Universidad Tecnológica de El Salvador

FACULTAD DE INFORMATICA Y CIENCIAS APLICADAS
CARRERA TECNICO EN INGENIERIA DE REDES COMPUTACIONALES

TEMA:

"IMPLEMENTACION DE UN SISTEMA IOT PARA EL MONITOREO AMBIENTAL DEL LABORATORIO DE DATA CENTER DE LA UNIVERSIDAD TECNOLOGICA DE EL SALVADOR".

TRABAJO PRESENTADO POR:

NELSON VLADIMIR FIGUEROA MENDOZA.

RENE ALEJANDRO QUINTANILLA BONILLA.

OSCAR ENRIQUE LOPEZ GUTIERREZ.

PARA OPTAR AL GRADO DE:

TECNICO EN INGENIERIA DE REDES COMPUTACIONALES.

SEPTIEMBRE, 2019

SAN SALVADOR, EL SALVADOR, CENTROAMERICA

AUTORIDADES ACADÉMICAS

ING. NELSON ZÁRATE SÁNCHEZ

RECTOR

LIC. JOSÉ MODESTO VENTURA ROMERO

VICERRECTOR ACADÉMICO

ING.FRANCISCO ARMANDO ZEPEDA

DECANO

JURADO EXAMINADOR

ING. RONY ADALBERTO CORTEZ

PRESIDENTE

ING. ANGEL ALFREDO CALDERON RODRIGUEZ

PRIMER VOCAL

ING. OMAR OTONIEL FLORES CORTEZ

SEGUNDO VOCAL

SEPTIEMBRE, 2019

SAN SALVADOR, EL SALVADOR, CENTRO AMERICA

Universidad Tecnológica de El Salvador

ACTA DE EXAMEN PROFESIONAL

HABIÉNDOSE REUNIDO EL JURADO CALIFICADOR INTEGRADO POR:
Ing.Angel Alfredo Calderón Rodríguez, Ing.Omar Otoniel Flores Cortez, Ing.Rony Adalberto Cortez, a las 12:00m del día Martes, dieciocho de Junio de dos mil diecinueve.

Y LUEGO DE HABER DELIBERADO SOBRE EL EXAMEN PROFESIONAL DE LOS ALUMNOS:

1- Oscar Enrique López Gutiérrez Carnet 26-0980-2017
2- René Alejandro Quintanilla Bonilla Carnet 26-0044-2017
3- Nelson Vladimir Figueroa Mendoza Carnet 26-1793-2017

QUIENES PRESENTARON DEFENSA DE SU TRABAJO DE GRADUACION TITULADO:
"Implementación de sistema IoT para el monitoreo ambiental del laboratorio de Data Center de la Universidad Tecnológica de El Salvador"

PARA OPTAR AL GRADO DE:

TÉCNICO EN INGENIERIA DE REDES COMPUTACIONALES

Y DEL CUAL TAMBIEN EVALUARON LOS CONOCIMIENTOS RELACIONADOS CON EL TEMA DEL MISMO. POR LO QUE ESTE JURADO RESUELVE DECLARAR EL EXAMEN COMO:

APROBADO.

YA QUE CUMPLE CON LOS REQUISITOS ESTABLECIDOS EN EL REGLAMENTO DE GRADUACION DE LA UNIVERSIDAD.

San Salvador, dieciocho de junio de dos mil diecinueve.

F._____
PRIMER VOCAL
Ing.Angel Alfredo Calderón Rodríguez

F._____
SEGUNDO VOCAL
Ing.Omar Otoniel Flores Cortez

F._____
PRESIDENTE
Ing.Rony Adalberto Cortez

Calle Arce No. 1020, San Salvador, El Salvador, C.A. Apdo. Postal 1770
Centro de llamadas: (503) 2275-8888, Fax: (503) 2271-4764
Email: contactanos@utec.edu.sv

Índice

Introducción

El internet de las cosas (en inglés Internet of Things o IoT), es un concepto que se refiere a la interconexión digital de objetos cotidianos con internet. Con esta tecnología es posible aplicarlos en el área empresarial con el objetivo de monitorear datos en tiempo real. Con los avances tecnológicos actuales es posible tener acceso a internet desde casi cualquier parte.

Un data center también llamado centro de datos o centro de procesos de datos (CPD) es una instalación para albergar un sistema de información de componentes asociados, como telecomunicaciones y los sistemas de almacenamientos donde generalmente incluyen fuentes de alimentación redundante o de respaldo de un proyecto típico de data center que ofrece espacio para hardware en un ambiente controlado. Dentro del Data Center, mantener la temperatura adecuada de forma estabilizada y controlada es una pauta fundamental del control ambiental, permitiendo el establecimiento y ejecución de una política claramente definida que contribuya a tener un Data Center robusto, confiable y durable.

El rango de temperatura óptimo para un Data Center es entre 17 °C y 21 °C. Es necesario aclarar que esa temperatura no es de carácter obligatorio e inamovible, sino que existe también un margen aceptable de operación que sería de 15 °C y 25 °C.

Cualquier temperatura mayor a 25 °C deberá ser corregida de manera inmediata, ya que implica poner en riesgo el equipamiento del Data Center.

Este trabajo de investigación va orientado a utilizar las tecnologías IoT en la implementación de un sistema de monitoreo ambiental de un Centro de Datos que nos permita mostrar las variaciones de temperatura en tiempo real para que los administradores del Centro de Datos puedan tener un mejor control del comportamiento del ambiente.

Capitulo I. Planteamiento del Problema.

1.1 Antecedentes

El 11 de noviembre de 2017 fue inaugurado el Laboratorio de Data Center en la primera planta del edificio Simón Bolívar ubicado frente a la calle Arce de la Universidad Tecnológica de El Salvador. Este laboratorio está enfocado en gestión y manejo de base de datos. El laboratorio cuenta con equipo de red especializado integrado por 15 servidores marca DELL, una red SAN y una red NAS; un UPS y equipos de conectividad.

Los equipos de red del laboratorio de Data Center (ver figura 1 y figura 2) funcionan durante las 24 horas del día durante 365 días en el año, por lo que es importante que se mantengan en una condición ambiental acorde a las condiciones sugeridas por los fabricantes. De lo contrario, si no se encuentran en un ambiente controlado, especialmente temperatura, podrían tener fallas como: problemas de rendimiento, cortocircuito, sobrecalentamiento; entre otros, que afectarían la integridad física y desempeño de los equipos.

Por esta razón es importante mantener un monitoreo constante del ambiente en el que están expuestos los equipos.

Figura 1 Vista del equipo de red del laboratorio de Data Center (fuente propia)

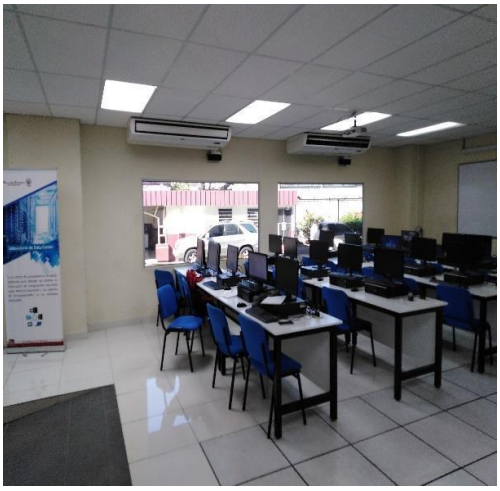


Figura 2 vista del equipo informático disponible en el Laboratorio de Data Center (fuente propia)

1.2 *Definición del Problema.*

¿Es posible la aplicación de técnicas de IoT en el diseño e implementación de un sistema de monitoreo ambiental para el laboratorio de Data Center de la Universidad Tecnológica de El Salvador?

1.3 Objetivo General.

Implementar un sistema de IoT para el monitoreo ambiental del laboratorio de Data Center de la Universidad Tecnológica de El Salvador.

1.3.1 Objetivos específicos.

- Aplicar técnicas de IoT en el diseño de un sistema electrónico con sensores para el monitoreo ambiental de temperatura y humedad al que se exponen los equipos

de red del laboratorio de Data Center de la Universidad Tecnológica de El Salvador.

- Construir un circuito electrónico basado en un sistema microcontrolador que sea capaz de registrar temperaturas en tiempo real y sus variaciones. Además, que permita enviar estos datos a través de una red local inalámbrica.
- Utilizar los servicios de Google Suite apps en la configuración de una plataforma en la nube para tener acceso a los monitoréos de temperatura desde cualquier equipo conectado a internet.

1.4 Justificación.

Un Data Center sin una buena climatización podría afectar el rendimiento de los servicios y otros equipos, por eso es importante tener monitorizado el ambiente en tiempo real y de manera remota tomando en cuenta que esta tecnología no existe en ninguno de los ocho laboratorios con equipos informáticos de la Universidad Tecnológica de El Salvador.

1.5 Limitaciones.

Limitación científica:

El proyecto está limitado al estudio y la aplicación de técnicas IoT, uso de dispositivos electrónicos y de naturaleza Open-source enfocado en el desarrollo de herramientas de bajo costo.

Limitación temporal:

Se plantea que este estudio se desarrollara a lo largo de ciclo 1 del año 2019, entre los meses de febrero a junio.

Limitación espacial:

La implementación del proyecto estará restringida al área comprendida por las instalaciones del laboratorio de Data Center de la Universidad Tecnológica de El Salvador ubicado en la primera planta del edificio Simón Bolívar.

Capitulo II. Marco Teórico.

2.1 Ciudades Inteligentes.

El concepto Smart City surge de la evolución de las llamadas: Ciudad inteligente o Ciudades Digitales que en el año 2004 nacen en España tras un trabajo que realizó el Ministerio de Industria de este país con la elaboración del primer programa de Ciudades Digitales que se abordaba en el mundo. Previo a la elaboración de este trabajo, la empresa española ACCEDA presidida por Enrique Ruz Bentué reunió, por primera vez en la historia, a más de 30 empresas de diversa procedencia sectorial (telecomunicaciones, seguridad, construcción, audiovisual, electrónica de consumo, material eléctrico, informática, salud, educación, etc), junto a gobiernos de regiones y ciudades españolas, para crear Comunidad Digital. El resultado de esa reunión multisectorial llevó a la presentación efímera de una ciudad de 5.000 m2 que incluía viviendas, un banco, hospital, hotel, oficina de tributación, correos, oficinas de gobierno, escuela y todo en un entorno urbano con alumbrado público, semáforos, mobiliario urbano y todo lo que conformaría una ciudad verdadera, pero en una presentación de formato cinematográfico. En Comunidad Digital, Enrique Ruz presentó de la mano de empresas como ZTE, Telefónica, Siemens, Gas Natural, Prosegur, Berker, INDRA, RACE y otras, la primera Ciudad Digital; que años más tarde IBM bautizaría como Smart City.

Ciudades inteligentes, dado su origen natural de las Ciudades Digitales, se basa en el uso intenso de las Tecnologías de la Información y Comunicación (TIC) en prestación

de servicios públicos de alta calidad y calidez, seguridad, productividad, competitividad, innovación, emprendimiento, participación, formación y capacitación.

Para Rudolf Giffinger, las "ciudades inteligentes" pueden ser identificadas y clasificadas, según seis criterios principales o dimensiones principales, y dichos criterios son:

- economía
- movilidad
- medioambiente
- habitantes
- forma de vida
- administración

Estos seis criterios o aspectos se conectan con las tradicionales teorías regionales y neoclásicas del crecimiento y desarrollo urbano, y respectivamente están basados en la teoría de la competitividad regional, en la economía de los transportes y de las tecnologías de la información y de la comunicación, en los recursos naturales, en el capital humano y social, en la calidad de vida, y en la participación de los ciudadanos en la vida democrática de la ciudad. (Sur, 2016)

Figura 3 Ciudad inteligente Fujisawa, Japón (TeleSur, 2016

El consultor del Banco Interamericano de Desarrollo, Mauricio Bouskela, destaca los siguientes rasgos de las ciudades inteligentes:

- Gestión racional del espacio urbano y los recursos naturales.
- Empleo de fuentes alternativas de energía y reducción de emisiones de CO2.
- Uso de redes de comunicación, sensores y sistemas inteligentes.
- Manejo de grandes bases de datos para prever o mitigar problemas.
- Aprovechamiento de herramientas digitales y plataformas interactivas.
- Conexión del gobierno y los ciudadanos y realización de trámites por Internet.
- Generación de nuevos servicios y empresas de base tecnológica. Ciudades sostenibles.

Alguna de las medidas para la sostenibilidad que propone el concepto de ciudad inteligente es la utilización de paneles fotovoltaicos en las comunidades, un mayor uso de medios de trasporte y vehículos eléctricos, paneles solares para semáforos o señales, molinos eólicos en farolas, así como la promoción y desarrollo del uso de las bicicletas.

El desarrollo de sistemas de gestión integral del ciclo del agua permite la reutilización del agua a fin de aprovechar al máximo el valioso y limitado recurso.

Figura 4 Los paneles solares captan la energía de la radiación solar para su aprovechamiento. (TeleSur, 2016)

¿Dónde hay ciudades inteligentes?

Encabezando el podio de las ciudades más inteligentes están Tokio (Japón), Londres (Reino Unido), Nueva York (Estados Unidos), Zúrich (Suiza) y París (Francia), según el índice IESE Cities in Motion (ICIM).

América Latina también tiene ciudades inteligentes: Santiago de Chile (Chile), Buenos Aires (Argentina), Monterrey (México), Ciudad de México (México), y

Bogotá (Colombia), según el Centro de Globalización y Estrategia del Instituto de Estudios Superiores de la Empresa (IESE).

Si para el año 2050 no se han tomado medidas drásticas que contrarresten el crecimiento demográfico, se espera un impacto medioambiental y social sin precedentes.

Por ello, es necesario la búsqueda de un modelo que permita la sostenibilidad del medio ambiente en el futuro. Las ciudades inteligentes son la propuesta, refiere la Organización para la Cooperación y el Desarrollo Económico (OCDE).

Ventajas de las ciudades inteligentes.

El internet de las cosas (IoT), el big data, aplicaciones móviles, industria 4.0… están consiguiendo mejorar la eficiencia de las ciudades, si sabemos utilizarlo de manera inteligente. En este sentido, una ciudad puede gestionar la tecnología para mejorar la vida de las personas y más concretamente, para conseguir beneficios como:

1. Contribuir a la mejora del medio ambiente.
2. Ahorrar costes a sus ciudadanos.
3. Optimizar los servicios públicos.
4. Mejorar la transparencia en la gestión de las administraciones.
5. Conseguir retener empresas y atraer talento.

Para que cualquier municipio se considere una ciudad inteligente, debe reunir estas condiciones:

1. Desarrollo económico, social y medioambiental sostenible y en armonía.

2. Gestión óptima de los recursos naturales a través de la participación de los ciudadanos.

3. Ciudadanos e instituciones comprometidas con el fin.

4. Infraestructuras e instituciones dotadas de soluciones tecnológicas para hacer la vida de los ciudadanos más sencilla.

5. Pero la pieza clave para el funcionamiento de una ciudad inteligente es la participación ciudadana. Si los ciudadanos no contribuyen activamente el uso y fomento de estas alternativas, no se conseguirán los objetivos previstos en su implantación. Es esencial la información, formación y difusión a los ciudadanos por parte de las administraciones públicas. (Sur, 2016)

2.2 Internet de las Cosas.

Internet de las cosas se refiere a la interconexión digital de objetos cotidianos con internet que pueden ser vehículos, Electrodomésticos, maquinas, otros. Con el objetivo de ofrecer datos en tiempo real. El concepto de internet de las cosas fue propuesto en 1999, por Kevin Ashton, en el Auto-ID Center del MIT, en donde se realizaban investigaciones en el campo de la identificación por radiofrecuencia en red (RFID) y tecnologías de sensores. (Leguízamo, 2018)

El concepto de combinar computadoras, sensores y redes para monitorear y controlar diferentes dispositivos ha existido durante décadas. Sin embargo, la reciente

confluencia de diferentes tendencias del mercado tecnológico está permitiendo que la Internet de las Cosas esté cada vez más cerca de ser una realidad generalizada. Estas tendencias incluyen la conectividad omnipresente, la adopción generalizada de redes basadas en el protocolo IP, la economía en la capacidad de cómputo, la miniaturización, los avances en el análisis de datos y el surgimiento de la computación en la nube. (Karen Rose, 2015)

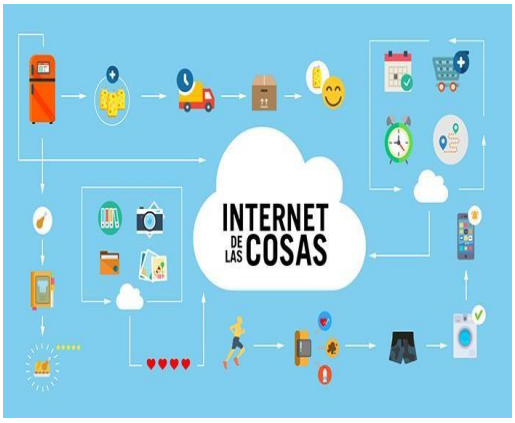


Figura 5 varios dispositivos conectados a La Nube (fuente de internet) (Bymovil, 2018)

Hoy día tenemos la posibilidad de encontrar conexión a internet en cualquier parte. De la misma manera, la tecnología para realizar esas conexiones ha mejorado mucho también. Eso significa que es posible hacer que casi cualquier cosa se conecte a internet. Y esa es la filosofía del Internet de las Cosas, conectar a la red aparatos que, hasta ahora, no lo estaban.

Básicamente, si algo tiene un interruptor, se puede conectar, y la idea es que, si algo se puede conectar a la red, se conecta a la red. Así, no solo nuestros ordenadores

y móviles lo estarán, sino también la nevera, la alarma que ponemos por las mañanas, o hasta las luces de las habitaciones. El futuro tiende cada vez más al Internet de las Cosas. (Lowi, 2018)

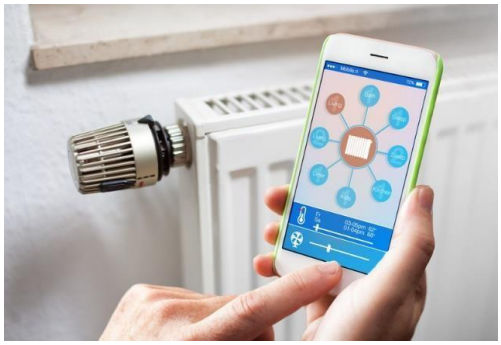


Figura 6 Interfaz de los dispositivos conectados (fuente de internet) (Lowi, 2018)

2.2.1 Arquitectura IOT.

La arquitectura de la internet de las cosas tiene que cumplir ciertos requerimientos para que esta tecnología sea viable debe permitir que la tecnología sea distribuida donde los objetos puedan interactuar entre ellos, escalable, flexible, robusta, eficiente y segura.

Requerimientos:

- Conectividad y comunicación.
- Gestión y control de dispositivos.
- La posibilidad de desconectar un dispositivo robado.
- La habilidad de actualizar el software de un dispositivo.
- La actualización de credenciales de seguridad.

- Autorizar o denegar algunas capacidades del hardware remotamente.
- Localizar dispositivos perdidos.
- Limpiar información confidencial de un dispositivo robado.
- Reconfigurar parámetros de Wi-Fi, GPRS u otras redes remotamente.
- Recolección, análisis y actuación de los datos.
- Escalabilidad.
- Flexibilidad.
- Alta disponibilidad.
- Integración.
- Seguridad.
- Riesgos inherentes de cualquier sistema de internet pero que los diseñadores IoT o de producto no tengan consciencia de ellos.
- Riesgos específicos de los dispositivos IoT.
- Seguridad para cerciorarnos de que no se causan daños por, por ejemplo, por el mal uso de los actuadores.

Figura 7 Arquitectura general IOT (fuente de internet) (Hernández, 2016)

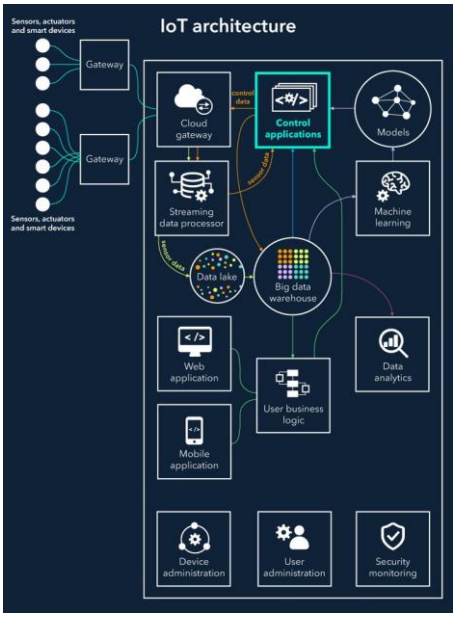


Figura 8 Esquema de los componentes de la arquitectura IoT (Fuente de internet) (Crespo, s.f.)

La arquitectura de IOT contiene los siguientes componentes:

- **Cosas** equipadas con sensores para recoger datos y **actuadores** para realizar comandos recibidos desde la nube.

- **Gateway** para filtrar, preprocesar y mover datos a la nube y viceversa, recibir comandos desde la nube.

- **Pasarelas en nube (Cloud Gateway)** para garantizar la transición de datos entre las pasarelas sobre el terreno y los servidores centrales de IoT.

- **Procesadores de datos en tiempo real** para distribuir los datos procedentes de los sensores entre los componentes de las soluciones de IoT pertinentes.

- **Bases de Datos** para almacenar todos los datos de valor definido e indefinido.

- **Big Data Warehouse** para la recogida de datos valiosos.

- **Aplicaciones de control** para enviar comandos a los actuadores.

- **Machine Learning** para generar los modelos que luego son utilizados por las aplicaciones de control.

- **Aplicaciones de usuario** que permiten a los usuarios monitorizar el control de sus cosas conectadas.

- **Análisis de datos** para el procesamiento manual de datos.

- Los dispositivos del IoT siguen un proceso por el cual la información fluye del medio físico a un medio virtual. Este proceso lo podemos dividir en cuatro fases según la arquitectura propuesta.

- Cosas, objetos y dispositivos conectados.

- Esta es la parte visible a los usuarios. Dentro de este bloque encontramos los sensores, actuadores y el hardware necesario para comunicar el mundo físico con el mundo virtual, Marcas como Intel, Arduino, Raspberry Pi, Qualcomm, AMD, ARM,

Microchip, etc.…, ponen a nuestra disposición placas para utilizar en nuestros desarrollos. El primer problema lo encontramos aquí, ¿cómo comunicar diferentes dispositivos de diferentes marcas? Cada fabricante utiliza su propio software, hardware y protocolo. Esto es un cuello de botella ya que no existe un estándar para la comunicación entre ellos.

- Puntos de acceso.

Los puntos de acceso permiten la conectividad de las cosas, objetos o dispositivos a Internet. El objetivo fundamental es establecer una conexión entre los periféricos (dispositivos u objetos conectados) y la nube, pero también debe permitir conectividad entre ellos. Esta conexión tiene que ser segura, robusta, tolerante a fallos a fin de que recoja la información obtenida de los dispositivos y a la vez, se puedan gestionar. A través de los Gateway o interfaces de comunicación, los dispositivos estarán conectados entre ellos y con la nube.

- Procesamiento de datos.

El eje central del IoT son los datos. El buen funcionamiento de un sistema con estas características dependerá de las capacidades en la gestión de estos datos y el uso inteligente que se haga de ellos. Por este motivo, un sistema del IoT debe ser capaz de recolectar información de los sensores, almacenarlos y analizarlos. En este punto las plataformas en la nube enfocadas a este sector tienen mucho que decir.

Además, deben ser capaces con los datos analizados lanzar alertas basadas en reglas. Un caso típico, que se suele poner como ejemplo, es el sensor de temperatura y

el sistema de calefacción de una casa. Este sensor es capaz de medir una magnitud física y enviar la información a través de un protocolo a algún centro de procesamiento de datos.

- Aplicaciones.

Para poder manejar y visualizar la información, necesitamos de aplicaciones que sean amigables para el ser humano. Da lo mismo si son nativas o web. Gracias al uso de APIs y servicios web, cualquier tipo de aplicación se podrá conectar a los datos y mostrarlos a los usuarios. Y no solo vamos a visualizar los datos. Estas aplicaciones tendrán la capacidad de modificar los parámetros para que los sistemas se comporten de una manera determinada. (Crespo, s.f.)

2.3 Hardware para IoT.

2.3.1 Microcontrolador ESP32.

ESP32 es una serie de sistemas de bajo consumo y bajo consumo en microcontroladores de chip con Wi-Fi integrado y Bluetooth de modo dual. La serie ESP32 emplea un microprocesador Tensilica Xtensa LX6 con variaciones de doble núcleo y de núcleo único e incluye interruptores de antena incorporados, balun RF, amplificador de potencia, amplificador de recepción con bajo nivel de ruido, filtros y módulos de administración de energía. ESP32 es creado y desarrollado por Espressif Systems, una compañía china con sede en Shanghai, y es fabricado por TSMC utilizando su proceso de 40 nm. [2] Es un sucesor del ESP8266. Microcontrolador. (Wikimedia Fundation, s.f.)

Entre las características más destacadas incluye las siguientes:

- Procesadores.
- CPU: Xtensa microprocesador LX6 de doble núcleo (o de un solo núcleo) de 32 bits, que funciona a 160 o 240 MHz y tiene un rendimiento de hasta 600 DMIPS.
- Coprocesador de potencia ultra baja (ULP).
- Memoria: 520 KiB SRAM.
- Conectividad inalámbrica.
- Wi-Fi: 802.11 b / g / n.
- Bluetooth: v4.2 BR / EDR y BLE.

Interfaces periféricas:

- ADC SAR de 12 bits hasta 18 canales.
- DAC de 2 × 8 bits.
- 10 sensores táctiles (GPIOs de detección capacitiva).

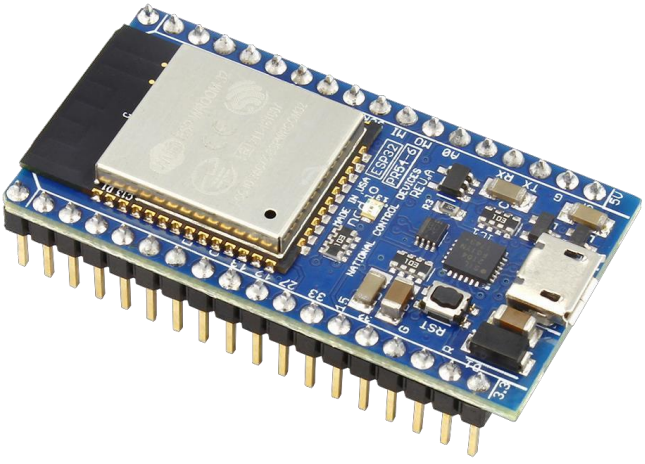


Figura 9 Microcontrolador ESP32 (NCD, s.f.).

2.3.2 Sensor de temperatura y humedad relativa DHT21 (AM2301).

El DHT21 es un sensor digital de temperatura y humedad relativa de buena precisión en un empaque robusto. Integra un sensor capacitivo de humedad, un termistor y un microcontrolador encargado de realizar la conversión analógica a digital. Su empaque de plástico es más robusto comparado a los sensores DHT11 y DHT22, esto hace del DHT21 un sensor ideal para aplicaciones en exteriores. Utilizado en aplicaciones de control automático de temperatura, aire acondicionado, monitoreo ambiental en agricultura y más.

Utilizar el sensor DHT21 con las plataformas Arduino/Raspberry Pi/Nodemcu es muy sencillo tanto a nivel de software como hardware. A nivel de software se dispone de librerías para Arduino con soporte para el protocolo "Single bus". En cuanto al hardware, solo es necesario conectar el cable de color Rojo a la fuente de alimentación 3-5V, el cable Negro a Tierra (0V) y el cable Amarillo a un pin digital en nuestro Arduino. Si se desea conectar varios sensores DHT21 a un mismo Arduino, cada sensor debe tener su propio pin de datos. Quizá la única desventaja del sensor es que sólo se puede obtener nuevos datos cada 2 segundos. Cada sensor es calibrado en fábrica para obtener unos coeficientes de calibración grabados en su memoria OTP, asegurando alta estabilidad y fiabilidad a lo largo del tiempo. El protocolo de comunicación entre el sensor y el microcontrolador emplea un único hilo o cable, la distancia máxima recomendable de longitud de cable es de 20m., de preferencia utilizar cable apantallado. Proteger el sensor de la luz directa del sol (radiación UV).

El DHT21 posee mejores prestaciones respecto a los sensores DHT11 y DHT22, como mejor resolución, mayor precisión y un empaque más robusto. (mechatronics, s.f.)

Especificaciones técnicas.

- Voltaje de Operación: 3.5V - 5.5V DC.
- Consumo corriente: 1mA - 1.5mA.
- Rango de Temperatura: -40 hasta 80°C.
- Precisión Temperatura: +- 0.5°C.
- Resolución Temperatura: 0.1°C.
- Rango de Humedad Relativa: 0 a 100% RH.
- Precisión HR: +- 3%.
- Resolución Humedad: 0.1%RH.
- Tiempo de sensado: 2s.
- Interface digital: Single-bus (bidireccional).
- Modelo: AM2301.
- Dimensiones: 60*28*13mm.
- Peso: 17 gr.
- Carcasa de plástico negro.
- Longitud cable: 50cm.

Figura 10 Sensor de temperatura y humedad relativa DHT21 (AM2301) (Mercado libre, s.f.).

2.4 Plataformas para IoT.

2.4.1 Ubidots.

Ubidots es un servicio en la nube que permite almacenar datos de sensores y visualizarlos en tiempo real a través de una página web. También puede configurar alertas Email o SMS dependiendo del valor de los sensores, como por ejemplo "Envíame un SMS cuando mi garaje esté abierto" o "Envíame un Email cada vez que haya un movimiento en mi habitación".

Nacida como una empresa privada de servicios de ingeniería en 2012, Ubidots se especializó en soluciones conectadas de hardware y software para monitorear, controlar y automatizar de forma remota los procesos para clientes de atención médica en empresas nuevas y Fortune 1,000 en el sureste de Estados Unidos y en toda América Latina.

Ubidots ofrece una API REST que le permite leer y escribir datos en los recursos disponibles: fuentes de datos, variables, valores, eventos y perspectivas. La API admite HTTP y HTTPS y se requiere una clave de API.

Sus datos estarán protegidos con dos replicaciones más, almacenamiento encriptado y soporte de datos TLS / SSL opcional. También puede personalizar los grupos de permisos para cada módulo de la plataforma, asegurándose de que la información correcta se muestra al usuario correcto. (Ubidots, s.f.)

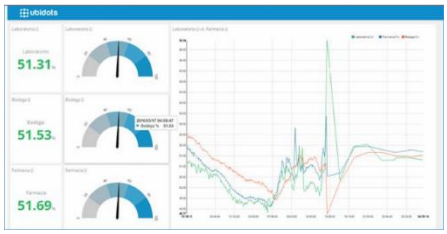


Figura 11 Plataforma Ubidots (Ubidots, s.f.).

2.4.2ThingSpeak.

ThingSpeak es un servicio de plataforma de análisis de IoT que le permite agregar, visualizar y analizar flujos de datos en vivo en la nube. Proporciona visualizaciones instantáneas de los datos publicados por sus dispositivos en ThingSpeak. Con la capacidad de ejecutar el código MATLAB en ThingSpeak, puede realizar el

análisis y el procesamiento en línea de los datos a medida que se incorporan. Se usa a menudo para la creación de prototipos y los sistemas de prueba de concepto IoT que requieren análisis.

ThingSpeak es un servicio de plataforma de análisis de IoT de MathWorks, los creadores de MATLAB y Simulink. ThingSpeak permite agregar, visualizar y analizar flujos de datos en vivo en la nube. ThingSpeak proporciona visualizaciones instantáneas de los datos publicados por sus dispositivos o equipos. Ejecuta el código MATLAB en ThingSpeak, y realiza el análisis y el procesamiento en línea de los datos a medida que ingresan. ThingSpeak acelera el desarrollo de los sistemas de IoT con prueba de concepto, especialmente aquellos que requieren análisis. Puede crear sistemas de IoT sin configurar servidores o desarrollar software web. Para sistemas IoT de tamaño pequeño a mediano, ThingSpeak proporciona una solución alojada que se puede utilizar en la producción.

ThingSpeak almacena toda la información que envía en una ubicación central en la nube, para que pueda acceder fácilmente a sus datos para el análisis en línea o fuera de línea. Sus datos privados están protegidos con una clave API que puede ser controlada por el usuario. Cuando haya iniciado sesión en su cuenta de ThingSpeak, puede utilizar la web para descargar de forma segura los datos almacenados en la nube. También puede leer sus datos mediante programación en formato CSV o JSON utilizando una API REST llamar y la clave API adecuada. Sus dispositivos también pueden leer datos de un canal ThingSpeak al suscribirse a un tema MQTT.

Importa datos de servicios web de terceros, incluidos datos climáticos de NOAA, datos de servicios públicos de proveedores de servicios públicos locales y datos de stock y precios de proveedores financieros. Puede usar esos datos junto con los datos que está recopilando de sus dispositivos y equipos para investigar las correlaciones y desarrollar algoritmos predictivos. (thnigspeak, s.f.)

Figura 12 Plataforma thingspeak (thingspeak, s.f.)

2.4.3 Google Sites.

Google Sites es una aplicación online ofrecida por la empresa estadounidense Google como parte de la suite de productividad de G Suite. Esta aplicación permite crear un sitio web o una Intranet de forma muy sencilla.

Con Google Sites los usuarios pueden reunir en un único lugar y de una forma rápida información variada, como pueden ser vídeos, calendarios, presentaciones, archivos, etc.

El objetivo de Google Sites es que cualquier persona pueda crear un sitio web permitiendo compartir información con un grupo reducido de personas, con toda su organización o con todo el mundo. Resulta muy útil en la creación de intranets, páginas de empleados, etc. (Comunica-web, 2019)

Las 4 ventajas de las que dispone Google Sites para la creación de una Intranet que mejore la comunicación interna de la organización pueden ser:

- Fácil de utilizar. No requiere programación HTML o CSS, aunque se puede editar directamente parte del código, la integración de contenidos no requiere contar con estos conocimientos. Su funcionamiento es muy sencillo y permite que cualquiera añada y modifique cualquier elemento, ya sea de texto, páginas o documentos de descarga.

- Personalizable. En cuanto a la apariencia de la página es totalmente personalizable, pues permite incluir el logo de la empresa y los colores corporativos. El funcionamiento de Google Sites se basa en unos elementos preestablecidos que facilitan en gran medida su usabilidad, pero casi todo es modificable y se puede organizar según mejor le convenga al propietario.

- Mejora la comunicación interna. La correcta utilización de Google Sites puede suponer una mejora de las comunicaciones internas de la empresa, parte fundamental para una organización.

- Mejora la comunicación interna. La correcta utilización de Google Sites puede suponer una mejora de las comunicaciones internas de la empresa, parte fundamental para una organización.

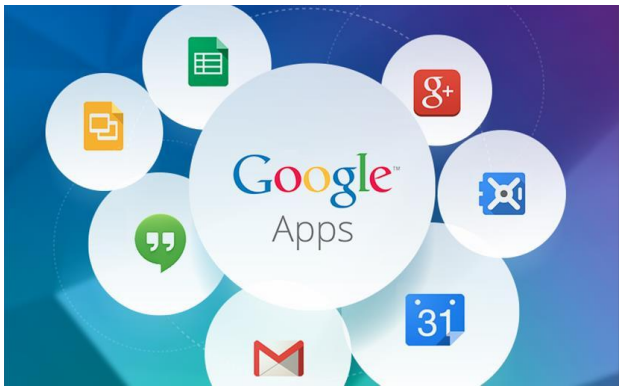


Figura 13 Plataforma Google apps (Akbar, 2015)

Capitulo III. Metodología.

En esta sección se hace una breve descripción de los pasos seguidos para desarrollar el proyecto. Para lograr los objetivos planteados, el proyecto fue dividido en las siguientes fases principales: a). definición del proyecto mediante reuniones con el tutor, b). aprendizaje, c). instrumentos, d). elaboración y e). Desarrollo de pruebas y resultados.

a) Definición del proyecto.

Parte importante es definir el alcance y los objetivos del proyecto, los resultados que se buscaban y la forma en que se van a analizar. Para ello se iban evaluando constantemente los avances, teniendo en cuenta los objetivos planteados; también se tuvieron algunas reuniones con especialistas en el tema para solventar dudas o solicitar información pertinente.

Según la elección de elementos y herramientas descritas en la metodología de la investigación, se diseñó un circuito electrónico de conexión para el sistema embebido encargado de capturar la información dentro de los sensores AM2301. Como se mencionó, el circuito se basa en la plataforma de desarrollo NodeMCU y en el microcontrolador ESP32 lo cual permite un diseño minimalista pero eficiente técnicamente, esto gracias a las diversas características descritas en capítulos anteriores, como su inclusión en la misma plataforma, de un chip para el manejo del acceso a internet vía conexión WI-FI. En la figura 29 se puede apreciar el circuito diseñado para la etapa de captura de datos, procesamiento y envió de datos al internet.

b) Aprendizaje.

Para lograr los resultados finales planteados en el proyecto fue necesario adquirir conocimientos sobre el uso de tecnologías IoT, microcontroladores basados en software de Arduino y el uso de sensores de temperatura y humedad. Conceptos, implementación, ventajas y funcionamiento también fueron parte del conocimiento adquirido.

Se consultaron diferentes fuentes, como por ejemplo los sitios oficiales de ThingSpeak, Ubidots y tutoriales en You Tube para aprender a realizar las configuraciones necesarias en equipos con base en Arduino.

Una de las etapas donde se encontraron dificultades fue al momento de agregar las librerías al programa Arduino aparecieron unos errores que tenían que ver con la librería SHT20 que es la librería de los sensores y la librería U8G2 que es la librería de la pantalla led del controlador.

c) Instrumentos.

Los elementos utilizados para llevar a cabo el proyecto fueron los siguientes: Microcontrolador ESP32.

Figura 14 Microcontrolador ESP32.

Sensor de temperatura y humedad relativa DHT21 (AM2301).

Figura 15 1DTH21 (AM2301).

Una placa de cobre virgen de 10x10.

Figura 16 Placa de cobre.

Borneras de conexión eléctrica.

Figura17 Bornera.electrica

Percloruro de hierro en polvo.

Figura 18 Percloruro de hierro.

d) Elaboración.

Fue necesario hacer pruebas antes de soldar todos los instrumentos a la placa de cobre, para eso se utilizó una bread board electrónica y luego realizar el circuito.

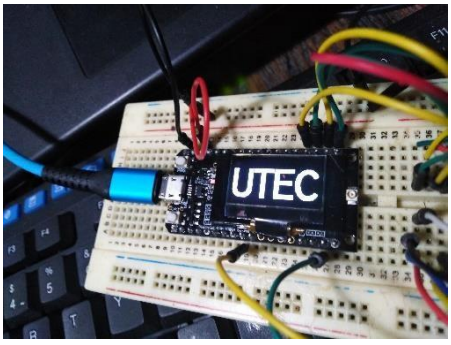


Figura 19 Circuito funcional de prueba sobre bread board (Fuente Propia).

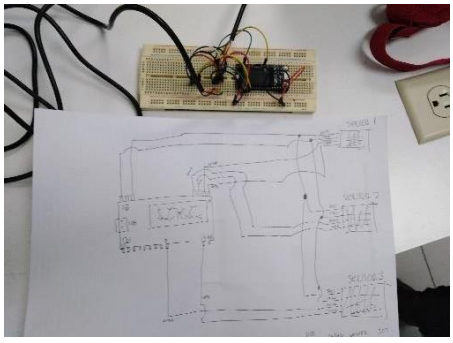


Figura 20 Diseño de circuito electrónico (Fuente propia).

Luego de comprobar que el microcontrolador y los sensores de temperatura y humedad trabajaban correctamente según lo deseado, se prosiguió a crear el diseño del circuito que se grabaría en la placa de cobre.

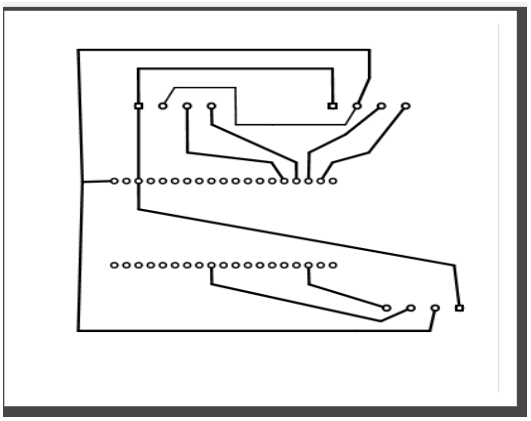


Figura 21 Diseño final de circuito electrónico Fuente propia).

Una vez diseñado el circuito, se prosiguió a grabar el diseño en la placa de cobre para posteriormente remover el cobre que no iba a ser necesitado.

Figura 22 Diseño electrónico grabado sobre pista de cobre virgen (Fuente propia).

Antes de remover el cobre que no se iba a necesitar, se taladraron los orificios necesarios para agregar el microcontrolador, los sensores y las borneras.

Figura 23 Perforación de Pista de cobre (Fuente propia).

Luego de grabar el diseño en la placa de cobre y perforarla, se disolvió percloruro de hierro en agua y se sumergió la placa de cobre hasta que el exceso de cobre fue removido.

Figura 24 Eliminación de exceso de cobre (Fuente propia).

Ya con la placa lista el siguiente paso es montar los elementos en la placa de cobre y soldarlos.

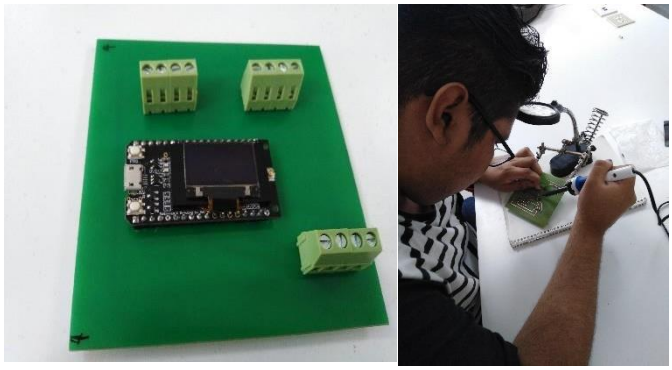


Figura 25 Soldado de elementos sobre la pista de cobre (Fuente propia)

Figura 26 Elementos soldados sobre la pista (Fuente propia)

Por último, se realizaron las respetivas pruebas para confirmar que todos los elementos funcionan correctamente en la placa de cobre.

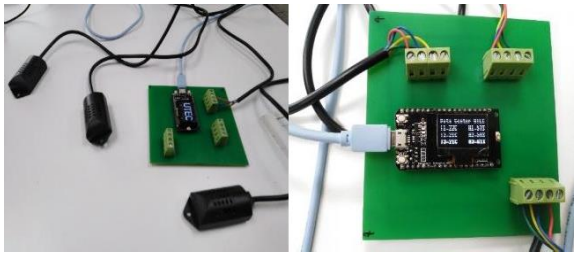


Figura 27 Circuito funcional (Fuente propia)

e) Desarrollo de pruebas y resultados.

Para realizar las pruebas de monitoreo de temperatura y ambiente, se creó una página web a través de Google Sites para ver los resultados enviados por los sensores a través del microcontrolador ESP32.

Figura 28 Creación de página web haciendo uso de Googole Sites

Capitulo IV Análisis de Resultados.

4.1 Circuito electrónico embebido.

El microcontrolador a cargo de todos los procesos internos del circuito electrónico fue programado usando un *firmware* escrito para la función específica requerida por el sistema, dicho programa se escribió usando lenguaje de programación C++ y basado en el algoritmo básico pero eficaz de tres funciones: captura, procesamiento, conexión y envió a ThingSpeak para finalmente mostrar los resultados finales a través de una página web en Google sites, esto cada vez que se presenta una nueva lectura de temperatura y humedad.

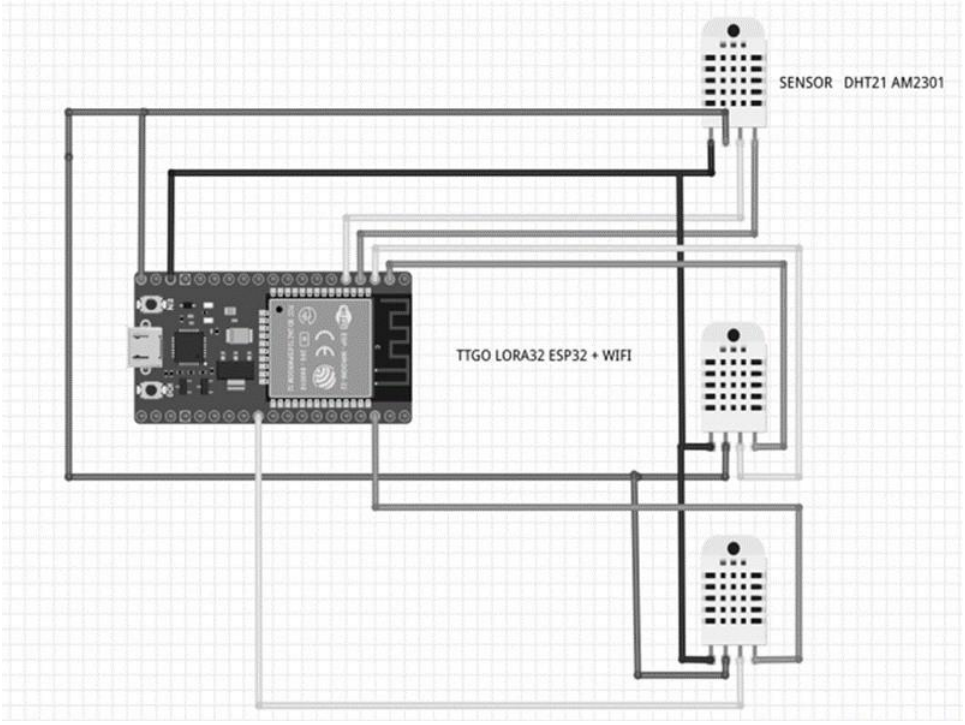


Figura 29 Diseño físico del circuito electrónico (Fuente propia).

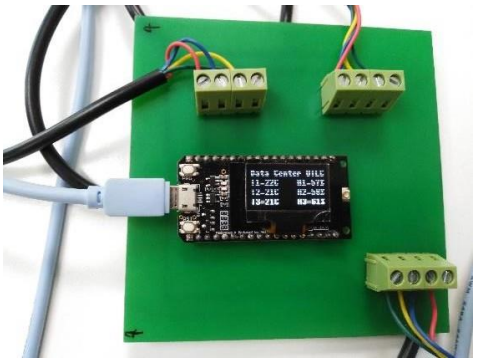


Figura 30 Muestra de resultado de los sensores (Fuente propia).

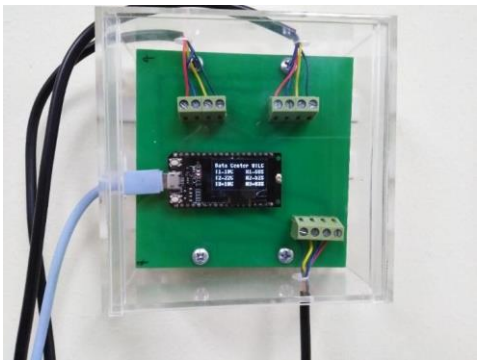
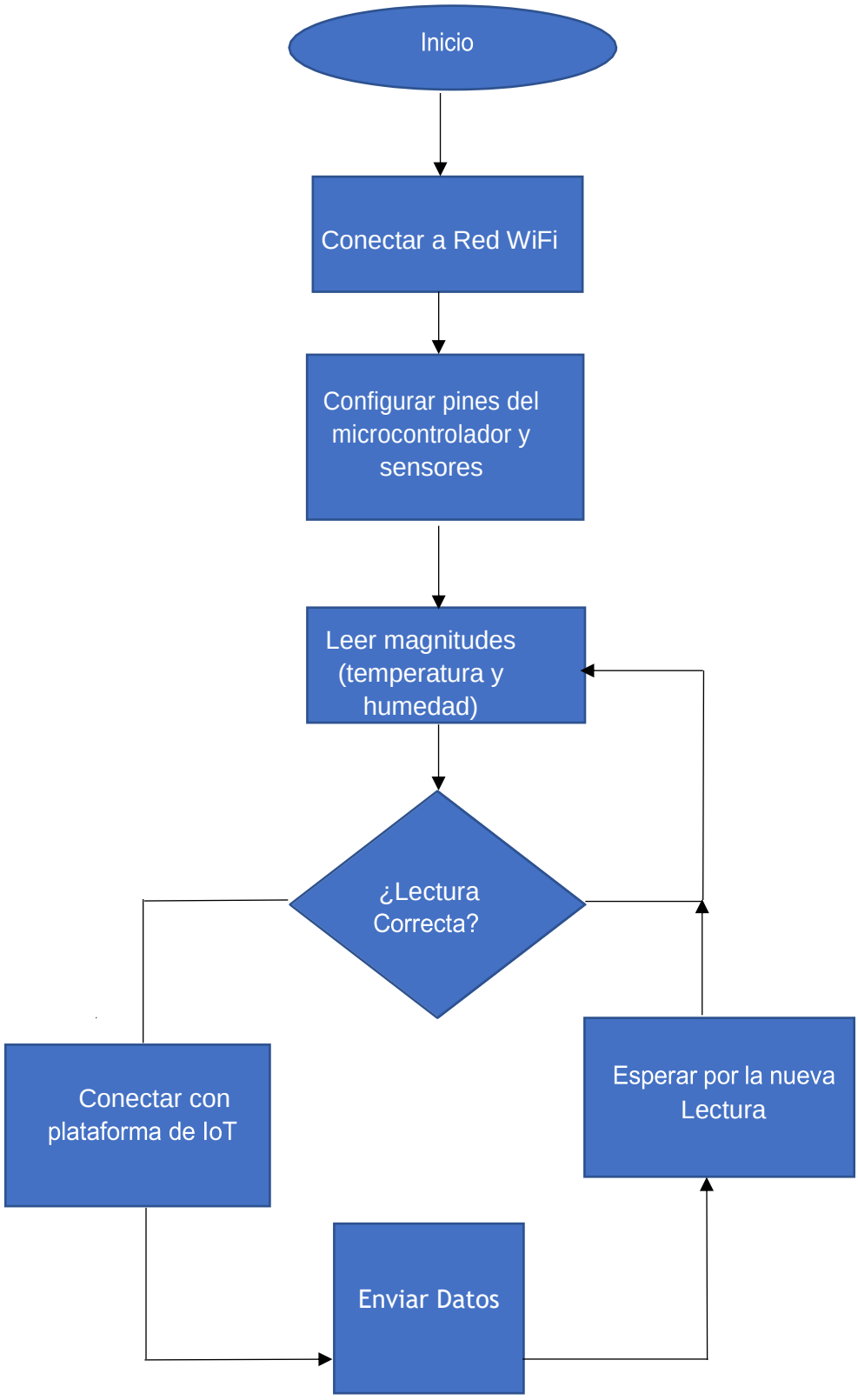


Figura 31 Carcasa de acrílico (Fuente propia).

4.2 Creación de firmware.

Para hacer funcionar el microcontrolador ESP32 junto con los sensores de temperatura y humedad, se creó un código Arduino basado en código C++. La función principal del código es ordenarle al microcontrolador que recopilar la información captada por los sensores de temperatura y humedad y luego enviar los datos vía WI-FI a la nube para su presentación gráfica. El flujograma se muestra a continuación y el código completo puede ser consultado en el anexo 1 de este documento.


Inicio
Conectar a Red WiFi
Configurar pines del microcontrolador y sensores
Leer magnitudes (temperatura y humedad)
¿Lectura Correcta?
Conectar con plataforma de IoT
Esperar por la nueva Lectura
Enviar Datos

4.3 Aplicación IoT y Actualización de Datos.

Para la etapa del *firmware* que se debe ejecutar en el microcontrolador, se diseñó el código fuente, que se puede apreciar en el anexo 1. En la etapa de visualización se utilizó la herramienta de Google Sites para montar un sitio web para acceso desde un navegador con https://sites.google.com/mail.utec.edu.sv/monitoreolabdata/p%C3%A1gina-principal

Las pruebas se realizaron en el laboratorio de Data Center ubicado en la primera planta del edificio Simón Bolívar de la Universidad Tecnológica de El Salvador el día 14 de mayo de 2019. Se comenzó a recolectar datos desde las 12:pm hasta las 5:00pm del mismo día. La temperatura fue medida en grados Celsius (°C), tomando en cuenta el porcentaje de humedad en el ambiente captado por cada uno de los tres sensores. Los resultados fueron los siguientes:

Sensor 1.

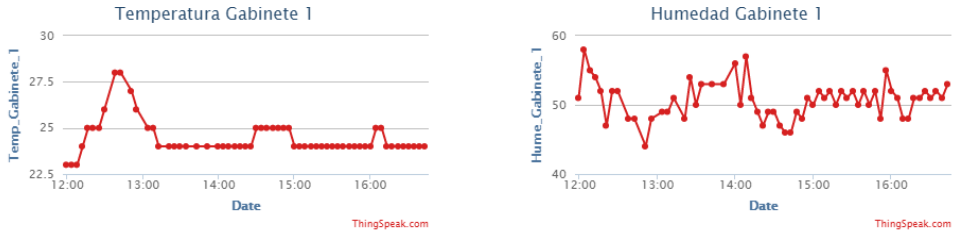


Figura 32 Resultados de sensor 1.

La temperatura más alta registrada fue de 28°C a las 12:38 pm con una humedad en el ambiente del 48%. Para el final de la prueba el sensor 1 registro una temperatura de 24°c a las 4:48pm con una humedad en el ambiente del 52%.

Sensor 2.

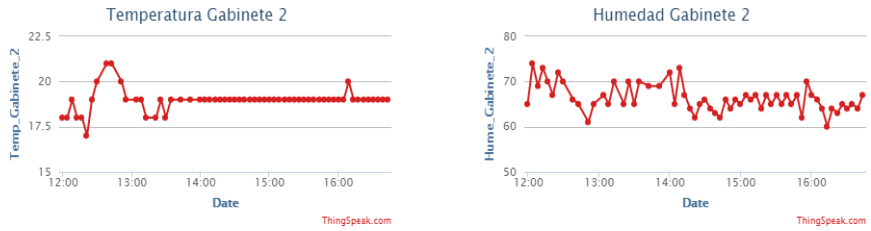


Figura 33 Resultados sensor 2.

La temperatura más alta registrada fue de 21°C a las 12:38pm con una humedad en el ambiente del 64%. Para el final de la prueba el sensor 2 registró una temperatura de 18°C a las 4:48pm con una humedad en el ambiente del 68%.

Sensor 3.

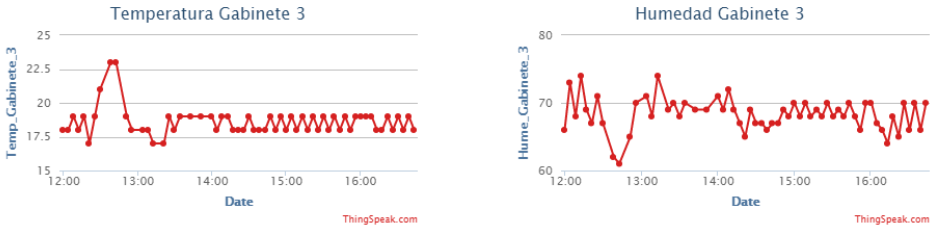


Figura 34 Resultados sensor 3.

La temperatura más alta registrada fue de 23°C a las 12:38pm con una humedad en el ambiente del 61%. Para el final de la prueba el sensor 3 registró una temperatura de 18°C a las 4:48pm con una humedad en el ambiente del 70%.

Los resultados de esta investigación son los siguientes:

- El diseño y construcción del circuito electrónico especifico capaz de realizar la lectura de temperatura y humedad del Laboratorio de Data Center de la Universidad Tecnológica de El Salvador; y enviar estos datos a internet vía conexión WI-FI.
- Creación de un firmware para la placa ESP32 en Arduino con leguaje C++ (ver anexo 1).
- Aplicación o página Web, junto con la configuración de las gráficas generadas por ThingSpeak para la visualización en tiempo real de la información de los sensores de temperatura y humedad.

Las funciones principales del circuito electrónico embebido son: Lectura de la temperatura y humedad del Laboratorio, conexión a internet vía WI-FI y envio de la información a la plataforma de ThingSpeak. La construcción de circuito fue realizada con éxito en una tarjeta de circuito impreso diseñada para el caso (ver figura 32); para su instalación final se utilizó una carcasa de acrílico (ver figura 33).

Capítulo V Conclusiones y Recomendaciones.

5.1 Conclusiones

A modo de conclusión global del trabajo realizado, el sistema IoT diseñado para el registro automatizado de temperatura y humedad cumple con el objetivo general planteado; y se convierte en una herramienta tecnológica de bajo costo que sirve de apoyo en las labores logísticas de una institución donde controlar la temperatura y humedad sea una tarea periódica y muy importante dentro de su quehacer administrativo.

Específicamente, a partir de lo realizado con cada una de las labores en esta investigación aplicada, se pueden formular las conclusiones siguientes:

- Con esta investigación se ha podido aportar nuevo conocimiento científico, de manera que se ha mostrado nuevas formas de utilizar un sistema IoT, como en la solución de problemas de automatización, en el monitoreo de control remoto de procesos con herramientas tecnológicas recientes de bajo costo y eficiente, tales como el microcontrolador ESP32 y la plataforma Google, asequibles en el entorno local.
- El uso de los componentes mencionados permitió el diseño y construcción de un circuito electrónico embebido eficiente y de bajo costo, que cumple con la función de permitir el monitoreo de temperatura y humedad del laboratorio de Data Center de la Universidad Tecnológica de El Salvador, leer su información interna única, decodificarla y enviarla via WI-FI a internet. Este circuito además es de fácil configuración y reproducción para una producción e implementación masiva, por

ejemplo, en un Data Center con más equipos activos.

- Para el software producido, se diseñaron dos soportes: un firmware para el microcontrolador ESP32 y un script para ser almacenado y ejecutado en internet dentro de los servicios de la plataforma Google Sites. El uso de software de desarrollo de licenciamiento libre y fuente abierta Arduino C y el Google App Script permitió el cómodo desarrollo de los códigos completamente funcionales en ambos casos, lo cual demuestra que se pueden realizar desarrollos de software con herramientas disponibles sin costo alguno.
- A manera de validación de lo anterior, se implementó el prototipo de sistema IoT construido en el Laboratorio de Data Center de la Utec, cuyo funcionamiento fue óptimo y se logró las expectativas estimadas: el monitoreo de forma inmediata, desde cualquier punto de acceso a internet, de la información de la temperatura y humedad del laboratorio.

Además, se ha de mencionar que este no es el final de la investigación aplicada en el área de internet de las cosas, hay muchas líneas de aplicaciones que se deben desarrollar y un vasto campo de investigación por delante.

5.2 Recomendaciones.

El equipo de trabajo recomienda continuar con el apoyo a investigaciones de este tipo, ya que es un campo poco explorado dentro del entorno académico, comercial e industrial del país, tomando como base que el IoT es un área que se vislumbra que tiene mucho potencial dentro de nuestro ámbito. El equipo de trabajo propone que se tome como base el conocimiento científico técnico aportado por este trabajo, en el desarrollo de sistemas IoT como el planteado, en instituciones similares a la UTEC.

El equipo de trabajo recomienda utilizar equipos como:

- Arduino Uno o Arduino MEGA.
- Utilizar una Shield ethernet o WI-FI.
- Utilizar una Raspberry pi para garantizar una mejor conexión a internet.
- Utilizar sensores modelo DHT11 para lectura de humedad y temperatura.

El equipo de trabajo recomienda que si se utiliza el microcontrolador ESP32 no exceder de los dos metros de longitud de cable en los sensores.

Referencias

Comunica-web. (2019). *¿Qué es Google Sites? ¿Cuales son sus 4 Ventajas?*. Recuperado de https://www.comunica-web.com/verarticulo--Que-es-Google-Sites-Cuales-son-sus-4-ventajas-_980.php

Crespo, E. (s.f.). *Aprendiendo Arduino*. Recuperado de https://aprendiendoarduino. wordpress.com/2018/11/11/arquitecturas-iot/

Karen Rose, S. E. (2015). *La Internet de las cosas*. Recuperado de https://www.internetsociety. org/wp-content/uploads/2017/09/report-InternetOfThings-20160817-es-1.pdf

Leguízamo, J. L. (2018). *¿Qué es el Internet de las cosas y como funciona?*. Recuperado de https://codigoespagueti.com/noticias/internet/que-es-el-internet-de-las-cosas/

Lowi. (2018). *Internet de las cosas: actualidad y futuro*. Recuperado de https://www.lowi. es/blog/internet-de-las-cosas-futuro/

Nylamp Mechatronics SAC. (s.f.). *Sensor de temperatura y humedad relativa DHT21 (AM2301)*. Recuperado de https://naylampmechatronics.com/sensores-temperatura-y-humedad/354-sensor-de-temperatura-y-humedad-relativa-dht21-am2301.html

Tele Sur. (2016). *¿Qué son las ciudades inteligentes?*. Recuperado de https://www.telesurtv. net/telesuragenda/Que- son-las-ciudades-inteligentes-20161027-0021.html

The MathWorks, Inc. (s.f.). *ThingSpeak*. Recuperado de https://thingspeak.

com/pages/learn_more

Ubidots. (s.f.). *Ubidots*. Recuperado de https://ubidots.com/about/

Fundació Wikimedia, Inc. (s.f.). *ESP32*. Recuperado de

https://en.wikipedia.org/wiki/ESP32

Anexos.

Anexo 1.

```
/*********************************************************************/
/*      ***********************************************************
*       Sistema IoT monitor Temperatura y Humedad
*       Harware:
*       uC Board = TTGO OLED LORA B
*       https://primalcortex.wordpress.com/2017/11/24/the-esp32-oled-lora-ttgo-lora32-board-and-connecting-it-
        tottn/
*       sensors = SHT20
*       Firmware:
*       toma de lectura de magnitudes y reporte a nuvbe IoT
*       en intervalos regulares - 1 por minuto - max nube gratuita
*       *  ******************************************************
*       * Original por:                                          *
*       * Autor:Omar Otoniel FLores                              *
*       * Mail: otoniel.flores@mail.utec.edu.sv                  *
*       * Licencia: GNU General Public License v3 or later *
* ********************************************************** */
/*********************************************************************/


/*********************************************************************
*       Librerias a utilizar                                *
*********************************************************************/
/* acceso WiFi */
        #include
        <WiFi.h>
String api_key = "X8M6QK5KDUF2HQFM";                    // Enter your Write API key from
        ThingSpeak const char *ssid =  "IoT";                   // replace with your wifi ssid
        and wpa2 key const char *password = "D4t4.l4b";
const char* host =
        "LDATA";
        WiFiClient client;

/*** sensor SHT20***/
        #include
        <Wire.h>
#include
        "DFRobot_SHT20.h"
        DFRobot_SHT20
        sht20;
float  humd1  =
        0  ;  float
        temp1  =
        0  ;  float
        humd2 =
        0  ;  float
        temp2  =
        0  ;  float
        humd3 =
```

```
        0 ;  float
        temp3  =
        0 ;
String tempS1 = " "; String tempS2 = ""; String tempS3 = "";
        String humS1 = ""; String humS2 = ""; String humS3 =
        "";

/***   OLED                    ***/
// Universal 8bit Graphics Library
        (https://github.com/olikraus/u8g2/) #include <Arduino.h>
#include <U8x8lib.h>
#ifdef
        U8X8_HAVE_HW_S
        PI #include <SPI.h>
        #endif
// constructor
U8X8_SSD1306_128X64_NONAME_SW_I2C Oled(/* clock=*/ 15, /* data=*/ 4, /* reset=*/ 16);

/*****************************************************************
*       Setup                   *
*****************************************************************/
        void setup() {
/*   Oled                               */ Oled.begin();
Oled.setFont(u8x8_font_inb46_4x8_r);
        Oled.drawString(0, 1, "UTEC");
        delay(3000);
/*      monitor      Serial      */
        Serial.begin(115200);
        delay(10);
/* conectar a WiFi */
        Serial.println("Connecting to ");
        Serial.println(ssid);
        WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED)
{
delay(500); Serial.print(".");
}
Serial.println(""); Serial.println("WiFi
        connected"); Oled.clear();
Oled.setFont(u8x8_font_7x14B_1x2_f); Oled.drawString(0, 0, "conectado a WiFi");
        Oled.setFont(u8x8_font_7x14B_1x2_f); Oled.drawString(0, 4, ssid);
delay(5000); }

/*****************************************************************
*       Programa Principal LOOP                                 *
*****************************************************************/
        void loop()
{
leerSensores(); //leer los sensores aprox 30 seg. subirDatos();
        //se suben los datos (6) a la nube de IoT esperar();
                                //esperar hasta proxima
        lectura }

/****************** Funciones **********************/
```

```
/*********************************************************/
/*** Esperar pediodo entre lectura y subida a Nube IoT ***/
/*********************************************************
        **/ void esperar() {
delay(10000);
        delay(10000);
        delay(10000);
        delay(10000);
        delay(10000);
        delay(10000);

delay(10000);
        delay(10000);
        delay(10000);
        delay(10000);
        delay(10000);
        delay(10000);
delay(10000);
        delay(10000);
        delay(10000);
        delay(10000);
        delay(10000);
        delay(10000);
delay(10000);
        delay(10000);
        delay(10000);
        delay(10000);
        delay(10000);
        delay(10000);

}
/*********************************************************/
/*** Leer los sensores de Temp y Humd***/
/*********************************************************/
        void leerSensores()
{
Oled.clear();
        Oled.setFont(u8x8_font_inb46_4x8_r);
        Oled.drawString(0, 1, "leer");
//* inicial toma de lectura de sensores uno a la vez *//
//activar y leer sensor 1
        Wire.begin(21,13);
        sht20.initSHT20();
        delay(100);
        sht20.checkSHT20();
humd1 = sht20.readHumidity(); temp1 =
        sht20.readTemperature();
        delay(4000);
//activar y leer sensor 2
        Wire.begin(12,14);
        sht20.initSHT20();
        delay(100);
        sht20.checkSHT20();
```

```
humd2 = sht20.readHumidity(); temp2 =
        sht20.readTemperature(); delay(4000);
//activar y leer sensor 3
        Wire.begin(17,22);
        sht20.initSHT20();
        delay(100);
        sht20.checkSHT20();
humd3 = sht20.readHumidity();
temp3 = sht20.readTemperature(); delay(4000);
//*** Imprimir en OLED ***// Oled.clear();

tempS1 = String(int(temp1));tempS2 = String(int(temp2));tempS3 =
        String(int(temp3)); humS1 = String(int(humd1));humS2 =
        String(int(humd2));humS3 = String(int(humd3));
        Oled.setFont(u8x8_font_7x14B_1x2_f); Oled.drawString(0, 0, "Data Center
        UTEC");
Oled.drawString(0, 2, "T1= "); Oled.setCursor(3, 2); Oled.print(tempS1); Oled.drawString(5, 2, "C");
        Oled.drawString(10, 2, "H1= "); Oled.setCursor(13, 2); Oled.print(humS1); Oled.drawString(15, 2, "%");
        Oled.drawString(0, 4, "T2= "); Oled.setCursor(3, 4); Oled.print(tempS2); Oled.drawString(5, 4, "C");
        Oled.drawString(10, 4, "H2= "); Oled.setCursor(13, 4); Oled.print(humS2); Oled.drawString(15, 4, "%");

Oled.drawString(0, 6, "T3= "); Oled.setCursor(3, 6); Oled.print(tempS3); Oled.drawString(5, 6, "C");
        Oled.drawString(10, 6, "H3= "); Oled.setCursor(13, 6); Oled.print(humS3); Oled.drawString(15, 6, "%"); }

/*********************************************************/
/*** Enviar a la nube de IoT ***/
/*********************************************************/
        void subirDatos()
{
Oled.clear();
Oled.setFont(u8x8_font_inb46_4x8_r);
        Oled.drawString(0, 1, "aIoT");
Serial.println("Prepare to send data");
        WiFiClient client;
const int httpPort = 80;
if (!client.connect(host, httpPort)) {
        Serial.println("connection failed"); return;
            }
            else
            {
                String data_to_send = api_key;
                        data_to_send += "&field1=";
                        data_to_send +=
                        String(tempS1); data_to_send
                        += "&field2="; data_to_send
                        += String(humS1);
                        data_to_send += "&field3=";
                        data_to_send +=
                        String(tempS2); data_to_send
                        += "&field4="; data_to_send
                        += String(humS2);
                        data_to_send += "&field5=";
                        data_to_send +=
                        String(tempS3); data_to_send
                        += "&field6="; data_to_send
```

```
        +                tring(humS3);
        =
        S        data_to_send += "\r\n\r\n";

client.print("POST      /update      HTTP/1.1\n");
        client.print("Host:   api.thingspeak.com\n");
        client.print("Connection: close\n");
client.print("X-THINGSPEAKAPIKEY: " + api_key + "\n");
        client.print("Content-Type: application/x-www-form-urlencoded\n");
        client.print("Content-Length: ");
client.print(data_to_send.length()); client.print("\n\n");
                                              client.print(dat
        a_to_send);

delay(3000);
        Oled.clear();
}

client.stop();
Oled.setFont(u8x8_font_7x14B_1x2_f); Oled.drawString(0, 0, "Data Center UTEC");
        Oled.drawString(0, 2, "T1= "); Oled.setCursor(3, 2); Oled.print(tempS1); Oled.drawString(5, 2,
        "C"); Oled.drawString(10, 2, "H1= "); Oled.setCursor(13, 2); Oled.print(humS1);
        Oled.drawString(15, 2, "%");
Oled.drawString(0, 4, "T2= "); Oled.setCursor(3, 4); Oled.print(tempS2); Oled.drawString(5, 4, "C");
        Oled.drawString(10, 4, "H2= "); Oled.setCursor(13, 4); Oled.print(humS2); Oled.drawString(15, 4, "%");

Oled.drawString(0, 6, "T3= "); Oled.setCursor(3, 6); Oled.print(tempS3); Oled.drawString(5, 6, "C");
        Oled.drawString(10, 6, "H3= "); Oled.setCursor(13, 6); Oled.print(humS3); Oled.drawString(15, 6, "%");

}

/*********************************************************/
/* FIN DEL CODIGO*/
/*******************************************************
```