finalmente, las *Medidas* son el registro de los datos anteriores generados por un sensor, estos se encuentran desactivados por defecto.

D.4.3. Como manipular los elementos

Al ingresar al elemento de su elección se desplegará una tabla con todos los ítems guardados en la base de datos. En la figura **D-14** se observa la tabla para el caso de los usuarios. En la parte superior encontrar el botón de agregar un nuevo elemento (usuario para este ejemplo). Al lado izquierdo de cada ítem encontrará los botones de *Ver*, *Editar* y *Borrar*.

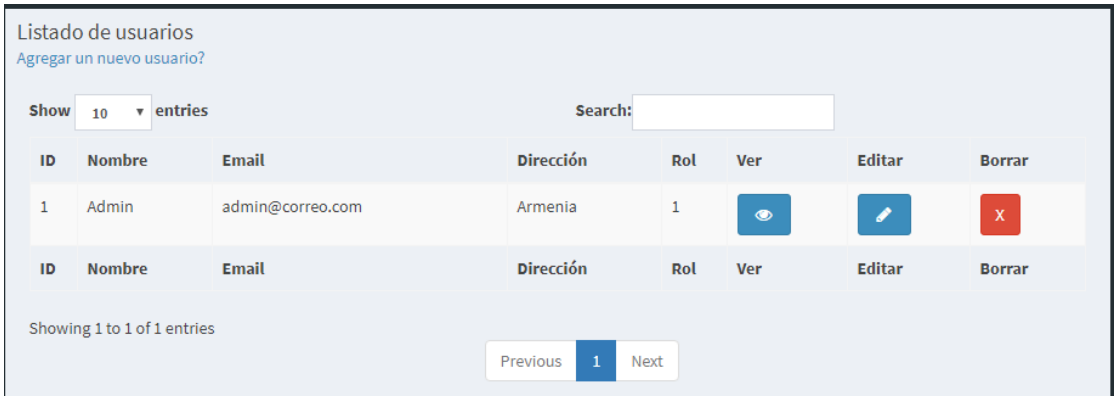

Figura D-14.: Panel de control de los usuarios

En caso de crear un nuevo ítem, aparecerá un formulario con los datos necesarios para tal fin, ver figura **D-15**.

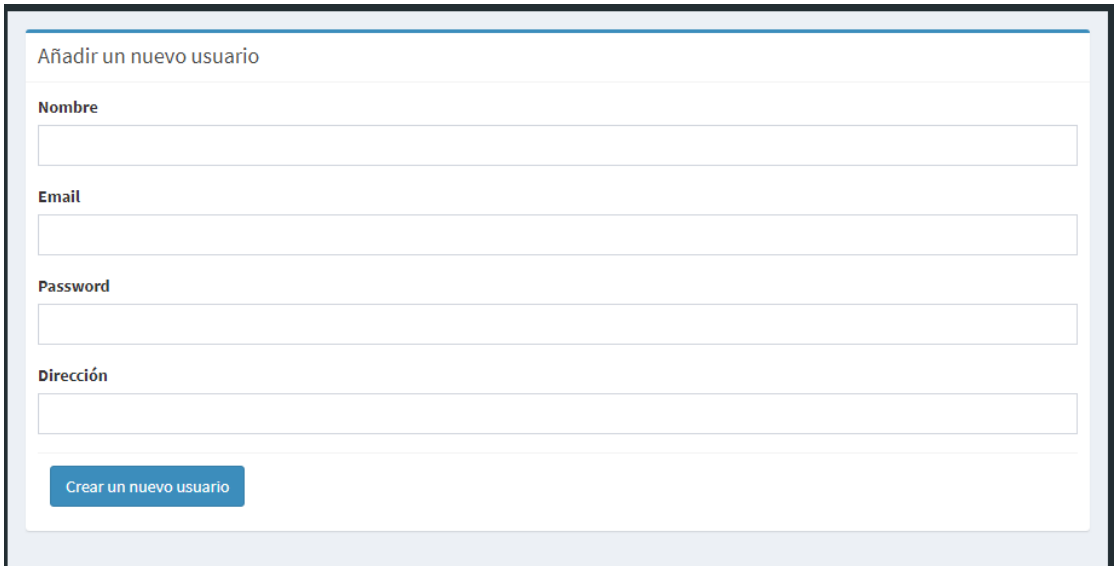

Figura D-15.: Formulario para crear un nuevo usuario

Par el caso de editar, en el formulario aparecerán los datos anteriores y usted contara con la opción de cambiarlos o cancelar la acción, ver figura **D-16**.

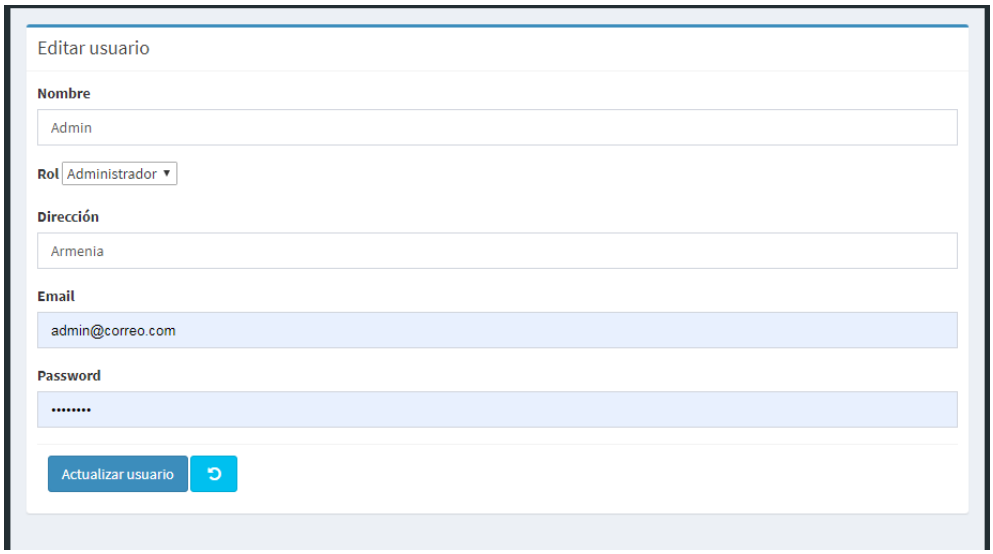

Figura D-16.: Formulario para editar usuario

Para eliminar un ítem solo se requiere con presionar el botón de *Borrar*.

D.4.4. Zona pública

La zona pública es el espacio donde son redireccionados los usuarios no registrados. En esta parte solo aparecen las cantidades de ítems de la aplicación. En la figura **D-17** se muestra la zona pública.

Figura D-17.: Zona pública de la aplicación IotView

D.4.5. Zona privada

La zona privada es el espacio donde son redireccionados los usuarios registrados. En esta parte aparecen las cantidades de ítems de la aplicación y se puede seleccionar un elemento para ver su contenido. En la figura **D-18** se muestra la zona privada.

Figura D-18.: Zona privada de la aplicación IotView

Una vez seleccionado el sistema aparecerá el sistema y sus medidas en tiempo real. Los datos presentados son obtenidos directamente de los sistemas embebidos programados con *IotEsp*. En la figura **D-19** se muestra la visualización de los datos recibidos.

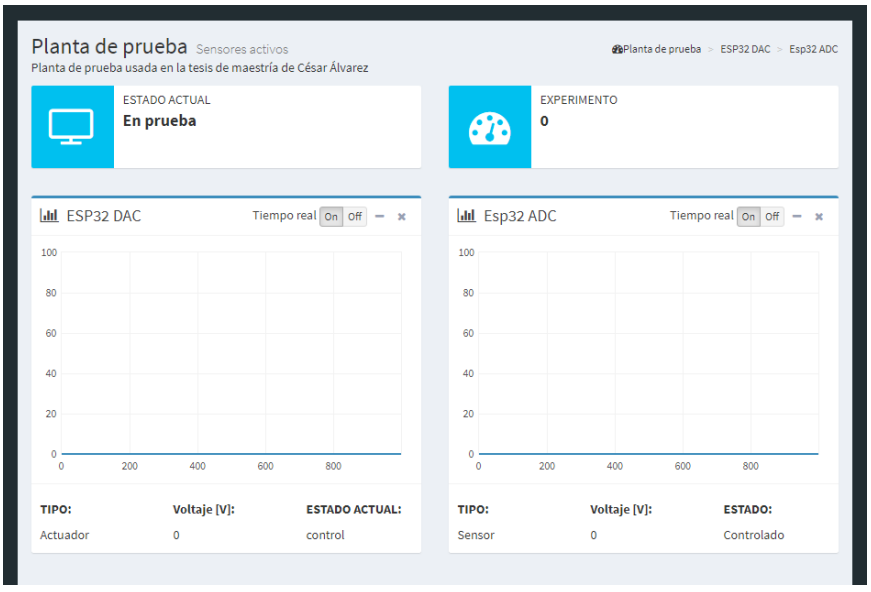

Figura D-19.: Vista de los datos del sistema seleccionado en tiempo real

Cuando el administrador ingresa a la zona privada se le activa una serie de controles adicionales. En la figura **D-20** se observa el detalle de estos controles.

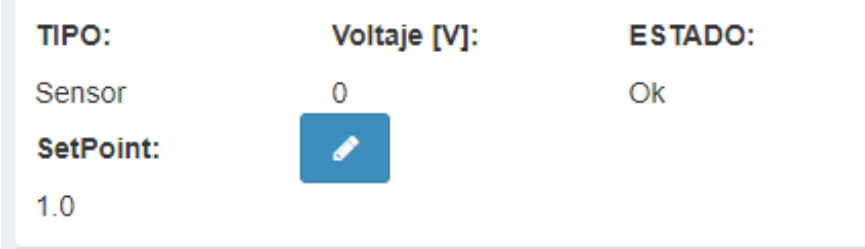

Figura D-20.: Controles exclusivos del administrador en la zona privada

D.5. IotEsp

IotEsp es el Firmware que permite comunicar a los Esp32 con la aplicación *IotView*. Para obtener el Firmware descárguelo del repositorio `https://github.com/CesarAlvarezG/IoTEsp`. En los cuadroa **D-1**, **D-2** y **D-3** se observan el ejemplo del código fuente.

Tabla D-1.: Cabecera del ejemplo de una sola variable del IotEsp a usar la aplicación IotView

```
#include <dummy.h> //Libreria para identificar los pines del EPS32
#include <WiFi.h>//Libreria para el manejo del WiFi
#include "IotView.h"//Libreria para la conexion a la plataforma IotView
#define TAZA_SERIAL 115200  //Velocidad por defecto en el ESP32
#define TAZA_REFRESCO 1000//Taza de refresco del envio
int status = WL_IDLE_STATUS;//Declaracion de los elementos para usar IotView
const int httpPort = 80;
char host[]="iotview.herokuapp.com";
char token[]="RphPq81BeT";
int IDsistema=1;
WiFiClient client; //Declaracion de objetos usados en IotView
TIotView IoTViewSistema(host,IDsistema,token,httpPort,&client);//Declaracion
char ssid[] = "ssid";//Declaracion de las variables para el uso del WiFi
char password[] = "password";
```

Tabla D-2.: Función Setup del ejemplo de una sola variable del IotEsp a usar la aplicación IotView

```
void setup() {
        Serial.begin(TAZA_SERIAL);
        Serial.println("\nPrograma de ejemplo");
        //Inicializacion del WiFi
        WiFi.begin(ssid, password);
        while (WiFi.status() != WL_CONNECTED){
                delay(500);
                Serial.print(".");}
        Serial.println("");
        Serial.println("WiFi connectado");
        Serial.println("Direccion IP: ");
        Serial.println(WiFi.localIP());
        //Inicializacion de la comunicacion a IotView
        Serial.print("connecting to ");
        Serial.println(host);
        IoTViewSistema.GetConfiguracion();
}
```

Tabla D-3.: Función Loop del ejemplo de una sola variable del IotEsp a usar la aplicación IotView

```
int i;
void loop(){
        for(i=0;i<100;i++){
                //Envio a la plataforma IotView
                IoTViewSistema.Sistema.Sensores[0].SetVar(i);
                IoTViewSistema.Push();
                delay(TAZA_REFRESCO);
        }
}
```