Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

DISEÑO DE UN SISTEMA DIGITAL DE SENSORES PARA MEDIR DEFORMACIONES DE SUPERFICIES

José Alejandro López Quel
Asesorado por el Ing. Iván René Morales Argueta

Guatemala, enero de 2020

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DISEÑO DE UN SISTEMA DIGITAL DE SENSORES PARA MEDIR DEFORMACIONES DE SUPERFICIES

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

JOSÉ ALEJANDRO LÓPEZ QUEL
ASESORADO POR EL ING. IVÁN RENÉ MORALES ARGUETA

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN ELECTRONICA

GUATEMALA, ENERO DE 2020

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANA	Inga. Aurelia Anabela Cordova Estrada
VOCAL I	Ing. José Francisco Gómez Rivera
VOCAL II	Ing. Mario Renato Escobedo Martinez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Luis Diego Aguilar Ralón
VOCAL V	Br. Christian Daniel Estrada Santizo
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Pedro Antonio Aguilar Polanco
EXAMINADORA	Inga. Ingrid Salomé Rodríguez de Loukota
EXAMINADOR	Ing. Guillermo Antonio Puente Romero
EXAMINADOR	Ing. Carlos Eduardo Guzmán Salazar
SECRETARIA	Inga. Lesbia Magalí Herrera López

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DISEÑO DE UN SISTEMA DIGITAL DE SENSORES PARA MEDIR DEFORMACIONES DE SUPERFICIES

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha junio de 2017.

José Alejandro López Quel

Guatemala 18 de junio de 2019

Ingeniero
Julio Cesar Solares Peñate
Coordinador del Área de Electrónica
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC.

Apreciable Ingeniero Solares.

Me permito dar aprobación al trabajo de graduación titulado **"Diseño de un sistema digital de sensores para medir deformaciones de superficies"**, del señor José Alejandro López Quel, por considerar que cumple con los requisitos establecidos.

Por tanto, el autor de este trabajo de graduación y, yo, como su asesor, nos hacemos responsables por el contenido y conclusiones del mismo.

Sin otro particular, me es grato saludarle.

Atentamente.

Iván René Morales Argueta
Ingeniero Electrónico
Colegiado 12489

F._____

Ing. Iván Rene Morales Argueta

Colegiado 12,489

Asesor.

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÍA

Guatemala, 10 de julio de 2019

Señor Director
Armando Alonso Rivera Carrillo
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC

Estimado Señor Director:

Por este medio me permito dar aprobación al Trabajo de Graduación titulado **DISEÑO DE UN SISTEMA DIGITAL DE SENSORES PARA MEDIR DEFORMACIONES DE SUPERFICIES**, desarrollado por el estudiante **José Alejandro López Quel**, ya que considero que cumple con los requisitos establecidos.

Sin otro particular, aprovecho la oportunidad para saludarlo.

Atentamente,

ID Y ENSEÑAD A TODOS

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

Ing. Julio César Solares Peñate
Coordinador de Electrónica

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÍA

REF. EIME 42. 2019.

El Director de la Escuela de Ingeniería Mecánica Eléctrica, después de conocer el dictamen del Asesor, con el Visto bueno del Coordinador de Área, al trabajo de Graduación del estudiante: JOSÉ ALEJANDRO LÓPEZ QUEL Titulado; DISEÑO DE UN SISTEMA DIGITAL DE SENSORES PARA MEDIR DEFORMACIONES EN SUPERFICIES, procede a la autorización del mismo.

Ing. Armando Alonso Rivera Carrillo

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

GUATEMALA, 9 DE OCTUBRE 2019.

Universidad de San Carlos
De Guatemala

Facultad de Ingeniería
Decanato

Ref. DTG.016-2020

La Decana de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica, al trabajo de graduación titulado: **DISEÑO DE UN SISTEMA DIGITAL DE SENSORES PARA MEDIR DEFORMACIONES DE SUPERFICIES,** presentado por el estudiante universitario: **José Alejandro López Quel,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, se autoriza la impresión del mismo.

IMPRÍMASE.

Inga. Aurelia Anabela Cordova Estrada
Decana

Guatemala, enero de 2020

AACE/asga
cc

ACTO QUE DEDICO A:

Mis padres	Emma Jeanette Quel y José López. Por su apoyo incondicional y amor a través de los años, por ser la guía y fuente de motivación de mi vida.
Mis hermanos	Jeannette y Alvaro López. Por compartir y ser parte de mi vida.
Mis abuelos	María Ramírez y Leopoldo Quel. Por toda su sabiduría, experiencias y todo su amor que me han brindado.
Mi familia	María Isabel Quel, Ronny Quel, Lorena Santos, Pablo Quel y David Quel. Por siempre creer en mí.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por ser la casa de estudios que me preparó y formó como profesional.
Facultad de Ingeniería	Por brindarme la oportunidad de aprender y desarrollar mis conocimientos y habilidades.
Ing. Iván Morales	Por su asesoría en la realización de este trabajo.
IEEE	Por las oportunidades que me brindaron y por permitirme pertenecer a la institución.
Amigos y compañeros de proyectos	Freddy Lorenti e Irving Cosillo. Por las experiencias, la paciencia y amistad a lo largo de la carrera.
Mis amigos	Alan Ramírez, Andrés Tejeda, Alejandra Contreras y Mauricio Estrada. Por su apoyo incondicional y los momentos compartidos.
Amigos y compañeros de trabajo	Gurgui Juárez, Alvaro García y Andrés Rivera. Por el apoyo brindado.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
A	Amperios
LiPo	Batería de polímero de litio
QoS	Calidad de servicio
cm	Centímetro
I2C	Circuito inter-integrado
PC	Computador personal
LADAR	Detección y rango de imágenes láser
LIDAR	Detección y rango de luz
OLED	Diodo orgánico de emisión de luz
IDE	Entorno de desarrollo integrado
E/S	Entrada/Salida
GPIO	Entrada/Salida de propósito general
WiFi	Fidelidad inalámbrica
GHz	Gigahertz
SSID	Identificador de conjunto de servicios
API	Interfaz de programación de aplicaciones
SPI	Interfaz periférica serial
IoT	Internet de las cosas
M2M	Máquina a máquina
MB	Megabytes
MHz	Megahertz
m	Metro
µm	Micrómetro

mA	Miliamperios
mAh	Miliamperios hora
mm	Milímetro
ms	Milisegundos
mW	Miliwatts
MVC	Modelo vista-controlador
PWM	Modulación por ancho de pulsos
nm	Nanómetro
SBC	Ordenador de placa simple
IP	Protocolo de internet
ToF	Tiempo de vuelo
UART	Transmisor-Receptor asíncrono universal
ALU	Unidad aritmeticológica
V	Voltios
VDC	Voltios de corriente directa
Wh	Watts hora

GLOSARIO

ADC Conversión análoga a digital, por sus siglas en inglés.

ARM Tipo de arquitectura de procesador, utilizada en los sistemas embebidos.

Bit Dígito del sistema de numeración binario. Utilizado para medir la capacidad de almacenamiento de memoria digital.

CPU Unidad central de procesamiento, por sus siglas en inglés.

DHCP Protocolo de configuración dinámica de host, por sus siglas en inglés. Es un protocolo de red de tipo cliente/servidor mediante el cual se asigna dinámicamente una dirección IP y otros parámetros de configuración a cada dispositivo en una red.

FQDN Nombre de dominio completo, por sus siglas en inglés. Es un nombre que incluye el nombre de la computadora y el nombre de dominio asociado a ese equipo, por el cual se puede descubrir y acceder a través de una red.

Google Chrome	Explorador web de código cerrado desarrollado por Google.
Hardware	Conjunto de elementos físicos o materiales que constituyen un dispositivo o sistema electrónico.
Internet de las cosas	Sistema de dispositivos interrelacionados, que poseen identificadores únicos y la capacidad de transferir datos y comunicarse a través de una red, sin requerir interacciones humano a humano o humano a computadora.
LAN	Red de área local, por sus siglas en inglés.
Microcontrolador	Circuito integrado programable, capaz de ejecutar las instrucciones grabadas en su memoria.
MQTT	Transporte de telemetría de cola de mensajes, por sus siglas en inglés. Es un protocolo de comunicación de maquina a máquina, orientado a la comunicación de sensores.
PWM	Modulación por ancho de pulsos, por sus siglas en inglés.
RAM	Memoria de acceso aleatorio, por sus siglas en inglés.
USB	Bus serial universal, por sus siglas en inglés. Sigue un estándar que define los cables, conectores y

	protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre computadoras, periféricos y dispositivos electrónicos.
Python	Lenguaje de programación interpretado, multiparadigma.
ROM	Memoria de solo lectura, por sus siglas en inglés.
Router	Dispositivo que permite interconectar ordenadores en el marco de una red.
SBC	Ordenador de placa única, por sus siglas en inglés.
Script	Código de programación que permite la automatización de tareas, utilizando un conjunto de instrucciones.
Sensor	Dispositivo que posee una propiedad sensible a una magnitud del medio, y al variar esta magnitud también varia con cierta intensidad esta propiedad.
Web framework	Tipo de software diseñado para el soporte de desarrollo de aplicaciones web.

RESUMEN

Se presenta el diseño de un sistema de sensores para medir deformaciones, empleando una interfaz web para la visualización de los datos obtenidos.

En el primer capítulo se describe los conceptos básicos del funcionamiento del instrumento que se utiliza como base en el desarrollo del sistema de sensores, el deformímetro. Estos conocimientos son necesarios para entender e interpretar el contexto de este trabajo, el cual consiste en actualizar el instrumento antes descrito añadiéndole un módulo de toma de datos automatizado, visualización y almacenamiento de estos.

En el segundo capítulo se expone la teoría de los componentes electrónicos utilizados para el desarrollo del prototipo. Se explica las formas de medir distancias empleando distintas técnicas ópticas. Después de esto se define los protocolos de comunicación máquina a máquina y sus principios de funcionamiento. Por último, se describen generalidades de los microcontroladores y ordenadores de placa simple, así como también, otros dispositivos utilizados en el dispositivo.

El tercer capítulo comprende los conceptos complementarios de los distintos componentes involucrados en la realización del dispositivo. Se exponen elementos específicos utilizados en el desarrollo del dispositivo.

En el cuarto capítulo se detalla el diseño final, los diagramas de funcionamiento, algoritmos de programación y guías para utilizar el sistema. De igual manera, se incluye un presupuesto para la realización de los sensores.

Por último, se explica la programación, el diseño y el desarrollo del software del sistema. Se establecen consideraciones básicas para ser implementado y se explica el funcionamiento de los distintos módulos mediante diagramas de bloques.

OBJETIVOS

General

Diseñar un sistema digital de sensores confiable y preciso, para medir deformaciones superficiales.

Específicos

1. Exponer y demostrar que empleando sistemas de IoT se puede desarrollar y renovar instrumentos de laboratorio.

2. Automatizar el proceso de medición y recolección de datos, que se obtienen de los deformómetros.

3. Desarrollar una plataforma para visualización y almacenamiento de los datos obtenidos del instrumento.

4. Proporcionar un instrumento replicable de bajo costo.

INTRODUCCIÓN

Los instrumentos de medición han sido un aporte importante para la innovación, desarrollo, automatización y control de calidad, tanto de empresas y laboratorios, ya que los datos que aportan estos dispositivos ayudan al análisis y toma de decisiones. Existen distintos instrumentos de medición los cuales se clasifican por la magnitud que pueden medir. Los instrumentos que poseen la capacidad de medir longitudes constituyen una de las clases más importantes y la forman instrumentos como el flexómetro, el calibre, el micrómetro, el interferómetro, etc. Uno de los instrumentos de medición de longitud más utilizados es el deformómetro, el cual se utiliza para medir superficies, verificar o rectificar su forma o perímetro, tal como sucedería en los procesos con fresadoras, tornos, centros de mecanizado, en el ajuste de los pistones de un motor o ensayos en un laboratorio de estructuras. También es utilizado para el control del error de forma de piezas y para la medida comparativa entre la dimensión de una pieza sujeta a evaluaciones ya sea en laboratorios, talleres o empresas de fabricación.

El deformómetro que se sigue utilizando es un instrumento analógico, el cual tiene que estar en contacto con la superficie de las máquinas o piezas para determinar las deformaciones, siendo esta una desventaja ya que la mayoría de los ambientes en los que se utiliza este debe interactuar con máquinas en funcionamiento o piezas en movimiento. Otra desventaja que presenta es la forma de obtener los datos, ya que esta se realiza mediante observación directa del instrumento, por lo que una persona debe estar anotando manualmente los datos obtenidos.

Por esto se presenta una solución para actualizar este instrumento de medición. Mediante el uso de sensores de tiempo de vuelo, tecnología IoT, protocolos de comunicación y desarrollo web, se plantea el diseño de un sistema para medir deformaciones superficiales, con instrumentos que puedan ser replicados y utilizados en conjunto, con el cual se pretende optimizar el proceso de recolección de datos y brindar mayor precisión y confiabilidad.

1. DIAGNÓSTICO DEL INSTRUMENTO POR MEJORAR

1.1. Deformómetro

El deformómetro también conocido como deformímetro, comparador de caratula o reloj comparador, es un instrumento compuesto por una carátula similar a un reloj, en donde muestra las diferencias superficiales (deformaciones). Puede presentar una pieza o deformaciones entre una o más piezas, causadas por ensayos realizados con ellas y por el exceso o carencia de material, defectos de fabricación, desalineamientos, descentramientos, excentricidad, desviaciones y todo tipo de errores de planitud, circularidad, esfericidad o desplazamientos incorrectos.

Figura 1. **Deformómetro**

Fuente: Deformómetro 0.050” de rango y 0.0001" de precisión. http://www.aconstructoras.com/product_info.php?products_id=4448. Consulta: 7 de abril de 2018.

1.2. Partes de un deformómetro

Consta de una barra central (husillo cilíndrico) que en el extremo inferior tiene un palpador y en el superior una cremallera dentada, la cual, a su vez, está conectada a un tren de engranajes que transmiten el movimiento a dos agujas en el reloj (aguja principal y aguja cuenta vueltas). Estas rotan sobre una escala reglada en la caratula para representar las variaciones superficiales de las piezas. Cada vuelta completa de la aguja principal, a lo largo de la escala del dial, es marcada por una unidad en la aguja pequeña.

Figura 2. **Partes de un deformómetro**

Fuente: LÓPEZ H. Revista Metal Actual. Instrumentos Infaltables en un Taller: El Deformímetro. p. 54

1.3. Determinación de la deformación

A continuación, se detallan los pasos a seguir para utilizar el deformómetro.

- Primero se debe de colocar sobre un soporte de base magnética, la cual se sujeta a un componente metálico cerca de la pieza a medir o comparar.
- Luego es necesario aproximar la punta del palpador a la superficie de la pieza hasta realizar un contacto directo, mínimo de 200 micrómetros.
- Después se debe ajustar manualmente el aro giratorio del reloj para que la aguja principal coincida exactamente con el punto cero de la escala.
- Luego procede a medir la deformación de la pieza en movimiento o comparar una pieza con respecto a otra.

1.4. Lectura del deformómetro

Las deformaciones se basan en los movimientos del husillo cilíndrico. El husillo cilíndrico se desplaza en movimientos longitudinales, de abajo hacia arriba, mientras las agujas se mueven circularmente y muestran las variaciones superficiales de la pieza en la escala reglada de la carátula.

Por ejemplo, en un deformímetro con rango de medición entre 0,01 mm a 10 mm, la aguja principal gira en total diez vueltas, que son registradas, una a una, por la aguja pequeña, en la carátula secundaria. Así el usuario logra determinar cuántas vueltas ha dado la aguja principal en la escala y verificar la deformidad de la pieza. En este caso, cada división (línea) de la escala equivalente a 10 micrómetros; entonces, una vuelta completa es proporcional a 1 000 micrómetros (1,0 mm).

Si la aguja principal del dial se mueve en el sentido de las manecillas de un reloj el valor es positivo, y representa un pico (bache) en la superficie de la pieza; por el contrario, si la aguja se mueve en contra del reloj, el valor indicado por el instrumento es negativo y representa un valle o hendidura.

1.5. Tipos de deformómetros y derivados

Los deformómetros se pueden clasificar por la exactitud que poseen, según el tipo de medidas que se requieran verificar o comparar. Generalmente se suelen medir rangos en milímetros o pulgadas, de 0,25 mm a 300 mm (0,015 a 12,0 pulgadas) con resoluciones de 0,001 mm a 0,01 mm o 0,00005 a 0,001 pulgadas.

Según la forma de lectura, los deformómetros, se clasifican en análogos o digitales. Los primeros registran variaciones en milímetros o pulgadas, mientras que los digitales muestran ambas lecturas. Los comparadores digitales son más precisos, pues mientras la división de escala de un deformómetro análogo es de 10 micrómetros, en el digital es de 1,0 micrómetros; es decir, diez veces más. Los instrumentos digitales poseen teclas para iniciar la medición en ceros y seleccionar el cambio de unidad, de milímetros a pulgadas.

Figura 3. **Deformómetro digital**

Fuente: Deformímetro, digital 0-12.7 mm graduación 0.01 mm. http://www.aconstructoras.com/product_info.php?products_id=3961. Consulta: 7 de abril de 2018.

Además, a este instrumento se le pueden agregar accesorios y monturas que, en conjunto, forman un nuevo instrumento de medición con base en el deformímetro básico. Esto confiere más comodidad y agilidad en mediciones especiales. Entre ellos se encuentran el medidor de espesores, como su nombre lo indica, mide el espesor de piezas pequeñas, en general menores a 10 mm, y el medidor de interiores, que es utilizado para medir diámetros internos o conjuntos de diámetros.

Figura 4. **Medidor de espesores**

Fuente: Medidor de espesores. https://techindustrysac.com/productos/178-medidor-de-espesores-mitutoyo-7360. Consultado: 7 de abril de 2018.

Figura 5. **Medidor de interiores**

Fuente: Medidor de interiores. https://www.tpfcomercial.com/fichaformat/metrologia/Medidor-de-espesores-compas-rapido-de-interiores-y-exteriores/Palpador-rapido-int-InterInter-70-120-mmKroplin/4053569108904. Consultado: 7 de abril de 2018.

1.6. Ventajas y Desventajas de las condiciones actuales

El deformímetro es un instrumento que, en su evolución, se ha adaptado a diferentes propósitos, como se mencionaba en el apartado anterior. Entre los avances tecnológicos que han tenido estos instrumentos, están los deformímetros digitales, cuyo principio de medición se realiza a través de un rayo infrarrojo. En estas herramientas, el husillo está provisto de un disparo infrarrojo que cae sobre un punto preciso de la superficie, a medida que el rayo se desplaza a lo largo de la superficie de la pieza, registra las deformaciones de esta. A continuación, se analizará las ventajas y desventajas del instrumento.

1.6.1. Ventajas

- Poseen distintas resoluciones para diferentes aplicaciones.
- Los instrumentos digitales poseen escalas de hasta 1,0 micrómetro, para medir deformaciones con mayor precisión.
- Existencia de bases, monturas y accesorios especiales para necesidades específicas.

1.6.2. Desventajas

- Desajuste de la aguja o de los engranajes internos por sobre carga o golpes, causando que no halla repetitividad con los datos obtenidos.
- Suciedad en el vástago causada por lubricantes o mala limpieza. Esto altera los datos de deformaciones obtenidas debido a que el vástago no se desplaza de forma continua.
- Observación errónea de los datos. Tanto el modelo analógico como el digital puede tener este problema, ya que al transcribir los datos ya sea a una hoja de cálculo o una base de datos para su análisis estos pueden ser

alterados por usuario debido al paralaje en el modelo análogo o la poca visibilidad en el modelo digital debido a que las pantallas que poseen son demasiado pequeñas.

2. TEORIA DE LOS COMPONENTES DEL PROTOTIPO

A continuación, se detalla la teoría de los componentes utilizados para el prototipo.

2.1. Métodos ópticos para la medición de distancias

Los medidores ópticos son también llamados LADAR o LIDAR (Detección y rastreo por láser y detección y rastreo por luz respectivamente). También se les denomina rangefinder por láser. La expresión radar láser incluye ambos, buscadores de rango por láser (medidores de distancia por láser) y los dispositivos medidores de la absorción y la dispersión de la luz en la atmosfera.

2.1.1. Clasificación de los métodos ópticos para la medición de distancias

A continuación, se describe como se clasifican los métodos ópticos para medir distancias.

2.1.1.1. Por medio de la fuente de luz que utiliza

Un criterio de clasificación es la fuente de luz que utilizan y se dividen en forma pasiva y activa.

2.1.1.1.1. Forma pasiva

El método pasivo no necesita de su propia fuente de luz, en este se utiliza la luz del ambiente para la recolección de la información de la distancia desde el obstáculo.

2.1.1.1.2. Forma activa

El método activo posee una fuente de luz propia para iluminar el objeto y determinar su distancia. Los métodos activos más importante son los métodos interferométricos, método geométrico (triangulación) y el de tiempo de vuelo.

2.1.1.2. Por medio de la fuente de los datos

Otro tipo de clasificación divide los métodos por medio de como obtienen los datos.

2.1.1.2.1. Método directo

El método directo entrega una distancia inequívoca para algunos puntos en el obstáculo.

2.1.1.2.2. Método basado en imágenes

En el método basado en imágenes la distancia es calculada con algunos algoritmos basados en tonos o posiciones relativas de las diferentes partes del obstáculo.

2.1.1.3. Por medio de las vistas

El tercer tipo de clasificación divide los métodos en métodos de medición de vistas múltiples y monocular.

2.1.1.3.1. Métodos monoculares

Los métodos monoculares están frecuentemente basados en el cálculo de las formas a partir del sombreado, textura, movimiento o enfocado desde una imagen creada con una CCD (una matriz de fotodiodo unidimensional) o sensores de imagen CMOS (matriz de fotodiodos). En este método los problemas típicos son las pequeñas diferencias en la iluminación (especialmente al aire libre) o la reflexión del obstáculo.

2.1.1.3.2. Métodos de vistas múltiples

Los métodos de vistas múltiples también se clasifican como métodos de triangulación o como estéreo visión. Los métodos de triangulación son los más comunes métodos medidores de distancia, porque son más simples y fiables. La triangulación pasiva se basa en la medición del ángulo formado entre los puntos de dos receptores (cámaras) y un obstáculo. Con estos métodos hay mayor posibilidad de "pérdida de partes", lo cual significa que el obstáculo está parcialmente sombreado por algún otro objeto en la escena.

Los métodos de vistas múltiples activas están basados en estéreo visión (triangulación) con una contribución de iluminación activa del obstáculo, los cuales hacen más fácil calcular la distancia. La contribución de iluminación es realizada con un haz de láser que, usualmente, tienen la forma de una banda, red o algunos otros patrones. La ventaja de utilizar un láser es la pequeña

divergencia y los pequeños tamaños de manchas, pero la alta coherencia puede también traer patrones de interferencia no deseados. Las manchas luminosas también pueden realizarse con una fibra rallada. En este contexto la fibra rallada significa que las dos capas (en un ángulo de 90 grados por cada otro) de fibras paralelizadas son usados como lo óptico para la creación de una red de manchas. La precisión para la medición de una mancha puede alcanzar, incluso, la precisión de 50 µm en una distancia de 3 metros.

2.1.2. Otros métodos

Además de los métodos descritos anteriormente, también existen métodos que no se pueden clasificar, los cuales se describen a continuación.

2.1.2.1. Método interferométrico

El método interferométrico incluye técnicas de Moiré, interferometría holográfica y difracción de Fresnel. En la técnica de Moiré el obstáculo es iluminado con una luz de láser a través de un rallado y la imagen del obstáculo es capturado a través de otro rallado con una cámara. Los rallados crean un patrón de difracción y la distancia puede ser calculada midiendo la posición de la difracción rayadas. En la interferometría holográfica un holograma es creada desde el obstáculo y el holograma es utilizado como una referencia del objeto. Cuando un holograma y el obstáculo original son iluminados con un haz de láser, su combinación crea un patrón de interferencia. La distancia puede ser calculada desde la distancia entre las líneas dentro del patrón.

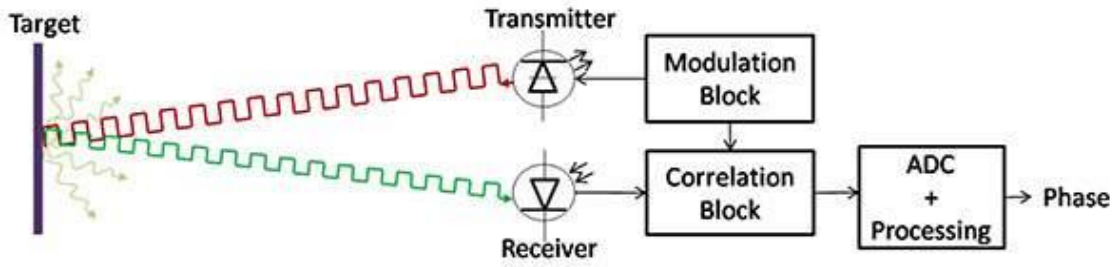
La difracción de Fresnel está basada en la iluminación de un rallado con una luz coherente, lo cual el rallado crea patrones de interferencia en distancias regulares, las cuales pueden ser para medir la distancia. En general, los

interferómetros son muy exactos, incluso en el rango de µm. El problema es que sin las distancias de referencia es imposible medir las distancias sin ambigüedades, y únicamente la diferencia en la distancia puede ser medida. Utilizando varias longitudes de onda también medidas absolutas pueden ser llevadas a cabo. Una aplicación de la interferometría es la medida de los movimientos de la corteza de la tierra.

2.1.2.2. Método de tiempo de vuelo (ToF)

Los métodos ToF se encuentran en el corazón de la detección remota de objetos utilizando ondas de luz, ultrasonido y tecnologías de radar. El principio de funcionamiento para estos sistemas ToF es relativamente simple. Un transmisor emite una señal modulada y el objetivo refleja una parte de la señal de vuelta a un receptor. La señal recibida se correlaciona con la señal transmitida por un procesador dedicado que mide el tiempo de vuelo y calcula el rango correspondiente al objetivo.

Figura 6. **Principio del método de tiempo de vuelo (ToF)**



Fuente: Simplifying Time-of-Flight Distance Measurements. https://www.digikey.com/en/articles/techzone/2017/jan/simplifying-time-of-flight-distance-measurements. Consultado: 5 de mayo de 2018.

Las dos principales ventajas del método de tiempo de vuelo (ToF) son:

- Los haces transmisores y receptores poseen una estructura coaxial, en otras palabras, el emisor envía el haz láser a través de un tubo metálico y el receptor situado alrededor del emisor, recibe la luz reflejada.
- La precisión en la medición no depende sobre la distancia.

Al utilizar este método, la señal transmitida puede utilizar modulaciones de pulso, amplitud o frecuencia. Cuando se emplea la modulación de pulso, el tiempo de vuelo es medido del pulso reflejado desde el obstáculo. En el método de la modulación de amplitud, se mide la diferencia de fase entre el seno modulado transmitido y el haz reflejado. En la técnica de la modulación de la frecuencia, la señal de frecuencia es barrida en algunas frecuencias de rangos limitados. Empleando técnicas heterodino, una frecuencia intermedia es generada, de la cual depende la distancia del objetivo. Los métodos más populares para la modulación son el de modulación por amplitud y pulso.

El método ToF cuenta con dos ejes ópticos, en los cuales el caso de un eje es para el transmisor y el otro para el receptor, pero por que los ejes ópticos están muy cercanos uno a otro, el método es clasificado como monocular.

2.2. Microcontrolador

Un microcontrolador es un dispositivo electrónico capaz de llevar a cabo procesos lógicos para desempeñar una tarea específica. Esta tarea puede ser programada por el usuario empleando un lenguaje de programación, siendo almacenada en su memoria. Está compuesto por tres componentes principales: unidad central de procesamiento, memorias y periféricos.

2.2.1. Arquitecturas

Entre los microcontroladores existen dos arquitecturas básicas de hardware.

- Arquitectura Von Neumann

Se caracteriza por tener una memoria única para el almacenamiento de datos y las instrucciones del programa. A esta memoria se accede a través de un sistema de buses único (control, direcciones y datos), en otras palabras, posee un único bus para conectar la memoria con el procesador.

- Arquitectura Harvard

En esta arquitectura, cada tipo de memoria posee un bus de datos, uno de direcciones y uno de control. Posee dos tipos de memorias, una contiene solamente las instrucciones del programa (Memoria de Instrucciones), y la otra solo almacena datos (Memoria de Datos). Esta es la arquitectura utilizada por los microcontroladores actualmente.

2.2.2. Estructura básica de un microcontrolador

En el mercado, existe una variedad amplia de microcontroladores, cada uno con características diferentes, por lo que no poseen una misma estructura interna. Sin embargo, todos ellos comparten componentes fundamentales, como se detalla a continuación.

- Registros

Espacio de memoria reservada, para almacenar los resultados de la ejecución de instrucciones, cargar datos desde la memoria externa o almacenarlos en ella.

- Unidad de control

Esta unidad se encarga de la lógica necesaria para la decodificación y ejecución de las instrucciones, el control de los registros, la ALU, los buses y otros tipos que se requieren de interacción con el procesador. La unidad de control es uno de los elementos fundamentales que determinan las prestaciones del procesador.

- Unidad aritmeticológica (ALU)

Unidad encargada de la realización de operaciones tales como sumas, restas y operaciones lógicas relacionadas con el algebra Booleana.

- Buses

Son el medio de comunicación que utilizan los diferentes componentes del procesador para intercambiar información entre sí. Existen tres tipos de buses:

  - Dirección. Se utiliza para seleccionar al dispositivo con el cual se quiere trabajar o en el caso de las memorias, seleccionar el dato que se desea leer o escribir.
  - Datos. Se utiliza para mover los datos entre los dispositivos de hardware (entrada y salida).

- Control. Se utiliza para gestionar los distintos procesos de escritura lectura y controlar la operación de los dispositivos del sistema.

- Memoria.

La memoria es un dispositivo de que permite el almacenamiento de información de forma permanente o de forma temporal. En la memoria de programas se utilizan diferentes tecnologías, y el uso de una u otra depende de las características de la aplicación a desarrollar, a continuación, se describen las cinco tecnologías existentes, que mayor utilización tienen o han tenido:

- Memoria ROM. Almacena información de forma permanente o invariante.
- Memoria EPROM. Memoria reprogramable por medio de la exposición a luz ultravioleta.
- Memoria EEPROM. Memoria reprogramable eléctricamente.
- Memoria Flash. Memoria reprogramable con las mismas tensiones de alimentación del microcontrolador.
- Memoria RAM. Memoria de almacenamiento temporal de información.

- Periféricos

Se considera periférico al conjunto de dispositivos que, sin pertenecer al núcleo fundamental, formado por la unidad central de procesamiento (CPU) y la memoria central, permitan realizar operaciones de entrada/salida (E/S) complementarias al proceso de datos que realiza la CPU. Los periféricos permiten la comunicación del microcontrolador con fuentes externas, a

continuación, se describen algunos de los periféricos que con mayor frecuencia son utilizados en los microcontroladores.

- Entradas y salidas de propósito general. También conocidos como puertos de E/S. Algunos puertos de E/S poseen características especiales que le permiten manejar salidas con determinados requerimientos de corriente, o incorporan mecanismos especiales de interrupción para el procesador.
- Temporizadores y contadores. Son circuitos sincrónicos para el conteo de los pulsos que llegan a su poder para conseguir la entrada de reloj. Si la fuente de un gran conteo es el oscilador interno del microcontrolador es común que no tengan un pin asociado, y en este caso trabajan como temporizadores. Por otra parte, cuando la fuente de conteo es externa, entonces tienen asociado un pin configurado como entrada, este es el modo contador.
- Conversor analógico/digital (ADC).
- PWM. Puertos con capacidad de modular una señal por ancho de pulso

- Puertos de comunicación

  - Puerto serial

Este tipo de periférico está presente en casi cualquier microcontrolador disponible en el mercado, por lo general en forma de UART (*Universal Asynchronous Receiver Transmitter*) o USART (*Universal Synchronous Asynchronous Receiver Transmitter*). La principal función de este periférico es la comunicación con otro microcontrolador o con una PC y en la mayoría de los

casos hay que agregar circuitos externos para completar la interfaz de comunicación. Otra función destacada es la comunicación con otros periféricos que permiten completar el funcionamiento del microcontrolador.

- Interfaz periférica serial (SPI)

Es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. El bus de interfaz de periféricos serie o bus SPI es un estándar para controlar casi cualquier dispositivo electrónico digital que acepte un flujo de bits serie regulado por un reloj (comunicación sincrónica).

- I2C

Corresponde a las siglas del nombre en inglés *Inter-Integrated Circuit*, se usa para la comunicación entre partes de un circuito. Está diseñado como un bus maestro-esclavo, cumple las mismas funciones que el SPI, pero requiere menos señales de comunicación y cualquier nodo puede iniciar una transacción.

2.2.3. Programación del microcontrolador

El microcontrolador ejecuta el programa cargado en la memoria flash. Esto se denomina el código ejecutable y está compuesto por una serie de unos y ceros. Dependiendo de la arquitectura del microcontrolador, el código binario está compuesto por palabras de 12, 14 o 16 bits de longitud. Cada palabra se interpreta por la CPU como una instrucción a ser ejecutada durante el funcionamiento del microcontrolador. Todas las instrucciones que el microcontrolador puede reconocer y ejecutar se les denominan colectivamente conjunto de instrucciones.

2.2.3.1. Tipos de lenguajes

Existen dos tendencias en la programación de microcontroladores, las cuales se describen a continuación.

- Lenguaje de bajo nivel o ensamblador.

En este lenguaje las instrucciones ejercen un control directo sobre el hardware y está condicionado por la estructura física de los componentes que poseen. Se parece al lenguaje de máquina, este tipo de programación es ideal para aprovechar al máximo los recursos disponibles, ya que permite controlar con detalle todos los procesos puestos en marcha dentro del chip.

- Lenguajes de alto nivel.

Uno de los lenguajes de alto nivel más populares para programar microcontroladores es el lenguaje C o lenguajes basados en este. Poseen sintaxis mucho más amigables con el lenguaje usado por los humanos lo que hace que su programación sea más sencilla. En lenguajes de programación de alto nivel, varias instrucciones en ensamblador se sustituyen por una única sentencia. Una de sus principales ventajas es que el programador ya no tiene que conocer el conjunto de instrucciones o características del hardware del microcontrolador utilizado.

2.2.3.2. Comparación de lenguajes

Los lenguajes de alto nivel no reciben ese nombre por ser considerados mejores que los lenguajes de bajo nivel. Cualquier programa elaborado en un lenguaje de alto nivel, puede realizarse en un lenguaje de bajo nivel. Un lenguaje

de alto nivel empleara un número reducido de instrucciones para realizar una acción; mientras que un lenguaje de bajo nivel emplea múltiples líneas de instrucciones para realizar la misma acción, como se muestra en la figura tal. En ciertos casos la capacidad de comunicarse directamente con el procesador de la computadora en un lenguaje de bajo nivel puede resolver los problemas que las capas de abstracción de un lenguaje de alto nivel pueden dificultar. Un lenguaje de alto nivel puede tratar sólo con un sistema operativo determinado o con un programa determinado. Estos programas son muy útiles para cualquiera que quiera manipularlos sin tener que saber cómo funciona, pero un lenguaje así no sería útil para alguien que intente escribir un programa propio y que necesita un lenguaje de nivel más bajo. Por lo tanto, un programador elige un lenguaje dependiendo del trabajo que necesite llevar a cabo.

2.3. Protocolos de comunicación *machine-to-machine* (M2M)

El concepto *machine-to-machine* se refiere al intercambio de información o comunicación en forma de datos entre dos máquinas remotas. Todo entorno M2M debe de contar con los siguientes elementos.

- Las máquinas gestionan la información entre ellas.
- Los dispositivos M2M que se conectan a una máquina remota y proveen de comunicación al servidor.
- El servidor que gestiona el envío y la recepción de la información, y la red de comunicación por cable o a través de redes inalámbricas.

Esta comunicación se realiza de manera telemática (por la convergencia entre las tecnologías de las Telecomunicaciones y de la Informática) a través de redes privadas e inalámbricas. Estas herramientas aumentan la productividad, automatización y eliminación manual.

2.3.1. Protocolo MQTT

Message Queue Telemetry Transport (MQTT, por sus siglas en inglés, o Transporte de telemetría de cola de mensajes, en español), es un protocolo de comunicación M2M, orientado a la comunicación de sensores, ligero en tanto que requiere de pocos recursos para su ejecución y con una implementación basada, como tantos otros protocolos, en un mecanismo de publicación también conocido como suscripción. La arquitectura de MQTT sigue una topología de estrella, con un nodo central que hace de servidor, con una capacidad de hasta 10 000 clientes. Este servidor actúa como canal de comunicación entre los dispositivos que publican la información y los dispositivos que reciben la notificación de dicha información. Este intermediario o broker tiene la misión de ordenar y racionalizar la comunicación entre los publicadores y los consumidores de los datos.

2.3.1.1. Principios de funcionamiento del protocolo MQTT

- El broker es el encargado de gestionar la red y de transmitir los mensajes, para mantener activo el canal, los clientes mandan periódicamente un paquete (PINGREQ) y esperan la respuesta del broker (PINGRESP). La comunicación puede ser cifrada entre otras muchas opciones.
- La comunicación se basa en *topics* (temas), que el cliente que publica el mensaje crea y los nodos que deseen recibirlo deben subscribirse a él. La comunicación puede ser de uno a uno, o de uno a muchos. Un *topic* se representa mediante una cadena y tiene una estructura jerárquica. Cada jerarquía se separa con '/'.

2.3.1.2. Ventajas del protocolo MQTT

La amplia difusión de este protocolo se fundamenta, sin duda, en algunos aciertos en su diseño que lo convierten en óptimo para el despliegue de soluciones bajo el paradigma del Internet de las cosas (IoT) en determinados escenarios.

- La existencia de un broker o elemento intermedio que ordena la comunicación simplifica extraordinariamente el desarrollo de esta. Más aún cuando existen soluciones de terceros ya implementadas, algunas de uso libre, y contrastada solvencia, que pueden reutilizarse sin ningún esfuerzo de desarrollo específico para su integración, tales como RabbitMQ, Mosquitto, ZeroMQ, entre otros.
- La existencia de un elemento intermediario, junto con la simplicidad de su implementación, además, dota a la solución de una gran escalabilidad, puesto que, con aumentar recursos en un único elemento de la arquitectura, la solución puede crecer hasta alcanzar tasas de rendimiento altas.
- La existencia de múltiples canales, también llamados etiquetas, que permite una muy eficiente segmentación de la información y el tráfico de datos.
- Soporte directo para protocolos de calidad de servicio (QoS) que garantiza la fiabilidad y la integridad de la comunicación.

2.4. IoT

El internet de las cosas (en inglés, *Internet of Things*, IoT) es un concepto que se refiere a la interconexión digital de objetos cotidianos con Internet. En otras palabras, el internet de las cosas es una red de objetos físicos que utiliza

sensores y API para conectarse e intercambiar datos por internet. Las implementaciones del IoT utilizan diferentes modelos de conectividad, cada uno de los cuales tiene sus propias características. Los cuatro de los modelos de conectividad descritos por la Junta de Arquitectura de Internet incluyen: *Machine-to-Machine* (dispositivo a dispositivo), *Device-to-Cloud* (dispositivo a la nube), *Device-to-Gateway* (dispositivo a puerta de enlace) y *Back-End Data-Sharing* (intercambio de datos a través del *back-end*). Estos modelos destacan la flexibilidad en las formas en que los dispositivos de la IoT pueden conectarse y proporcionar un valor para el usuario.

2.4.1. Principales características del IoT

- Conectividad ubicua. La conectividad generalizada, de bajo costo y alta velocidad, sobre todo a través de servicios y tecnología inalámbricos con y sin licencia, hace que casi todo sea conectable.
- Adopción generalizada de redes basadas en el protocolo IP. El protocolo IP se ha convertido en el estándar dominante para la creación de redes y ofrece una plataforma bien definida y ampliamente implementada en software y herramientas que se pueden incorporar en una variedad de dispositivos de forma fácil y económica.
- Miniaturización. Los avances logrados en la fabricación permiten incorporar tecnología de cómputo y comunicaciones de vanguardia en objetos muy pequeños.
- Avances en el análisis de datos. La existencia de algoritmos y rápido aumento de la potencia de cálculo, el almacenamiento de datos y los servicios en la nube permiten agregar, correlacionar y analizar grandes cantidades de datos. Estos conjuntos de datos grandes y dinámicos ofrecen nuevas oportunidades para extraer información y conocimiento.

- Computación en la nube. La computación en la nube aprovecha recursos informáticos remotos conectados en red para procesar, gestionar y almacenar datos. Este paradigma permite que dispositivos distribuidos, con pocos recursos, interactúen con potentes sistemas de soporte que brindan capacidades analíticas y de control.

2.5. Web *Frameworks*

Un *web framework* o también llamado *web application framework* es un tipo de software diseñado para el soporte de desarrollo de aplicaciones web incluyendo, servicios web (*web services*), recursos web (*web resources*) e interfaces de programación de aplicaciones web (web APIs). Los *webs frameworks* proveen una manera estándar para realizar y desarrollar aplicaciones web. Tienen como objetivo automatizar la sobrecarga asociada a las actividades comunes que se realiza en los entornos de desarrollo web, tales como conexiones a bases de datos, uso de plantillas, administradores de sesiones y reutilización de código. Aunque están orientados al desarrollo de sitios web dinámicos, estos también se pueden aplicar a sitios web estáticos.

2.5.1. Razones por las que se debe utilizar un *framework*

- Reutilización del código. Los proyectos tienen partes en común necesarias para el funcionamiento, como conexiones con bases de datos, validación de formularios o seguridad. Al utilizar un *framework* evita reprogramar estas funciones de manera que resulte más fácil centrarse en programar las aplicaciones.
- Utilizar buenas prácticas. Los *frameworks* están basados en patrones de desarrollo, por lo general MVC (Modelo-Vista-Controlador, por sus siglas en inglés) que ayudan a separar los datos y la lógica de negociación de la

interfaz de la aplicación con el usuario, obteniendo como resultado un entorno de funciones más ordenado.

- Implementar código ya desarrollado: Permite incorporar funciones avanzadas creadas por otros desarrolladores que, para el usuario común, resultaría más difícil codificar, esto implica un ahorro de tiempo de programación.
- Desarrollo más rápido. El uso de un *framework* permite que el desarrollo de aplicaciones sea más rápido, más limpio y seguro, por medio de la implementación de estándares.

2.6. Arduino (C++)

El lenguaje de programación Arduino está basado en C++. El lenguaje utilizado no es un C++ puro, este es una adaptación que deriva de avr-libc, el cual es un software de código libre que provee de una librería de C de alta calidad para usar con GCC (compilador de C y C++) en los microcontroladores AVR de Atmel y muchas utilidades específicas de estos.

Aunque de forma común se hable de Arduino como un lenguaje propio de programación, esto no es cierto, la programación se hace en C++, pero Arduino brinda una api o core que facilita el acceso, la configuración y el control de los pines de entrada, salida y de los puertos de comunicación, así como otras librerías para operaciones específicas. Esto lo realiza a través de su propio IDE, ya que incluye las librerías de forma automática para que no sean necesarias declararlas expresamente.

Una de las principales diferencias frente a C++ estándar, es la estructura del programa. En el lenguaje utilizado por Arduino son necesarias dos funciones: setup y loop. La función setup es la encargada de recoger la información

necesaria para el funcionamiento del microcontrolador. En esta parte se inicializan las librerías, pines y distintos tipos de comunicación que se emplearan en el programa. La función loop es la que contiene las instrucciones, funciones, y operaciones que se ejecutaran de forma cíclica. Ambas funciones son necesarias para que el programa funcione.

2.7. Python

Python es un lenguaje de *scripting* independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License,2 que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

Python se caracteriza por ser un lenguaje versátil y con muchas características, entre las que se incluyen:

- La cantidad de librerías que contiene, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas habituales sin necesidad de tener que programarlas desde cero.
- Puede ser utilizado para distintas tareas. Entre las principales se encuentran el análisis de datos, desarrollo de aplicaciones web y de escritorio.

- La sencillez y velocidad con la que se crean los programas. Un programa en Python puede tener de 3 a 5 líneas de código menos que su equivalente en Java o C.
- Se puede desarrollar en múltiples plataformas, entre las principales se encuentran Windows, Linux y Mac.
- Python es gratuito, incluso para propósitos empresariales.

3. COMPONENTES DEL PROTOTIPO

En el siguiente capítulo se exponen los componentes que se utilizarán en la elaboración del prototipo del sistema de sensores, para adaptarlo a cualquier ambiente donde se desee medir deformaciones. Para que el sistema sea versátil y adaptable se ha planteado un diseño modular. También se presentan los diagramas esquemáticos de los circuitos electrónicos, el algoritmo de funcionamiento del dispositivo, las conexiones entre módulos, el diagrama de uso de la aplicación y, por último, una cotización de los componentes del proyecto.

3.1. Descripción del prototipo

El prototipo por realizar es un sistema para medir deformaciones superficiales, este sistema se compone de un dispositivo para medir deformación y una aplicación web para visualizar los datos obtenidos. El dispositivo debe ser capaz de medir las deformaciones a una distancia no mayor a un metro de los objetos con una precisión de milímetros. La interfaz debe mostrar las deformaciones registradas por segundo, así como también el dispositivo contará con una pantalla OLED para seleccionar las distintas opciones que se mostraran en el menú.

El dispositivo contará con un menú para ejecutar operaciones básicas como comprobar la conectividad, seleccionar el número del sensor y otras opciones que se detallan a continuación:

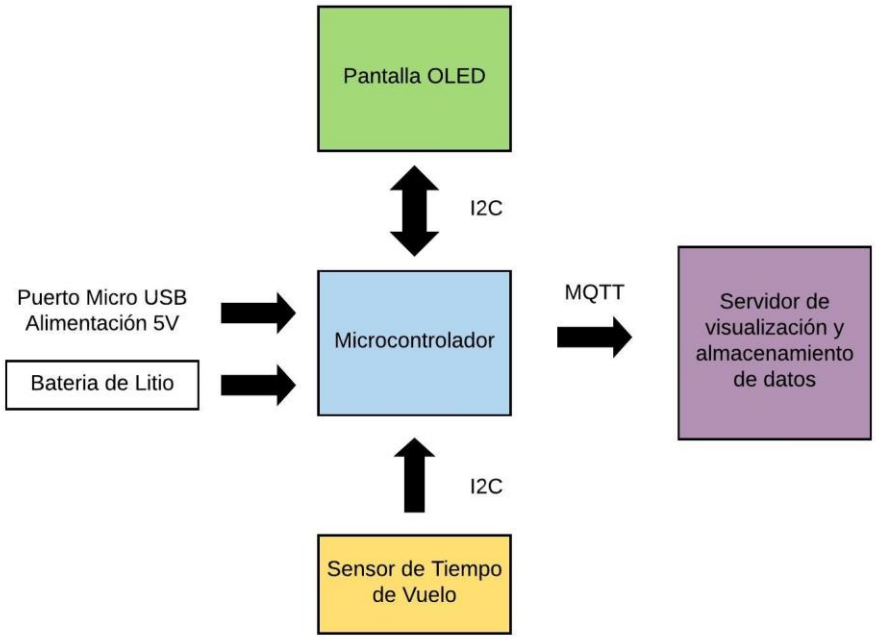
- Opción 1: Conexión wifi. Esta opción permite comprobar que el dispositivo está conectado a la red, para poder transmitir los datos a la aplicación web, almacenarlos en la base de datos y utilizarlos posteriormente.
- Opción 2: Número de sensor. Al seleccionar esta opción, se podrá elegir el número el cual se identificará el sensor.
- Opción 3: Distancia. Esta opción permite colocar el sensor en su lugar y medir la distancia inicial para iniciar a observar las variaciones (deformación) que se den a partir de ese punto.
- Opción 4: Inicio. Con esta opción se inicia el proceso para medir deformaciones.
- Opción 5: Prueba. Esta opción permite que el sensor muestre en su pantalla la distancia, de manera continua, que el sensor percibe, sin enviar datos a la aplicación. Facilita la realización de pruebas y comprobar que el sensor esté funcionando correctamente.
- Opción 6: Nivel de batería. Muestra en la pantalla el nivel de la batería del sensor.
- Opción 7: *Reset*. Permite reiniciar el dispositivo, ya sea para renovar la conectividad a través de wifi o solo reiniciar el dispositivo.

El operario debe saber navegar y seleccionar entre estas opciones del menú para iniciar con los experimentos y comprobar que el dispositivo esté conectado a la red para enviar los datos a la aplicación web para su visualización. Para esto, el prototipo cuenta con tres botones, de los cuales dos servirán para el desplazamiento del menú (arriba y abajo) y uno para seleccionar las opciones que se presentan.

El prototipo se conecta a un servidor utilizando el protocolo MQTT, para almacenar los datos recolectados en una base de datos. La aplicación se conecta directamente a la base de datos, para mostrarlos en un navegador web donde el

operario podrá exportar estos para su análisis posterior. El diagrama de bloques del prototipo del sistema se muestra en la figura 7.

Figura 7. **Diagrama de bloques del prototipo**



Fuente: elaboración propia, empleando Adobe Photoshop®.

3.2. Descripción de módulos electrónicos

A continuación, se detallan las características de los módulos del prototipo.

3.2.1. Microcontrolador

El microcontrolador que se propone para el diseño de los sensores es el ESP8266 Wifi, el cual está contenido en la placa de desarrollo *Adafruit Feather*

HUZZAH ESP8266. Este microcontrolador cuenta con un procesador L106 de 32 bits RISC basado en el Tensilica Xtensa Diamond Standard 106Micro que trabaja a 80 MHz. Es programable mediante el IDE de Arduino. Cuenta con un integrado USB-Serial SiLabs CP2104, con el cual puede se le puede cargar el código a una tasa de 921600 baudios para acelerar el tiempo de desarrollo. La placa de desarrollo cuenta con auto reinicio para evitar los reinicios manuales por medio de botones. Además, también tiene un cargador de baterías LiPo integrado, lo que permite utilizarlo con baterías y también mediante el puerto micro USB.

3.2.1.1. Especificaciones técnicas

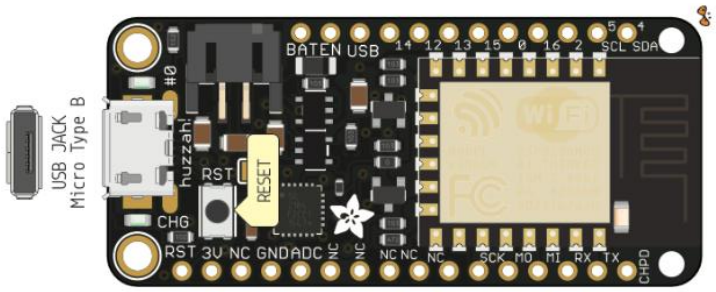
A continuación, se describen las especificaciones técnicas del microcontrolador utilizado.

Tabla I. **Características técnicas del microcontrolador y la placa de desarrollo**

Microcontrolador	**ESP8266 Wifi**
Dimensiones	51 x 23 x 8 mm (sin los pines soldados)
Peso	9.7 gramos
Memoria Flash	4 MB (32 MBit)
Estándares Wifi	802.11 b/g/n
Voltaje de operación	3.3 V
Voltaje de alimentación	5 V
Pines GPIO	9 (pueden ser utilizados como I2C y SPI)
Pines análogos	1 (1.0 V máximo de entrada)
Corriente máxima para los pines	12 mA
Velocidad del reloj	80 MHz

Fuente: Adafruit Feather HUZZAH with ESP8266.
https://www.adafruit.com/product/2821. Consulta: 9 de junio de 2018.

Figura 8. **Adafruit Feather HUZZAH ESP8266 y puertos de programación**



Fuente: *Adafruit Feather* HUZZAH ESP8266.
https://learn.adafruit.com/adafruit-feather-huzzah-esp8266/pinouts. Consulta: 9 de junio de 2018

3.2.2. Pantalla OLED

Para la interfaz de comunicación del sensor con el usuario, se propone el uso de la pantalla OLED de la familia *FeatherWing* de *Adafruit*. Este dispositivo contiene una pantalla OLED monocromática de 128x32 y cuenta con 3 botones con los cuales se puede interactuar. La diagonal de la pantalla mide 1 pulgada, su principal característica es que además de ser pequeña es muy legible, esto debido al contraste alto que presentan las pantallas OLED.

La pantalla está constituida por 128x32 pixeles OLED individuales blancos, y debido a que la pantalla produce su propia luz, no es requerida una luz de fondo como la que es utilizada en las pantallas LCD. Esto, a su vez, reduce la energía necesaria para que la pantalla funcione. Por ello, puede proporcionar un mayor contraste. La pantalla utiliza solo 2 pines I2C para comunicarse con el microcontrolador. Gracias a su diseño modular puede adaptarse a la placa de desarrollo *Adafruit Feather* HUZZAH ESP8266.

3.2.2.1. Especificaciones técnicas

- Pantalla conformada por 128 filas y 32 columnas de pixeles
- Cuenta con 3 botones programables
- Botón de *reset*
- Voltaje de alimentación de circuitos lógicos, 1,65 a 3,3 V.
- Voltaje de alimentación de la pantalla, 7 a 7,5 V
- Temperatura de operación, -40 a 70 °C

Tabla II. **Asignación de pines para la pantalla OLED *FeatherWing***

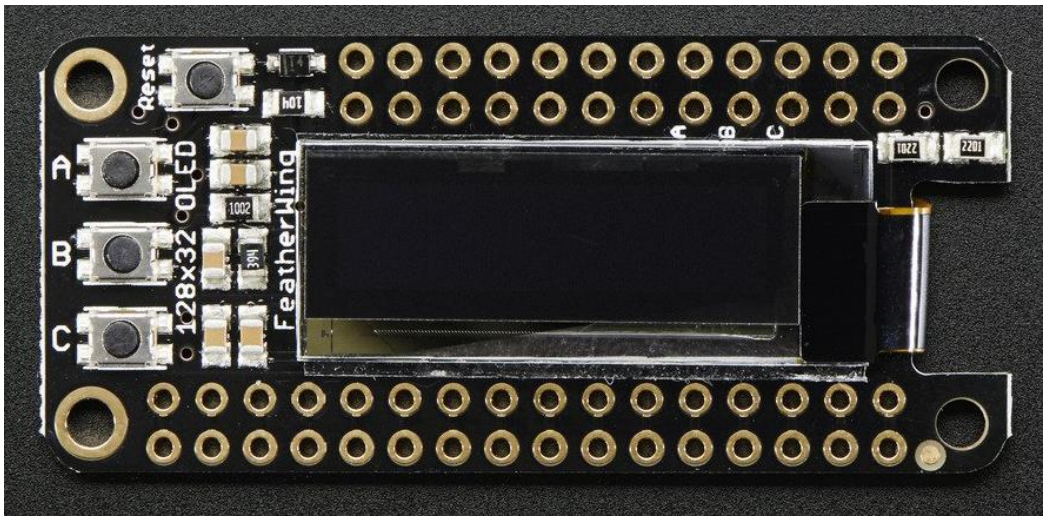
Pin	Nombre	Descripción
1	C2P	Terminal positivo del condensador inversor
2	C2N	Terminal negativo del condensador de impulso
3	C1P	Terminal positivo del condensador inversor
4	C1N	Terminal negativo del condensador de impulso
5	VBAT	Fuente de alimentación para el circuito convertidor DC / DC
6	VBREF	Voltaje de referencia para el circuito convertidor DC / DC
7	VSS	Tierra del sistema OEL
8	VDD	Fuente de alimentación para la lógica
9	RES#	*Reset* para el controlador
10	SCL	Señal de reloj para el bus I2C
11	SDA	Señal de datos para el bus I2C
12	IREF	Referencia actual para el ajuste de brillo

Continuación de la tabla II.

13	VCOMH	Voltaje de salida de alto nivel para la señal COM
14	VCC	Fuente de alimentación para el panel OEL

Fuente: elaboración propia.

Figura 9. **Pantalla OLED *FeatherWing***



Fuente: FeatherWing OLED. https://www.adafruit.com/product/2900.
Consultado: 10 de junio de 2018.

3.2.3. Sensor de distancia de tiempo de vuelo

Para medir las deformaciones que se produzcan en los objetos se propone el sensor de tiempo de vuelo Adafruit VL53L0X. Este contiene una fuente láser invisible muy pequeña y un sensor para medir cuánto tiempo ha tardado la luz en ser emitida por el láser hasta regresar al sensor. Debido a que utiliza una fuente

de luz muy estrecha, este sensor es el indicado para determinar la distancia de la superficie que se coloque directamente en frente de él, a diferencia de los sonares que hacen rebotar ondas ultrasónicas, el cono de detección es muy estrecho. A diferencia de los sensores de distancia IR que intentan medir la cantidad de luz rebotada, el VL53L0X es mucho más preciso y no presenta problemas de linealidad o “doble imagen” en los que no se puede determinar si un objeto está muy lejos o muy cerca.

3.2.3.1. Especificaciones técnicas

A continuación, se describen las principales características técnicas del sensor de tiempo de vuelo:

- Alimentación de 3 a 5 V.
- Interfaz de comunicación I2C.
- Dirección I2C programable.
- Dependiendo de la iluminación ambiental y la distancia, se obtendrá una precisión del 3 al 12% (una mejor iluminación y superficies brillantes brindan mejores resultados).
- Dimensiones: 21,0 x 18,0 x 2,8 mm.
- Distintos de funcionamiento: modo por defecto y modo de largo alcance.
- Láser: 940 nm VCSEL.
- Mide el rango absoluto hasta 2 m.
- Funciona en niveles altos de luz ambiental infrarroja.

Tabla III. **Capacidades de alcance máximo con un tiempo de 33 ms**

Nivel de reflectancia objetivo	Condiciones	En interiores	Cubierto al aire libre
Objetivo Blanco (88 %)	Típicas	200 cm	80 cm
	Mínimas	120 cm	60 cm
Objetivo Gris (17 %)	Típicas	80 cm	50 cm
	Mínimas	70 cm	40 cm

Fuente: elaboración propia.

Tabla IV. **Precisión de alcance**

	En interiores			Al aire libre		
Reflectancia	Distancia	33 ms	66 ms	Distancia	33 ms	66 ms
Objetivo Blanco (88 %)	120 cm	4 %	3 %	60 cm	7 %	6 %
Objetivo Gris (17 %)	70 cm	7 %	6 %	40 cm	12 %	9 %

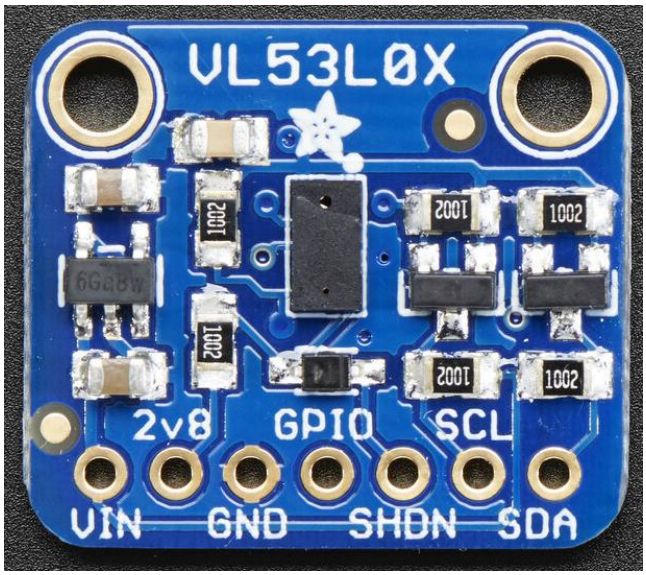
Fuente: elaboración propia.

Tabla V. **Asignación de pines para sensor de distancia de tiempo de vuelo VL53L0X**

Pin	**Nombre**	**Descripción**
1	VIN	3 a 5 VDC
2	2v8	Salida 2,8 V 100 mA
3	GND	Tierra
4	GPIO	Pin de propósito general
5	SHDN	Pin para apagar el sensor
6	SCL	Pin de reloj I2C
7	SDA	Pin de datos I2C

Fuente: elaboración propia.

Figura 10. **Sensor de distancia de tiempo de vuelo VL53L0X**



Fuente: VL53L0X Pinouts.
https://learn.adafruit.com/adafruit-vl53l0x-micro-lidar-distance-sensor-breakout/pinouts.
Consultado: 10 de junio de 2018.

3.2.4. Fuente de alimentación

Debido a que los módulos involucrados son de bajo consumo se propone utilizar, como fuente de alimentación, una batería de litio capaz de brindar el voltaje de alimentación necesario para el funcionamiento del dispositivo, incrementando la portabilidad del dispositivo.

En la tabla VI se muestra el rango de voltajes de alimentación permisibles de todos los módulos involucrados en el sistema de sensores de deformación, los valores estandarizados de voltaje por cada uno y el consumo de energía.

Tabla VI. **Voltajes de alimentación de los módulos**

Módulo	Rango de voltaje de alimentación	Voltaje estandarizado	Consumo de energía
Microcontrolador	3,3 - 5 VDC	3,7 VDC	600 mW
Pantalla OLED	1,65 - 3,3 VDC	3,3 VDC	30 mW
Sensor de distancia ToF	3 - 5 VDC	3,3 VDC	20 mW

Fuente: elaboración propia.

Con base en las columnas correspondientes al voltaje estandarizado y al consumo de energía de la tabla VI, se calcula la capacidad de la batería. Se propone utilizar una batería de 1 200 mAh con un voltaje nominal de 3,7 VDC para alimentar el sensor de deformación, obteniendo un tiempo de funcionamiento de 6 horas por carga completa de la batería.

Se propone utilizar la batería de polímero de iones de litio fabricada por Adafruit con los parámetros antes descritos. Esta batería proporciona un voltaje de salida que varía entre 4,2 VDC, cuando está completamente cargada, hasta 3,7 VDC. Esta batería tiene una capacidad de 1 200 mAh para un total de aproximadamente 4.5 Wh. Incluye por defecto un conector JST-PH de 2 pines para que sea utilizada con los modelos de microcontroladores de la línea *FeatherWing*, como el que se propone en este proyecto.

El circuito de protección que incluye la batería evita que la tensión de la batería sea demasiado alta (sobrecarga) o baja (uso excesivo), lo que significa que la batería se cortará cuando esté completamente descargada a 3.0 V. Este circuito también protege a la batería contra cortocircuitos de salida.

3.2.4.1. Especificaciones técnicas

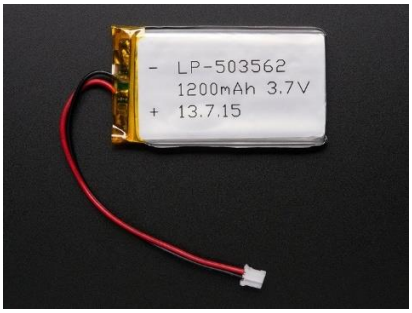
A continuación, se describen las especificaciones técnicas de la fuente de alimentación.

Tabla VII. **Características técnicas de la batería de litio**

Capacidad	Nominal: 1 200 mAh Mínima: 1 140 mAh
Voltaje nominal	3,7 V
Voltaje al final de la descarga	3,0 V
Voltaje de carga	4,2 V
Rango de temperatura de operación	Carga: 0 – 45 °C Descarga: -20 – 60 °C
Ciclos de vida	> 300 ciclos
Temperatura de almacenamiento	Durante 1 mes: -5 – 35 °C Durante 6 meses: -20 – 45 °C
Dimensiones	34 x 62 x 5 mm
Peso	23 g

Fuente: elaboración propia.

Figura 11. **Batería de litio con capacidad de 1 200 mAh y 3,7 V**



Fuente: Lithium Ion Polymer Battery - 3.7v 1200mAh.
https://www.adafruit.com/product/258.
Consultado: 15 de junio 2018

3.2.5. Módulo de almacenamiento y visualización de datos

Este módulo consiste en un ordenador de placa simple (SBC) que permitirá al usuario del sistema de sensores almacenar y visualizar los datos obtenidos. Se propone el uso de una *Raspberry* Pi modelo 3B+ para ser utilizada como servidor de base de datos y servidor de la aplicación web que desplegara los detalles de los sensores. La principal ventaja de utilizar esta SBC es que presenta un tamaño reducido comparado con un servidor dedicado o una computadora convencional, además de presentar distintos modos de conectividad ya que cuenta con soporte wifi 802.11ac de doble banda y puerto Gigabit Ethernet.

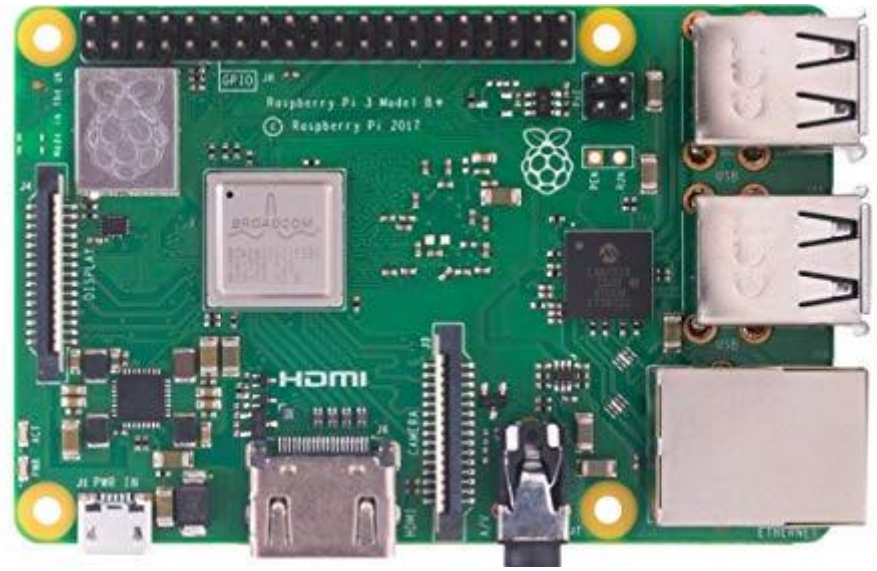
3.2.5.1. Especificaciones técnicas

A continuación, se describen las principales características técnicas de la *Raspberry* Pi modelo 3B+.

- Procesador: Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC
- Frecuencia de reloj: 1,4 GHz
- Memoria: 1 GB LPDDR2 SDRAM
- Bandas para conectividad Inalámbrica: 2,4 GHz / 5 GHz
- Protocolos inalámbricos soportados: IEEE 802.11.b/g/n/ac
- Conectividad de Red: Gigabit Ethernet
- Puertos
    - GPIO: 40 pines
    - HDMI
    - USB 2,0: 4 puertos
    - CSI (Cámara Raspberry Pi)
    - DSI (Pantalla táctil)
    - Puerto 3,5 mm para auriculares / video compuesto

- Micro SD
- Micro USB para alimentación

Figura 12. **Raspberry Pi modelo 3 B+**



Fuente: Raspberry Pi 3 Model B+.
https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/.
Consultado: 15 de junio 2018.

4. DIAGRAMAS Y DISEÑO FINAL DEL SISTEMA DE SENSORES

Este capítulo incluye la descripción del funcionamiento y los diagramas finales del diseño planteado, así como los materiales utilizados para la elaboración de este. El prototipo consiste en dos módulos: sensor y procesamiento de datos. A continuación, se explica a detalle en que consiste cada uno de dichos módulos.

4.1. Sensor

Este módulo incluye la base para el funcionamiento del sensor de deformación. En este se encuentra el microcontrolador, su sistema de alimentación, el sensor de tiempo de vuelo y los controles de este. La función de este módulo es albergar el microcontrolador y proporcionar las conexiones necesarias para la comunicación entre el sensor y el módulo de procesamiento de datos de manera eficiente y segura. A continuación, se enlistan los componentes utilizados en este módulo.

Tabla VIII. **Componentes del módulo de sensor**

Componente	Cantidad	Precio en dólares	Precio en quetzales
Microcontrolador ESP8266	1	$ 16,95	Q 131,02
Pantalla OLED FeatherWing	1	$ 14,95	Q 115,56
Sensor ToF VL53L0X	1	$ 14,95	Q 115,56
		Total	Q 362,14

Fuente: Cotización de módulos realizada en: https://www.adafruit.com. Tipo de cambio según: http://www.banguat.gob.gt/cambio/7,72983. Consulta: 15 de junio de 2018.

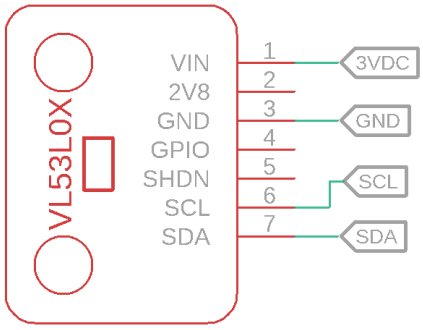
4.1.1. Diagramas esquemáticos de circuitos electrónicos

El módulo del sensor se compone de 4 diagramas esquemáticos. A continuación, se presenta la estructura y las interconexiones entre cada uno de ellos.

4.1.1.1. Diagrama de conexiones del sensor de distancia de tiempo de vuelo

El diagrama presentado a continuación muestra las conexiones necesarias para el utilizar el sensor de tiempo de vuelo. Este se alimenta con 3,3 VDC. Para la comunicación con el microcontrolador se utiliza el protocolo I2C, por lo que se requiere utilizar dos pines del microcontrolador: SCL, es el pulso de reloj que sincronizan el sistema, y SDA, que es la línea por donde se transmiten los datos entre los dispositivos.

Figura 13. **Conexiones del sensor de distancia de tiempo de vuelo**

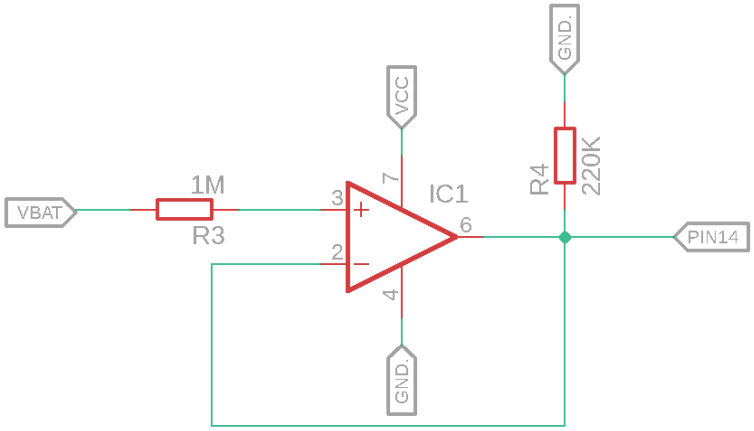


Fuente: elaboración propia, utilizando programa Eagle.

4.1.1.2. Diagrama de conexiones del circuito utilizado para medir el nivel de la batería

Consiste en un divisor de voltaje y un acople de impedancia con amplificador operacional. Al ser conectado al voltaje de la batería, en el punto medio de las resistencias, se obtiene el voltaje reducido en una proporción de 0,22, esto se conecta directamente al pin 14 del microcontrolador, de la forma en que se indica en la figura 14. El acople de impedancia está formado por un amplificador operacional con la configuración de un seguidor de voltaje, el cual proporciona a la salida la misma tensión que a la entrada, independientemente de la carga que se le acopla. En el microcontrolador se lee este voltaje y se muestra en la pantalla OLED el nivel actual de la batería.

Figura 14. **Circuito para medir el nivel de la batería**



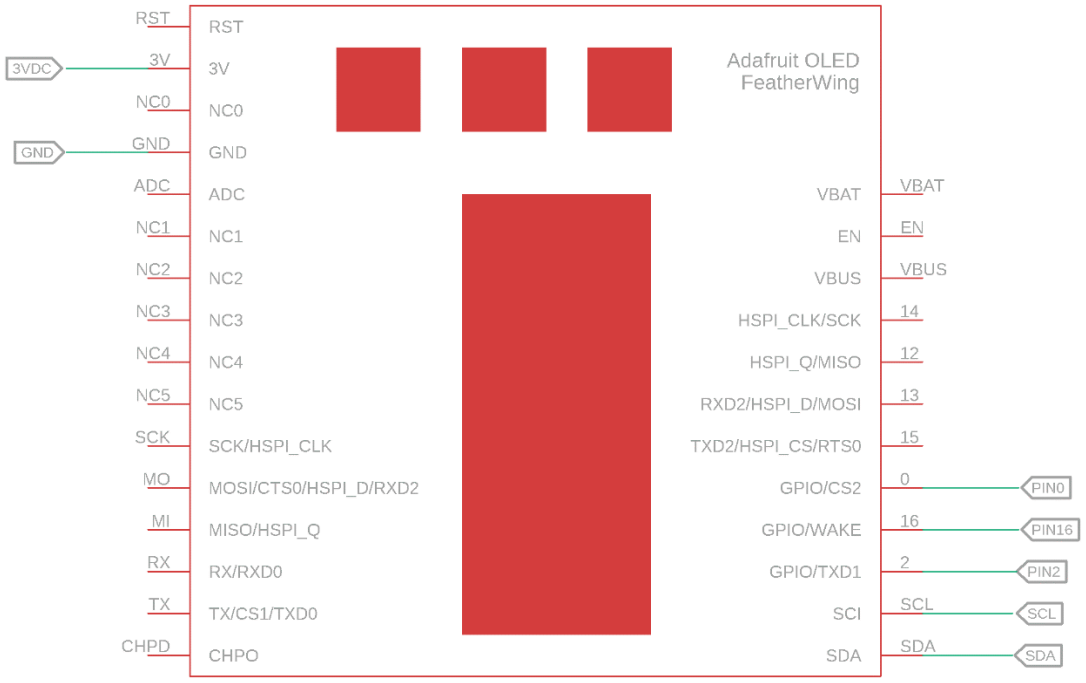
Fuente: elaboración propia, utilizando programa Eagle.

4.1.1.3. Diagrama de conexiones de la pantalla OLED

El diagrama presentado en la figura 14 muestra las conexiones necesarias para el funcionamiento de la pantalla OLED en donde se visualizará el control del sensor. Este módulo requiere una alimentación de 3,3 VDC y utiliza el protocolo I2C para comunicarse con el microcontrolador. Al tener una estructura modular diseñada para extender las funciones del microcontrolador, este módulo puede interconectar todos sus pines con el componente antes mencionado.

En la figura 15 se muestra las conexiones necesarias para el funcionamiento de este, siendo un total de 7 conexiones: 2 para la alimentación, 3 VDC y GND; 2 para la comunicación con el microcontrolador por medio de I2C para el despliegue de la pantalla; 3 pines para los botones utilizados para la navegación y selección de los distintos menús del sensor.

Figura 15. **Conexiones necesarias para el funcionamiento del módulo de la pantalla OLED**

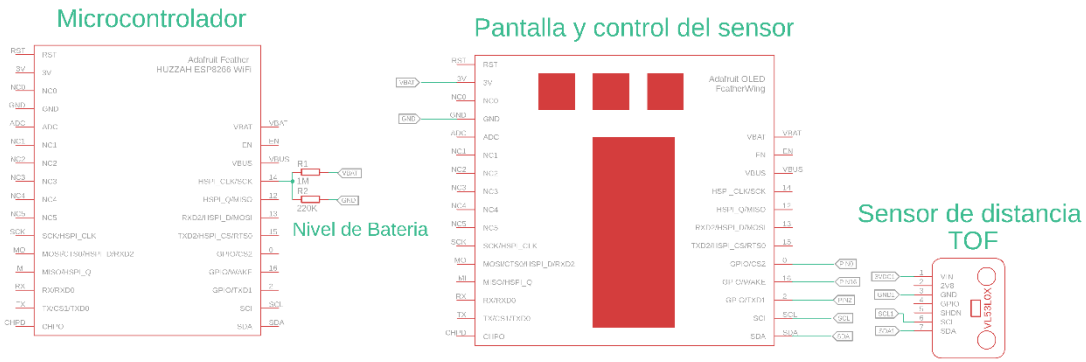


Fuente: elaboración propia, utilizando programa Eagle

4.1.1.4. Diagrama esquemático de los circuitos interconectados

En la figura 16 se muestra un diagrama esquemático de todos los circuitos y conexiones entre los módulos que componen el sensor de deformación.

Figura 16. **Diagrama esquemático general del sensor de deformación**



Fuente: elaboración propia, utilizando programa Eagle.

En la tabla IX se muestra una lista de los pines utilizados en el microcontrolador y la descripción de cada uno de ellos.

Tabla IX. **Pines utilizados en el microcontrolador**

Pines utilizados en el microcontrolador Adafruit Feather HUZZAH ESP8266	
Número de pin físico	Descripción
3V	Voltaje de 3,3 VDC
GND	Nivel de referencia, tierra
SDA	Señal de datos para el bus I2C
SCL	Señal de reloj para el bus I2C
VBAT	Voltaje de la batería conectada al jack JST
0	Entrada estado del botón 1
2	Entrada estado del botón 2
14	Entrada para medir nivel de la batería
16	Entrada estado del botón 3

Fuente: elaboración propia, utilizando mapa de pines proporcionado por Adafruit.

4.2. Procesamiento de datos

Para el módulo de procesamiento de datos se emplea una *Raspberry* Pi como servidor para la base de datos y la aplicación web para la visualización del sistema. El modelo utilizado es la *Raspberry* Pi 3 B+, el costo de esta SBC es de Q 595,00.

5. PROGRAMACIÓN Y DISEÑO DE SOFTWARE PARA EL SISTEMA DE SENSORES

El microcontrolador debe ser programado para controlar los módulos que componen el sensor de deformación. Por ello, se configuraron los controles necesarios para operar y visualizar las opciones de este. Además, para analizar los datos adquiridos mediante el sensor se realizó una aplicación web que muestra los datos obtenidos. Se utilizó un software compuesto por un broker, empleado para procesar los mensajes enviados por el sensor empleando el protocolo MQTT, una base de datos donde se almacenan los mensajes y la aplicación web. A continuación, se detalla el diseño de software para el sensor y el módulo de procesamiento de datos.

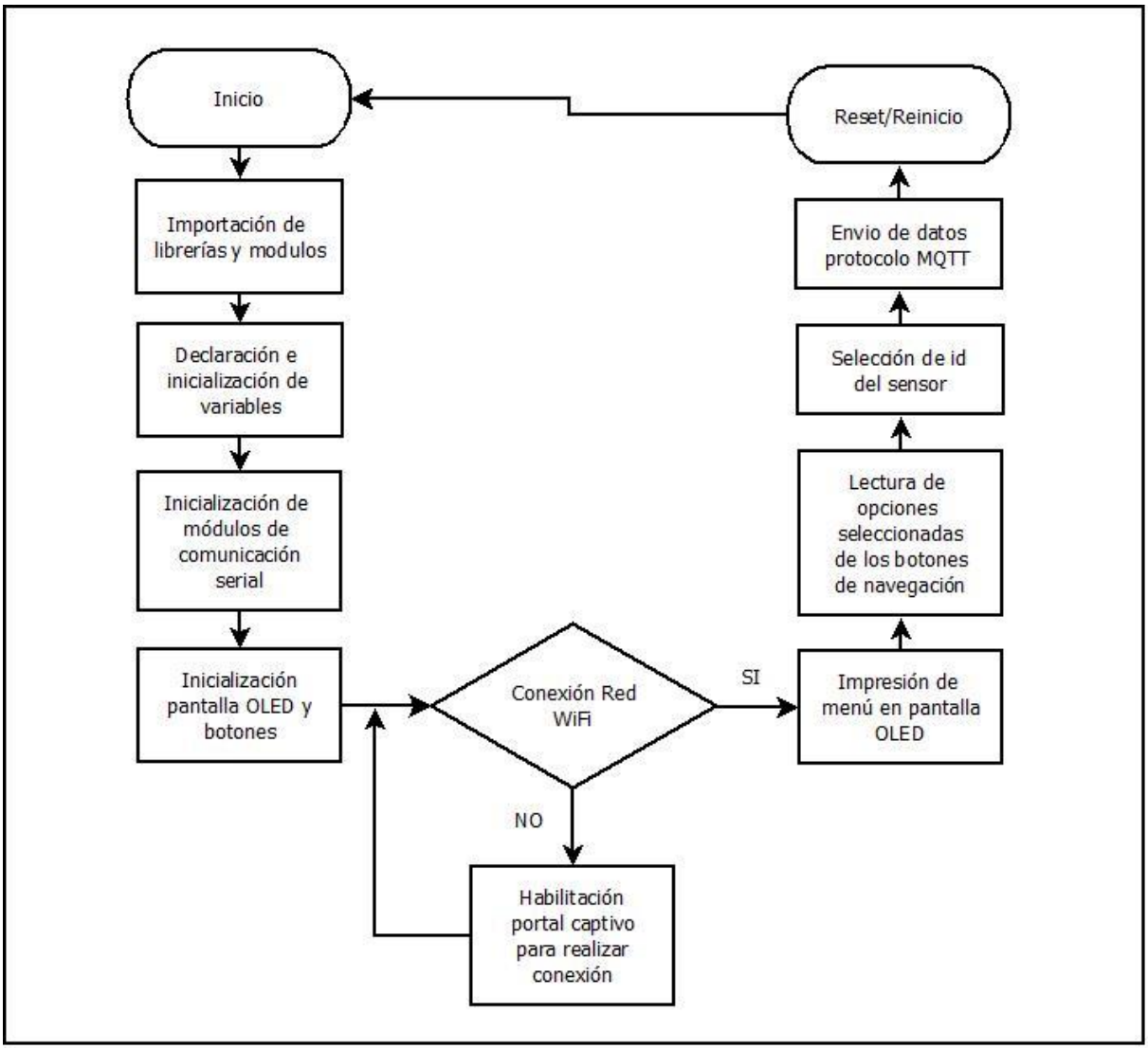
5.1. Diagramas de flujo

En esta sección se describe los procesos que realiza el prototipo para la adquisición, manejo y visualización de los datos.

5.1.1. Diagrama de flujo del sensor de deformación

En la figura 17 se muestra el flujo del programa de control del sensor de deformación.

Figura 17. **Diagrama de flujo del sensor de deformación**



Fuente: elaboración propia, empleando Dia.

5.1.1.1. Algoritmo del programa del sensor de deformación

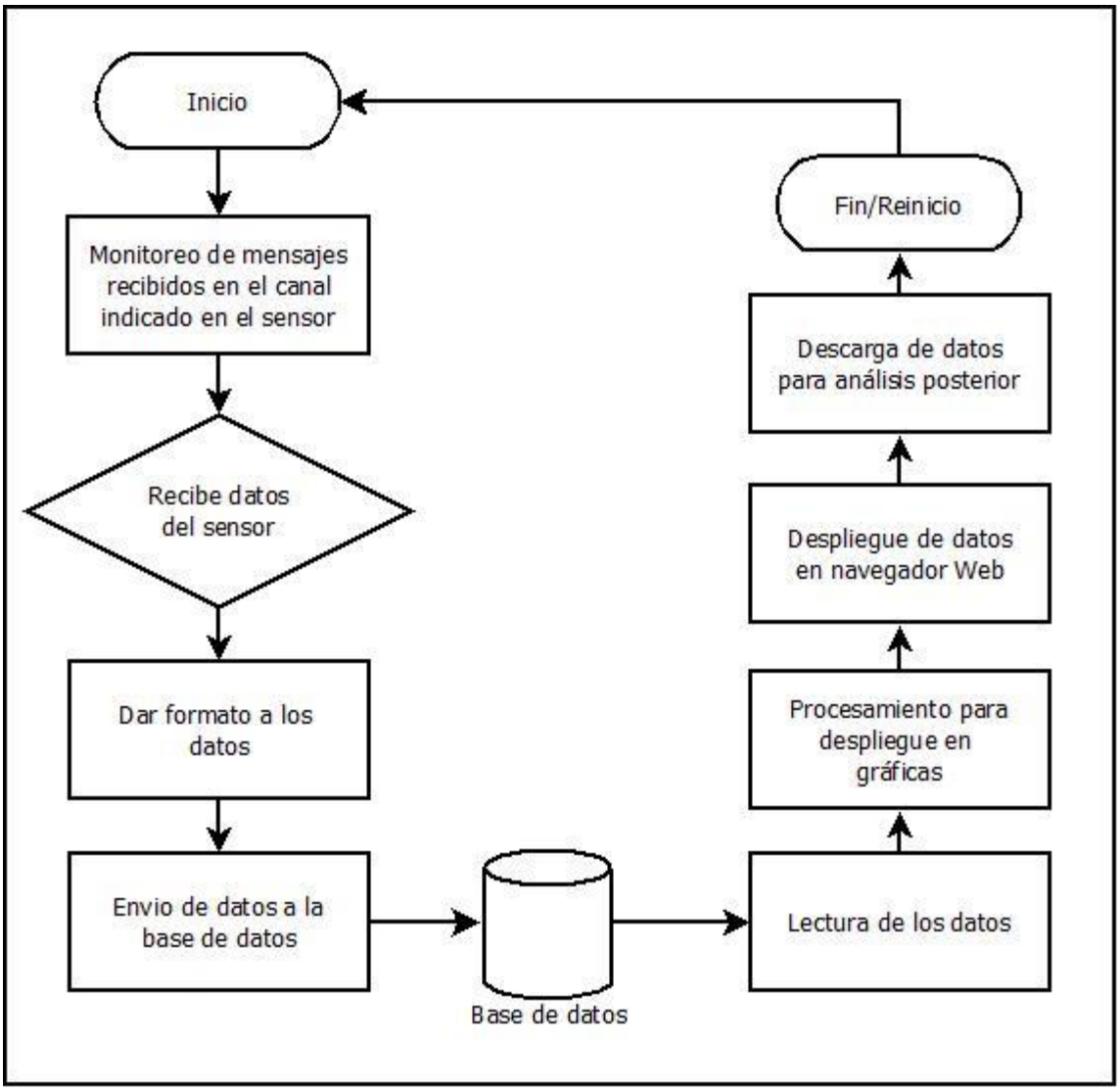
A continuación, se describe el algoritmo del programa de control propuesto para el sensor de deformación. El algoritmo se basa en el diagrama de flujo representado en la figura 17.

- Inicio del programa.
- Importación de librerías y módulos.
- Declaración e inicialización de variables.
- Inicialización de módulos de comunicación serial (velocidad de transmisión, puerto serial a utilizar, entre otros).
- Inicialización de la pantalla OLED (pines y dirección).
- Declaración e inicialización de pines de puertos a utilizar para los botones.
- Inicialización de la conexión de red vía wifi (reutilización de red guardada o habilitación de portal captivo para seleccionar conexión de red wifi).
- Inicialización del cliente para envió de mensajes utilizado protocolo MQTT.
- Impresión de menú en la pantalla OLED.
- Lectura de opciones seleccionadas empleando botones de navegación.
- Selección de número de identificación, id, del sensor.
- Envío de datos empleando protocolo MQTT.
- Fin de programa.

5.1.2. Diagrama de flujo del procesamiento de datos

En la figura 18 se muestra el funcionamiento y el procedimiento de la manipulación de estos datos empleando los *scripts* desarrollados.

Figura 18. **Diagrama de flujo del procesamiento de datos**



Fuente: elaboración propia, empleando Dia.

5.1.2.1. Algoritmo del flujo del procesamiento de datos

A continuación, se describe el proceso realizado para el procesamiento de los datos enviados por el sensor. Este algoritmo se basa en el diagrama presentado en la figura 18.

- Los datos son recibidos por el servidor MQTT en el canal indicado en el sensor.
- El script realizado para monitorear los elementos enviados por el canal de MQTT captura los datos del sensor.
- Se les da formato a los datos para poder ser ingresados a la base de datos.
- Se almacena la información en la base de datos.
- La aplicación web lee los datos que se almacenan en la base de datos.
- Procesa los datos para mostrarlos en las distintas gráficas.
- Despliega los datos en el navegador web.
- Proporciona la opción de descargarlos para análisis posterior.
- Fin del flujo.

5.2. Lenguajes de programación

Antes de iniciar el proceso para realizar las distintas partes de software que componen el sistema de sensores se evaluaron distintas opciones de lenguajes de programación para poder desarrollarlos. Después de analizar los tipos de lenguajes de programación, se eligió el lenguaje Arduino (C++) para el microcontrolador del sensor de deformación y el lenguaje Python para los *scripts* que componen el módulo de procesamiento de datos.

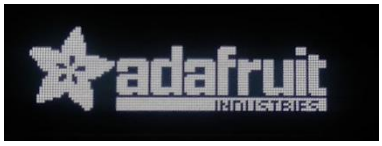
5.3. Funcionamiento del sensor de deformación

A continuación, se describen los pasos para el funcionamiento correcto del sensor que mide las deformaciones. Por medio de diagramas de flujo se realiza una representación gráfica de los pasos que deben ejecutarse para utilizar el dispositivo, cómo indicar el número de identificación del sensor, cómo mostrar el estado de este, y los pasos para iniciar el proceso de envío de datos para que sean procesados y mostrados por la aplicación web. De igual forma, se exponen imágenes de la interfaz de usuario del dispositivo, donde se indican las señales visuales y la navegación a través de ella. En los anexos se incluye el código utilizado para la programación del sensor de deformación.

5.3.1. Inicio del dispositivo

Para iniciar el dispositivo, se debe activar el interruptor de encendido si es que se desea utilizar con la batería como fuente, o bien, conectarlo empleando un cable micro USB. Al iniciar el dispositivo se mostrará la pantalla de carga con el logo de Adafruit Industries, como se muestra en la figura 19.

Figura 19. **Interfaz de usuario de inicio del sensor de deformación**



Fuente: elaboración propia, imágenes del prototipo empleando la pantalla OLED.

Si es la primera vez que se utiliza el dispositivo, este activará un portal captivo, empleando el módulo wifi como punto de acceso, propagando el SSID llamado SensorDeformacionESP8266, para habilitar el acceso al dispositivo y así

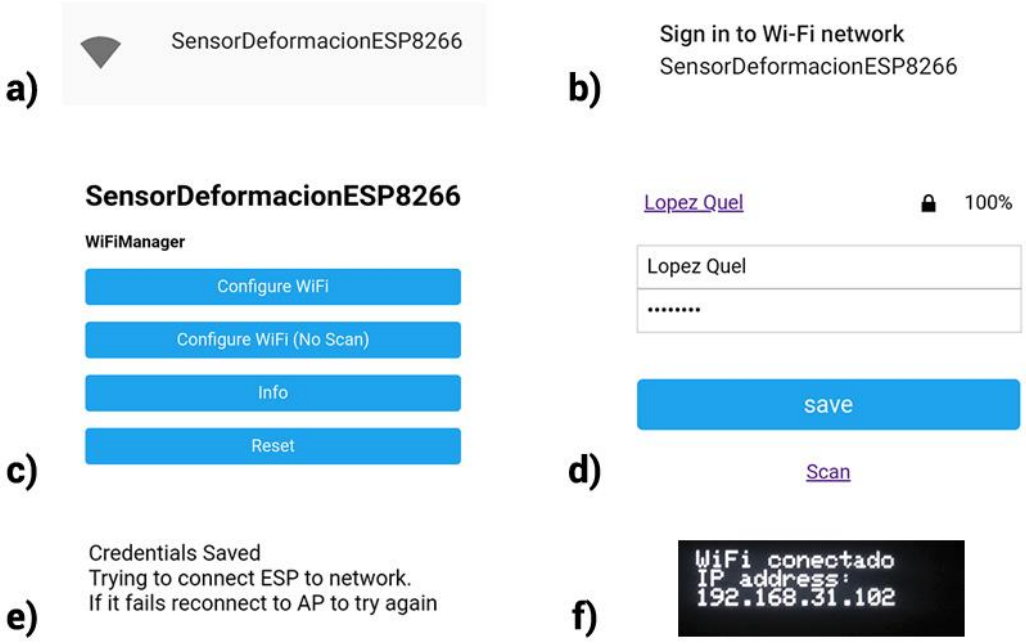
configurar los datos de la red a la cual se conectará el sensor de deformación. En caso contrario, el dispositivo empleará los datos de la última conexión realizada. Si estos datos no son encontrados o no es posible conectarse a la red el sensor repetirá el proceso de la habilitación del punto de acceso hasta que se conecte a una red estable.

5.3.1.1. Configuración para conectar el dispositivo a una red por medio de wifi

Para que el dispositivo interactúe con la base de datos y la aplicación web en la que se desplegaran estos, debe estar conectado a una red utilizando el módulo wifi. A continuación, se detalla el proceso que se debe seguir para poder conectar el dispositivo a la red utilizando el módulo wifi, el cual también se muestra en la figura 20.

- Luego de que se enciende el sensor, se busca el punto de acceso que este crea con el SSID SensorDeformacionESP8266, esto empleando otro dispositivo como un smartphone o una computadora.
- Al conectarse al punto de acceso el dispositivo redirige al usuario a la pantalla de configuración. En caso de no redirigir automáticamente a la pantalla de configuración se debe navegar manualmente, empleando un navegador web, a la IP por defecto 192.168.4.1.
- Desde la pantalla de configuración presionar el botón de *Configure wifi*, seleccionar la red a la que se desea conectar y, por último, en caso de utilizar autenticación, ingresar la contraseña.
- Al completarse este proceso de forma satisfactoria, la pantalla OLED del dispositivo mostrara el mensaje de wifi conectado y la IP asignada al dispositivo.

Figura 20. **Proceso para conectar el dispositivo a la red wifi**

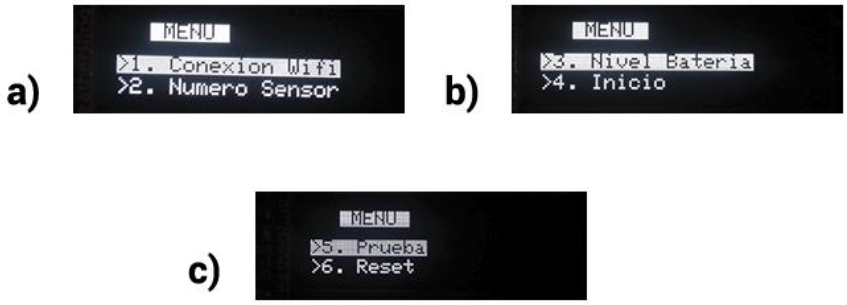


Fuente: elaboración propia, empleando Adobe Photoshop.

5.3.2. Menú y funciones del dispositivo

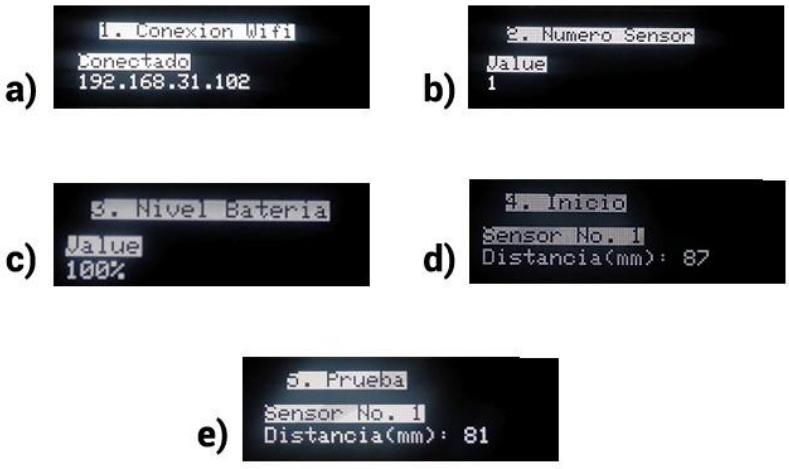
La pantalla OLED del dispositivo es la interfaz principal de comunicación con la que interactuará el usuario, ya que en ella se muestran los menús y mensajes que empleará el usuario para llevar a cabo los procedimientos que permitan el envió de datos a la base de datos y aplicación web para su análisis posterior.

Figura 21. **Visualización del menú principal del dispositivo**



Fuente: elaboración propia, empleando Adobe Photoshop.

Figura 22. **Visualización de las pantallas de menús utilizando el dispositivo**



Fuente: elaboración propia, empleando Adobe Photoshop.

En las figuras 21 y 22 se muestran las pantallas con los menús con los cuales interactuará el usuario cuando el dispositivo se está configurando para la recolección y envío de datos. A continuación, se detalla cada uno de los menús y su funcionamiento.

5.3.2.1. Menú principal

Esta pantalla despliega todas las opciones del sensor de deformación. A través de este se acceden a los siguientes menús.

- Conexión WiFi
- Configuración de número del sensor
- Nivel de batería
- Inicio de envío de datos
- Prueba del sensor de deformación
- Reset de los datos configurados

5.3.2.2. Menú de conexión wifi

En esta pantalla se despliega la información de la conexión a la red wifi del sensor de deformación. Los datos que muestra son los siguientes:

- Estado de la conexión: existen dos estados. Conectado cuando el dispositivo ha establecido una conexión satisfactoria con la red wifi. Y desconectado cuando se ha perdido la conexión.
- Dirección IP: muestra la dirección IP asignada al dispositivo. Esta se muestra solo cuando el dispositivo posee un estado de conectado.

5.3.2.3. Menú de configuración de número de sensor

La función principal de esta pantalla es asignar un número de identificación (id) al sensor para visualizar y clasificar los datos de manera correcta. Para seleccionar el número de sensor se utilizan los botones de navegación para incrementar o decrementar el número que identificara al sensor.

5.3.2.4. Menú de nivel de batería

En esta pantalla se muestra el nivel de batería del dispositivo, si se está empleando la batería como fuente de alimentación. Si el dispositivo está conectado a una fuente externa por medio del cable micro USB y la batería también se encuentre conectada, en esta pantalla se actualizará el nivel de carga.

5.3.2.5. Menú de inicio de envío de datos

Al acceder a esta pantalla el dispositivo iniciará con el proceso de envió de los datos, empleando el protocolo MQTT, también desplegara en la pantalla OLED los datos que serán enviados al módulo de procesamiento y almacenamiento.

5.3.2.6. Menú de prueba del sensor de deformación

Esta pantalla permite realizar pruebas con el sensor de deformación. Muestra en la pantalla OLED una simulación de toma de datos, desplegando las distancias obtenidas por el sensor sin ser enviadas al módulo de procesamiento y almacenamiento de datos.

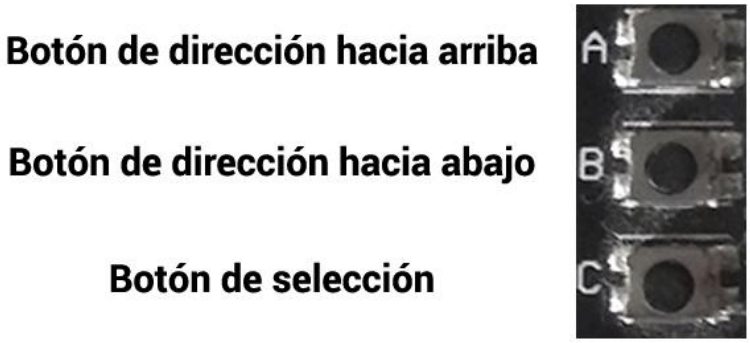
5.3.2.7. Menú de reset de los datos configurados

La función principal de esta pantalla es regresar a los valores definidos por defecto el sensor de deformación. Esta opción solo afecta el número de identificación del dispositivo. Si se requiere realizar una nueva conexión a la red WiFi se presiona el botón de *reset* del dispositivo y se sigue el proceso descrito.

5.3.3. Navegación y selección de menús del dispositivo

Para navegar a través de los menús y opciones, se emplean los botones de navegación del dispositivo. Son 3 botones que cumplen las funciones de dirección hacia arriba, dirección hacia abajo y selección respectivamente. En la figura 23 se muestra el diagrama e identificación de cada uno de ellos.

Figura 23. **Diagrama de los botones de navegación del sensor de deformación**

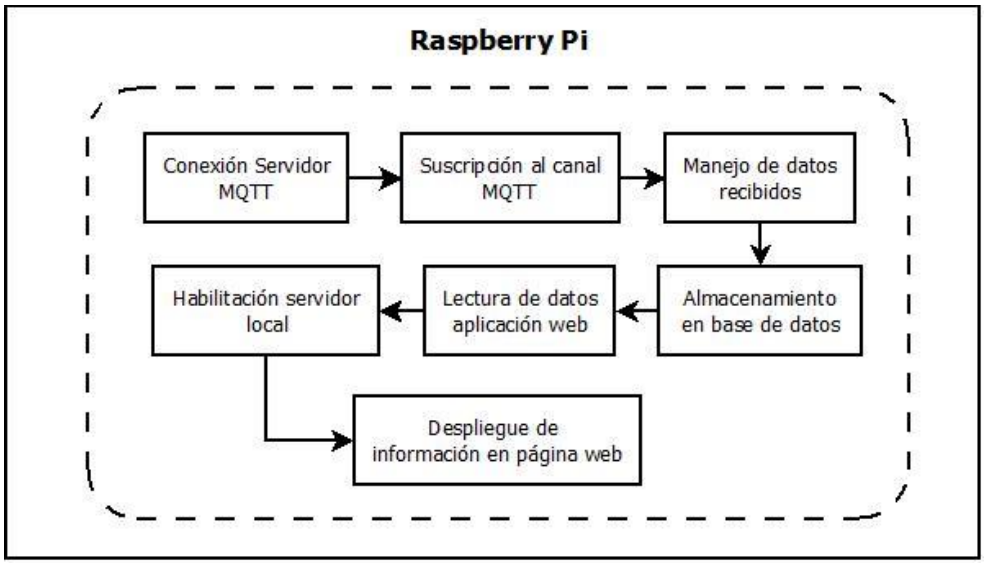


Fuente: elaboración propia, empleando Adobe Photoshop.

5.4. Funcionamiento del módulo de procesamiento de datos

La figura 24 describe el proceso que se ejecuta en el *script* realizado en Python, para interpretar los datos provenientes del sensor de deformación, a través del protocolo MQTT y generar la estructura necesaria que empleará la aplicación web para desplegar las distintas gráficas y datos. El programa utiliza una cola de datos, proporcionada por el protocolo MQTT, para procesar toda la información proveniente del sensor o los sensores que se utilicen en el sistema. Estos datos son almacenados en la base de datos y mostrados en tiempo real por la aplicación web.

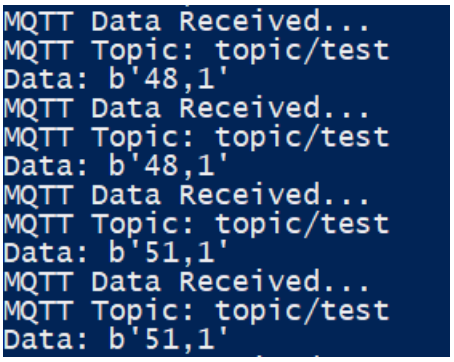
Figura 24. **Diagrama de procesamiento de datos**



Fuente: elaboración propia, empleando Dia.

En la figura 25 se observa la forma en que se reciben los datos empleando el protocolo MQTT, antes de ser procesados y almacenados en la base de datos.

Figura 25. **Recepción de datos en Raspberry pi por medio del protocolo MQTT**

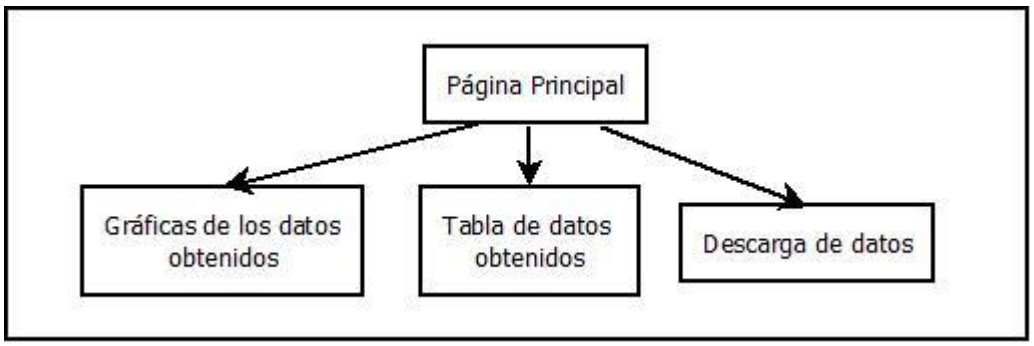


Fuente: elaboración propia.

5.4.1. Interfaz de visualización de la aplicación web

La figura 26 representa el mapa de la aplicación web. Esta se divide en 3 secciones: graficas de las distancias medidas por los sensores de deformación, tabla de datos obtenidos de los sensores de deformación y descarga de datos.

Figura 26. **Mapa de la aplicación web**



Fuente: elaboración propia, empleando Dia.

La aplicación web consiste en una sola página con las secciones antes descritas. La sección de graficas posee una lista desplegable en la que se puede seleccionar todos los sensores que se desean visualizar. La sección de la tabla de datos muestra los datos obtenidos los cuales se pueden filtrar por número de sensor, fecha y valores medidos. La sección de descarga consiste en un solo botón para obtener los datos en bruto de los sensores. A continuación, se describe a detalle la manera correcta de utilizar e interpretar la interfaz de la aplicación web diseñada.

5.4.1.1. Requisitos de la red LAN

El dispositivo Raspberry pi debe cumplir con los parámetros establecidos por la red en la que se encuentra, para ser visible tanto por los sensores como los usuarios que utilizaran la aplicación web, o bien, desean utilizar los datos directamente de la base para análisis más detallados. A continuación, se listan los requisitos.

- La dirección IP que se le asigne al dispositivo se debe encontrar en el rango correspondiente, generalmente este rango es proporcionado por un *router*.
- Debe contar con una máscara de subred y una puerta de enlace para enrutar el tráfico.
- En caso de que la *router* no cuente con un servidor DHCP o la red no cuente con un *router* para la asignación automática de direcciones, se debe configurar el dispositivo con los parámetros de los puntos anteriores de forma manual.

5.4.1.2. Página Inicial

Para acceder a la aplicación web se debe colocar la dirección IP en la barra de direcciones del explorador web. Al acceder desplegará las 3 secciones que se detallaron anteriormente. En la figura 27 se muestra el diseño de la página de inicio de la aplicación web.

Figura 27. **Página de inicio**

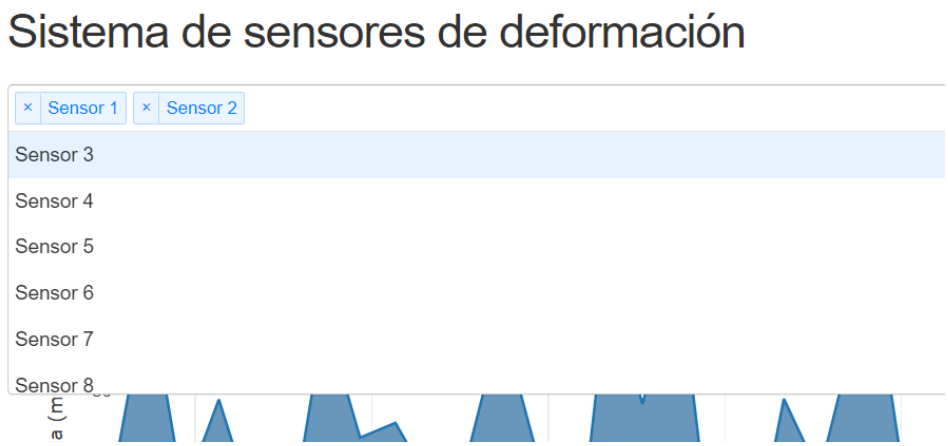
Fuente: elaboración propia, empleando Google Chrome.

5.4.1.2.1. Sección de gráficas

La primera sección de la página muestra las gráficas de los sensores. Por defecto, al iniciar, se muestra solo los datos del sensor 1. Esta sección consiste en las gráficas de los datos y en una lista desplegable que contiene los sensores disponibles del sistema, para filtrar las gráficas que serán visualizadas. Se

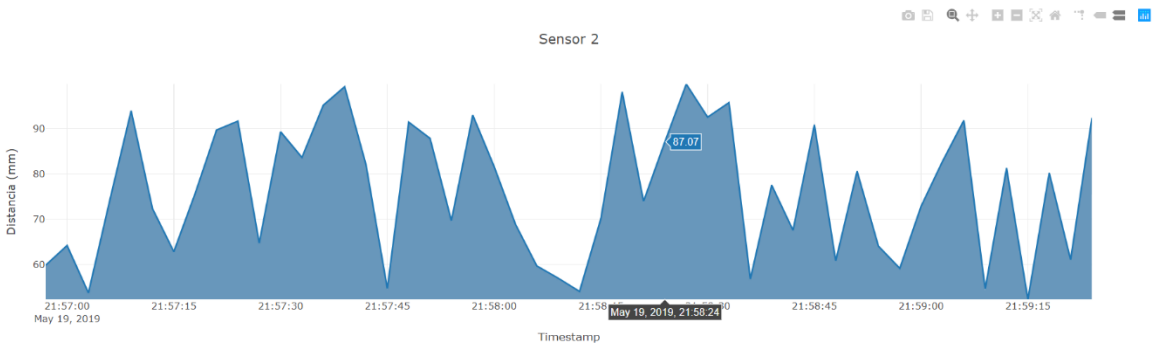
pueden seleccionar múltiples graficas de sensores o también se pueden ocultar para que solo se muestren las otras secciones.

Figura 28. **Lista desplegable para filtrar las gráficas de los sensores**



Fuente: elaboración propia, empleando Google Chrome.

Figura 29. **Visualización de las gráficas de los datos de los sensores**



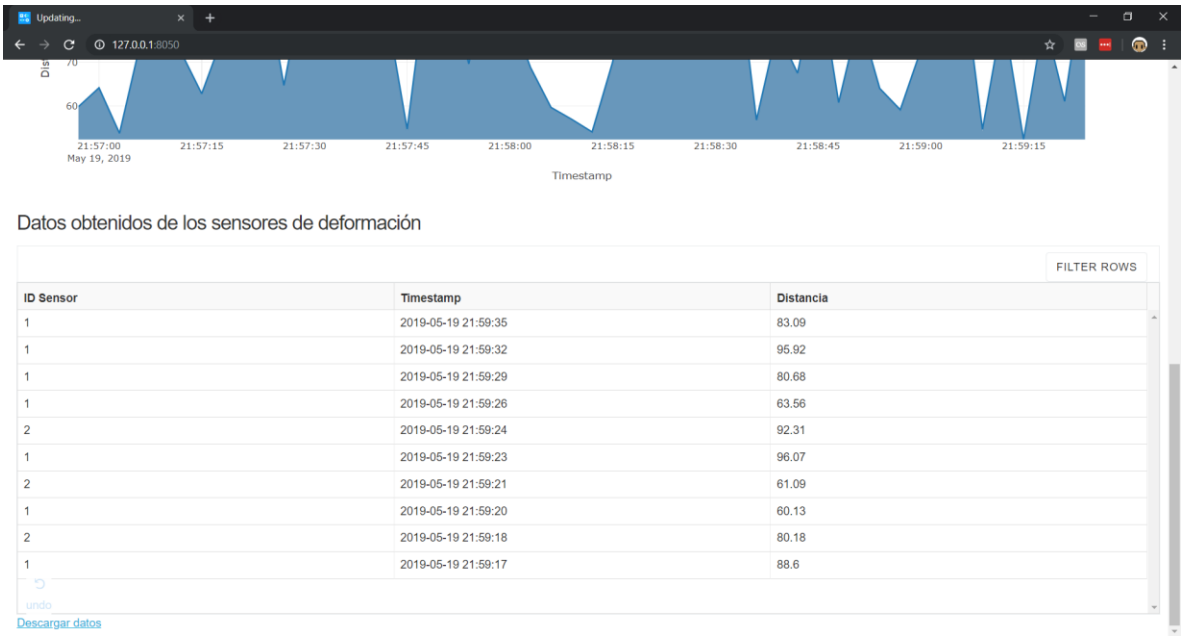
Fuente: elaboración propia, empleando Google Chrome.

5.4.1.2.2. Sección de tabla con los datos de los sensores

Esta sección de la aplicación web muestra los datos almacenados de los sensores activos. La tabla cuenta con un filtro con el cual se puede buscar datos específicos por número de sensor, fecha o distancia medida. Además, las columnas pueden ser ordenadas ya sea en forma ascendente o descendente. Esta tabla se actualiza al mismo tiempo que los datos se muestran en las gráficas de los sensores.

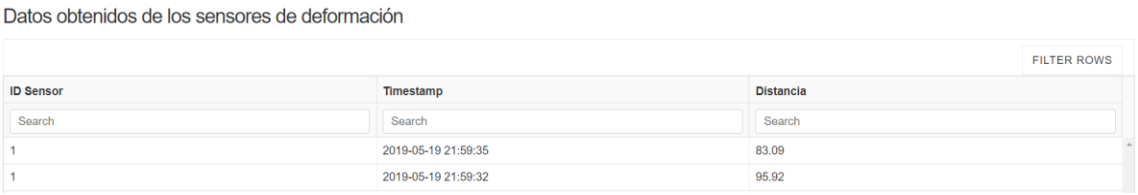
En la figura 30 se muestra la forma en que se visualiza la tabla con los datos en la aplicación web. En la figura 31 se muestran los filtros que se pueden aplicar a la tabla.

Figura 30. **Visualización de la tabla con los datos de los sensores**



Fuente: elaboración propia, empleando Google Chrome.

Figura 31. **Filtros aplicables a la tabla de datos de los sensores**

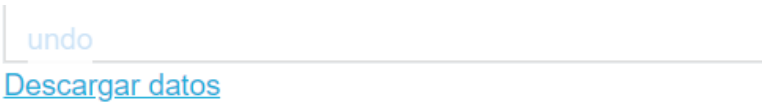


Fuente: elaboración propia, empleando Google Chrome.

5.4.1.2.3. Sección de descarga de datos

Esta sección consiste en un solo enlace, el cual descarga los datos en formato CSV para que el usuario los utilice para análisis posteriores, o bien, aplicaciones más complejas. En la figura 32 se muestra la forma en que se visualiza el enlace en la aplicación web.

Figura 32. **Visualización de enlace para descarga de datos de los sensores de deformación**



Fuente: elaboración propia, empleando Google Chrome.

CONCLUSIONES

1. El diseño presentado es capaz de realizar la función de medir las deformaciones superficiales y mostrar los datos en tiempo real a través de una aplicación web.

2. El dispositivo presenta una incerteza que oscila en el rango de 2 a 4 mm, dependiendo de la distancia a la que se encuentre, la cual cumple con la incerteza presentada por los deformímetros analógicos, los cuales presentan una incerteza en el rango de 2 a 3 mm.

3. El costo del diseño presentado es menor al de un deformímetro digital y cuenta con funciones de automatizar la toma de datos y presentarlos de una forma interactiva al usuario.

4. La automatización de toma de datos reduce el error de percepción en la lectura de las deformaciones que ocurre con los deformímetros tradicionales.

5. El uso de una batería de litio recargable y la conexión a la red por medio de WiFi brinda una mayor portabilidad para el uso del sensor de deformación en distintos escenarios.

6. La aplicación web se programó para visualizar los datos en tiempo real de todos los sensores que se conectan al sistema y para que los datos sean exportados de forma una sencilla y rápida para que sean utilizados en análisis posteriores.

RECOMENDACIONES

1. Utilizar el protocolo DHCP para la asignación dinámica de direcciones IP de los distintos dispositivos para que se cumpla con los lineamientos establecidos por el *router* que contiene el área local.

2. Utilizar un FQDN para el servidor que contiene la base de datos y la aplicación web para evitar utilizar direcciones IP estáticas y se pueda utilizar el dispositivo en distintas redes sin tener que cambiar este parámetro.

3. Agregar el FQDN del servidor que contiene la base de datos al servidor DNS para que resuelva la dirección IP que este tiene asignada.

4. El usuario debe de verificar el número de sensor asignado al dispositivo antes de iniciar el envío de datos hacia el servidor.

5. Realizar la prueba de medición desde el dispositivo antes de iniciar el envío de datos para comprobar el correcto funcionamiento del sensor de tiempo de vuelo.

6. Verificar que el dispositivo cuente con un nivel de batería adecuado para la duración de la prueba antes de iniciar el envió de datos.

7. Para obtener una incerteza menor se debe utilizar el modelo de sensor de distancia Micro-LIDAR de tiempo de vuelo VL6180X.

BIBLIOGRAFÍA

1. Adafruit Feather HUZZAH ESP826. [en línea]. <https://cdn-learn.adafruit.com/downloads/pdf/adafruit-feather-huzzah-esp8266.pdf >. [Consulta: 30 de marzo de 2018].

2. Arduino: Programación de Microcontroladores. [en línea]. <https://devcode.la/blog/arduino-programacion-microcontroladores/>. [Consulta: 10 de marzo de 2018].

3. CISCO. Direccionamiento de IP y conexión en subredes para los usuarios nuevos. [en línea]. [Consulta: 6 de marzo de 2018].

4. Dash Layout. [en línea]. <https://dash.plot.ly/getting-started?_ga=2.171526176.1421271689.1558920935-1702096862.1558920935>. [Consulta: 5 de junio de 2018].

5. ESTRADA, Nicolas; ASTUDILLO, Hernán. Comparing scalability of message queue system: ZeroMQ vs RabbitMQ. Universidad Técnica Federico Santa María Santiago, Chile. 2015. 6 p.

6. How to Run Your ESP8266 for Years on a Battery. [en línea]. <https://openhomeautomation.net/esp8266-battery>. [Consulta: 15 de junio de 2018].

7. IoT: Sobre protocolos M2M. [en línea]. <http://www.nearbysensor.com/iot-sobre-protocolos-m2m/>. [Consulta: 2 de marzo de 2018].

8. KILPELÄ, Ari. Pulsed time-of-flight laser range finder techniques for fast, high precision measurement applications. Universidad de Oulu: Oulu University Press, 2004. 96 p.

9. Li-Polymer 503562 1200mAh 3.7V with PCM. [en línea]. <https://cdn-shop.adafruit.com/product-files/258/C101-_Li-Polymer_503562_1200mAh_3.7V_with_PCM_APPROVED_8.18.pdf>. [Consulta: 30 de marzo de 2018].

10. OEL Display Module. [en línea]. <https://cdn-shop.adafruit.com/datasheets/UG-2832HSWEG02.pdf>. [Consulta: 30 de marzo de 2018].

11. ¿Qué es la comunicación M2M?. [en línea]. <https://www.tendencias21.net/telefonica/Que-es-la-comunicacion-M2M_a801.html>. [Consulta: 20 de febrero de 2018].

12. Raspberry Pi 3 Model B+. [en línea]. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>. [Consulta: 2 de junio de 2018].

13. ROSE, Karen; ELDRIDGE, Scott; CHAPIN Lyman. La internet de las cosas - Una breve reseña. Suiza: Internet Society (ISOC), 2015. 81 p.

14. Simplifying Time-of-Flight Distance Measurements. [en línea]. <https://www.digikey.com/en/articles/techzone/2017/jan/simplifying-time-of-flight-distance-measurements>. [Consulta: 18 de febrero de 2018].

15. Store MQTT Data from Sensors into SQL Database. [en línea]. <https://iotbytes.wordpress.com/store-mqtt-data-from-sensors-into-sql-database/>. [Consulta: 15 de marzo de 2018].

16. The Mosquitto MQTT Broker on Linux. [en línea]. <http://www.steves-internet-guide.com/install-mosquitto-linux/>. [Consulta: 10 de junio de 2018].

APÉNDICES

Apéndice 1. **Código fuente del microcontrolador ESP8266.**

```
#include <SPI.h>
#include <Wire.h>
#include <ESP8266WiFi.h>
#include <DNSServer.h>
#include <ESP8266WebServer.h>
#include <WiFiManager.h>
#include <WiFiUdp.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_VL53L0X.h>
#include <PubSubClient.h>

unsigned int localPort = 2390;
const char* mqtt_server = "mqtt.server.local";
char packetBuffer[255];
char  ReplyBuffer[] = "acknowledged";

Adafruit_SSD1306 display = Adafruit_SSD1306();
Adafruit_VL53L0X lox = Adafruit_VL53L0X();

WiFiUDP Udp;
WiFiClient mqttClient;
PubSubClient client(mqttClient);
//=====================================
long lastMsg = 0;
char msg[50];
int value = 0;
//=====================================
int current = 0;
```

Continuación apéndice 1.

```
int last = -1;
int menuitem = 1;
int frame = 1;
int page = 1;
int lastMenuItem = 1;
int keyButtons = 4;
int sensorMesuare = 0;
//========================================
//========================================
boolean backlight = true;
int selectedNumber = 0;
String number[10] = {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10"};
//========================================
//========================================
#define BUTTON_A 0
#define BUTTON_B 16
#define BUTTON_C 2
#define LED 0
#define btnUP     1
#define btnDOWN   2
#define btnSELECT 3
#define btnNONE   4
//========================================

String menuItem1 = "1. Conexion Wifi";
String menuItem2 = "2. Numero Sensor";
String menuItem3 = "3. Nivel Bateria";
String menuItem4 = "4. Inicio";
String menuItem5 = "5. Prueba";
String menuItem6 = "6. Reset";

boolean up = false;
boolean down = false;
```

Continuación apéndice 1.

```
boolean sel = false;

#if (SSD1306_LCDHEIGHT != 32)
#error("Height incorrect, please fix Adafruit_SSD1306.h!");
#endif


void setup()
{
  Serial.begin(115200);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.display();
  display.clearDisplay();
  pinMode(BUTTON_A, INPUT_PULLUP);
  pinMode(BUTTON_B, INPUT_PULLUP);
  pinMode(BUTTON_C, INPUT_PULLUP);
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  WiFiManager wifiManager;
  wifiManager.resetSettings();
  wifiManager.autoConnect("SensorDeformacionESP8266");
  if (!lox.begin())
  {
    Serial.println(F("Fallo al bootear VL53L0X"));
    while(1);
  }
  display.display();
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  display.println("WiFi conectado");
```

Continuación apéndice 1.

```
  display.println("IP address: ");
  display.println(WiFi.localIP());
  Udp.begin(localPort);
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  display.setCursor(0,0);
  display.display();
  delay(500);
  display.clearDisplay();
}

void loop()
{
  drawMenu();
  keyButtons = readButtons();
  Serial.println(up,down);
  switch (keyButtons)
  {
    case btnUP:
    {
      up = true;
      break;
    }
    case btnDOWN:
    {
      down = true;
      break;
    }
    case btnSELECT:
    {
      sel = true;
      break;
    }
```

Continuación apéndice 1.

```
  case btnNONE:
  {
    break;
  }
 }
 if (up && page == 1 )
 {
  up = false;
  if(menuitem==2 && frame ==2)
  {
    frame--;
  }
  if(menuitem==4 && frame ==4)
  {
    frame--;
  }
  if(menuitem==3 && frame ==3)
  {
    frame--;
  }
  lastMenuItem = menuitem;
  menuitem--;
  if (menuitem==0)
  {
    menuitem=1;
  }
 }
 else if (up && page == 2 && menuitem==1 )
 {
  up = false;
  displayWifiStatus(menuItem1);
 }
 else if (up && page == 2 && menuitem==2 )
```

Continuación apéndice 1.

```
{
  up = false;
  selectedNumber--;
  if(selectedNumber == -1)
  {
    selectedNumber = 9;
  }
}
else if (up && page == 2 && menuitem==3 )
{
  up = false;
}
else if (up && page == 2 && menuitem==4 )
{
  up = false;
}
if (down && page == 1)
{
  down = false;
  if(menuitem==3 && lastMenuItem == 2)
  {
    frame ++;
  }
  else  if(menuitem==4 && lastMenuItem == 3)
  {
    frame ++;
  }
  else  if(menuitem==5 && lastMenuItem == 4 && frame!=4)
  {
    frame ++;
  }
  lastMenuItem = menuitem;
  menuitem++;
```

Continuación apéndice 1.

```
  if (menuitem==7)
  {
    menuitem--;
  }

}
else if (down && page == 2 && menuitem==1)
{
  down = false;
  displayWifiStatus(menuItem1);
}
else if (down && page == 2 && menuitem==2)
{
  down = false;
  selectedNumber++;
  if(selectedNumber == 10)
  {
    selectedNumber = 0;
  }
}
else if (down && page == 2 && menuitem==3 )
{
  down = false;
}
else if (down && page == 2 && menuitem==4 )
{
  down = false;
}
if (sel)
{
  sel = false;
  if(page == 1 && menuitem ==6)
  {
```

Continuación apéndice 1.

```
      resetDefaults();
    }
    else if (page == 1 && menuitem<=5)
    {
      page=2;
    }
    else if (page == 2)
    {
      page=1;
    }
  }
  display.display();
  delay(1);
}

void drawMenu()
{
  if (page==1)
  {
    display.setTextSize(1);
    display.clearDisplay();
    display.setTextColor(BLACK, WHITE);
    display.setCursor(15, 0);
    display.print(" MENU ");
    display.drawFastHLine(0,10,83,BLACK);

    if(menuitem==1 && frame ==1)
    {
      displayMenuItem(menuItem1, 15,true);
      displayMenuItem(menuItem2, 25,false);
    }
    else if(menuitem == 2 && frame == 1)
    {
```

Continuación apéndice 1.

```
  displayMenuItem(menuItem1, 15,false);
  displayMenuItem(menuItem2, 25,true);
}
else if(menuitem == 3 && frame == 1)
{
  displayMenuItem(menuItem2, 15,false);
  displayMenuItem(menuItem3, 25,true);
}
else if(menuitem == 4 && frame == 2)
{
  displayMenuItem(menuItem3, 15,false);
  displayMenuItem(menuItem4, 25,true);
}
else if(menuitem == 3 && frame == 2)
{
  displayMenuItem(menuItem3, 15,true);
  displayMenuItem(menuItem4, 25,false);
}
else if(menuitem == 2 && frame == 2)
{
  displayMenuItem(menuItem2, 15,true);
  displayMenuItem(menuItem3, 25,false);
}
else if(menuitem == 5 && frame == 3)
{
  displayMenuItem(menuItem4, 15,false);
  displayMenuItem(menuItem5, 25,true);
}
else if(menuitem == 6 && frame == 4)
{
  displayMenuItem(menuItem5, 15,false);
  displayMenuItem(menuItem6, 25,true);
}
```

Continuación apéndice 1.

```
  else if(menuitem == 5 && frame == 4)
  {
    displayMenuItem(menuItem5, 15,true);
    displayMenuItem(menuItem6, 25,false);
  }
  else if(menuitem == 4 && frame == 4)
  {
    displayMenuItem(menuItem4, 15,true);
    displayMenuItem(menuItem5, 25,false);
  }
  else if(menuitem == 3 && frame == 3)
  {
    displayMenuItem(menuItem3, 15,true);
    displayMenuItem(menuItem4, 25,false);
  }
  else if(menuitem == 2 && frame == 2)
  {
    displayMenuItem(menuItem2, 15,true);
    displayMenuItem(menuItem3, 25,false);
  }
  else if(menuitem == 4 && frame == 3)
  {
    displayMenuItem(menuItem4, 15,true);
    displayMenuItem(menuItem5, 25,false);
  }
  display.display();
}
else if (page==2 && menuitem == 1)
{
  displayWifiStatus(menuItem1);
}
else if (page==2 && menuitem == 2)
{
```

Continuación apéndice 1.

```
    displayStringMenuPage(menuItem2, number[selectedNumber]);
  }
   else if (page==2 && menuitem == 3)
  {
    displayStringMenuPage(menuItem3, batteryLevel);
  }
  else if (page==2 && menuitem == 4)
  {
    displaySendData(menuItem4, number[selectedNumber]);
  }
  else if (page==2 && menuitem == 5)
  {
    displayTestData(menuItem5, number[selectedNumber]);
  }
  else if (page==2 && menuitem == 6)
  {
    displayStringMenuPage(menuItem4, number[selectedNumber]);
  }
}

void displayMenuItem(String item, int position, boolean selected)
{
    if(selected)
    {
      display.setTextColor(BLACK, WHITE);
    }
    else
    {
      display.setTextColor(WHITE, BLACK);
    }
    display.setCursor(0, position);
    display.print(">"+item);
}
```

Continuación apéndice 1.

```
void displayIntMenuPage(String menuItem, int value)
{
   display.setTextSize(1);
   display.clearDisplay();
   display.setTextColor(BLACK, WHITE);
   display.setCursor(15, 0);
   display.print(menuItem);
   display.drawFastHLine(0,10,83,BLACK);
   display.setCursor(5, 15);
   display.print("Value");
   display.setTextColor(WHITE, BLACK);
   display.setTextSize(1);
   display.setCursor(5, 25);
   display.print(value);
   display.setTextSize(1);
   display.display();
}

void displayStringMenuPage(String menuItem, String value)
{
   display.setTextSize(1);
   display.clearDisplay();
   display.setTextColor(BLACK, WHITE);
   display.setCursor(15, 0);
   display.print(menuItem);
   display.drawFastHLine(0,10,83,BLACK);
   display.setCursor(5, 15);
   display.print("Value");
   display.setTextColor(WHITE, BLACK);
   display.setTextSize(1);
   display.setCursor(5, 25);
   display.print(value);
   display.setTextSize(1);
```

Continuación apéndice 1.

```
    display.display();
}

void displayWifiStatus(String menuItem)
{
  if (WiFi.status() != WL_CONNECTED)
  {
    display.setTextSize(1);
    display.clearDisplay();
    display.setTextColor(BLACK, WHITE);
    display.setCursor(15, 0);
    display.print(menuItem);
    display.drawFastHLine(0,10,83,BLACK);
    display.setCursor(5, 15);
    display.print("Sin conexion WIFI");
    display.setTextColor(WHITE, BLACK);
    display.setTextSize(1);
    display.setCursor(5, 25);
    display.print("Revise su configuración");
    display.setTextSize(1);
    display.display();
  }
  else if (WiFi.status() == WL_CONNECTED)
  {
    display.setTextSize(1);
    display.clearDisplay();
    display.setTextColor(BLACK, WHITE);
    display.setCursor(15, 0);
    display.print(menuItem);
    display.drawFastHLine(0,10,83,BLACK);
    display.setCursor(5, 15);
    display.print("Conectado");
    display.setTextColor(WHITE, BLACK);
```

Continuación apéndice 1.

```
    display.setTextSize(1);
    display.setCursor(5, 25);
    display.print(WiFi.localIP());
    display.setTextSize(1);
    display.display();
  }
}

void displaySendData(String menuItem, String value)
{
  if (!client.connected())
  {
    reconnect();
  }
  long now = millis();
  if (now - lastMsg > 2000)
  {
    sensorMesuare = ReadSensor();
    lastMsg = now;
    snprintf (msg, 75, "%ld,%ld", sensorMesuare, value.toInt());
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish("topic/test", msg);
  }
  client.loop();
  display.setTextSize(1);
  display.clearDisplay();
  display.setTextColor(BLACK, WHITE);
  display.setCursor(15, 0);
  display.print(menuItem);
  display.drawFastHLine(0,10,83,BLACK);
  display.setCursor(5, 15);
  display.print("Sensor No. " + value);
```

Continuación apéndice 1.

```
  display.setTextColor(WHITE, BLACK);
  display.setTextSize(1);
  display.setCursor(5, 25);
  display.print("Distancia: ");
  if (sensorMesuare < 0)
  {
    display.print("Fuera Rango");
  }
  else
  {
    display.print(sensorMesuare);
  }
  display.setTextSize(1);
  display.display();
}

int readButtons()
{
  if (!digitalRead(BUTTON_A)) return btnUP;
  if (!digitalRead(BUTTON_B)) return btnDOWN;
  if (!digitalRead(BUTTON_C)) return btnSELECT;
  return btnNONE;
}

void callback(char* topic, byte* payload, unsigned int length)
{
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++)
  {
    Serial.print((char)payload[i]);
  }
```

Continuación apéndice 1.

```
}

void reconnect()
{
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("Sensor")) {
      Serial.println("connected");
      client.publish("outTopic", "Inicio");
      client.subscribe("SensorRx");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

int ReadSensor()
{
  VL53L0X_RangingMeasurementData_t measure;
  Serial.println("Leyendo el sensor... ");
  lox.rangingTest(&measure, false);
  if (measure.RangeStatus != 4)
  {
    current = measure.RangeMilliMeter;
  }
  else
  {
    Serial.println(" fuera del rango ");
    current = -1;
  }
```

Continuación apéndice 1.

```
  Serial.println(current);
  last = current;
  return current;
}

void displayTestData(String menuItem, String value)
{
  long now = millis();
  if (now - lastMsg > 500)
  {
    sensorMesuare = ReadSensor();
    lastMsg = now;
  }
  client.loop();
  display.setTextSize(1);
  display.clearDisplay();
  display.setTextColor(BLACK, WHITE);
  display.setCursor(15, 0);
  display.print(menuItem);
  display.drawFastHLine(0,10,83,BLACK);
  display.setCursor(5, 15);
  display.print("Sensor No. " + value);
  display.setTextColor(WHITE, BLACK);
  display.setTextSize(1);
  display.setCursor(5, 25);
  display.print("Distancia: ");
  if (sensorMesuare < 0)
  {
    display.print("Fuera Rango");
  }
  else
  {
    display.print(sensorMesuare);
```

Continuación apéndice 1.

```
  }
  display.setTextSize(1);
  display.display();
}
```

Fuente: elaboración propia.

Apéndice 2. **Código del script utilizado para almacenar los datos obtenidos a través del protocolo MQTT en la base datos.**

```
import paho.mqtt.client as mqtt
from store_Sensor_Data_to_DB import Sensor_Data_Handler

MQTT_Broker = "mqtt.server.local"
MQTT_Port = 1883
Keep_Alive_Interval = 45
MQTT_Topic_Test = "topic/test"

def on_connect(mosq, obj, rc):
    if rc != 0:
        pass
        print("Unable to connect to MQTT Broker...")
    else:
        print("Connected with MQTT Broker: " + str(MQTT_Broker))
        mqttc.subscribe(MQTT_Topic_Test, 0)


def on_message(mosq, obj, msg):
    print("MQTT Data Received...")
    print("MQTT Topic: " + msg.topic)
    print("Data: " + str(msg.payload))
    Sensor_Data_Handler(msg.topic, msg.payload)
```

Continuación apéndice 2.

```
def on_subscribe(mosq, obj, mid, granted_qos):
    pass

mqttc = mqtt.Client()

mqttc.on_connect = on_connect
mqttc.on_message = on_message
mqttc.on_subscribe = on_subscribe

mqttc.connect(MQTT_Broker, int(MQTT_Port), int(Keep_Alive_Interval))
mqttc.subscribe(MQTT_Topic_Test, 0)

mqttc.loop_forever()


import json
import psycopg2
import creds_Config as creds
import datetime


conn_string = 'host='+ creds.PGHOST +' port='+ creds.PGPORT \
            +' dbname='+ creds.PGDATABASE +' user=' + creds.PGUSER \
            + ' password='+ creds.PGPASSWORD


class DatabaseManager():
    def __init__(self):
        self.conn = psycopg2.connect(conn_string)
        self.cur = self.conn.cursor()

    def add_del_update_db_record(self, sql_query, args=()):
        print(sql_query, args)
```

Continuación apéndice 2.

```
            self.cur.execute(sql_query, args)
            self.conn.commit()
            return

        def __del__(self):
            self.cur.close()
            self.conn.close()


def Sensor_Data_Handler(topic , jsonData):
    json_Dict = json.loads(jsonData)
    SensorID = json_Dict['Sensor_ID']
    Date_and_Time = json_Dict['Date']
    Distance = json_Dict['Distance']

    print(SensorID, Distance, Date_and_Time)
    dbObj = DatabaseManager()
    dbObj.add_del_update_db_record("insert into sensor_distance_data (SensorID,
date_n_time, distance) values ('{}','{}',{})".format(SensorID, '{0:%Y-%m-%d
%H:%M:%S}'.format(datetime.datetime.now()), float(Distance)))
    del dbObj
    print("Inserted Distance Data into Database.")
    print("")
```

Fuente: elaboración propia.

Apéndice 3. **Código de la aplicación web.**

```
import dash
import dash_core_components as dcc
import dash_html_components as html
import dash_table_experiments as dtable
from pandas_datareader.data import DataReader
import time
from collections import deque
import plotly.graph_objs as go
import psycopg2
import pandas as pd
import creds_Config as creds
import random

conn_string = 'host=' + creds.PGHOST + ' port=' + creds.PGPORT \
        + ' dbname=' + creds.PGDATABASE + ' user=' + creds.PGUSER \
        + ' password=' + creds.PGPASSWORD

external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']

app = dash.Dash('sensor-data', external_stylesheets=external_stylesheets)

max_length = 50
sensor1 = deque(maxlen=max_length)
sensor2 = deque(maxlen=max_length)
sensor3 = deque(maxlen=max_length)
sensor4 = deque(maxlen=max_length)
sensor5 = deque(maxlen=max_length)
sensor6 = deque(maxlen=max_length)
sensor7 = deque(maxlen=max_length)
sensor8 = deque(maxlen=max_length)
sensor9 = deque(maxlen=max_length)
sensor10 = deque(maxlen=max_length)
time1 = deque(maxlen=max_length)
```

Continuación apéndice 3.

```
time2 = deque(maxlen=max_length)
time3 = deque(maxlen=max_length)
time4 = deque(maxlen=max_length)
time5 = deque(maxlen=max_length)
time6 = deque(maxlen=max_length)
time7 = deque(maxlen=max_length)
time8 = deque(maxlen=max_length)
time9 = deque(maxlen=max_length)
time10 = deque(maxlen=max_length)

data_dict = {
    "Sensor 1": [sensor1, time1],
    "Sensor 2": [sensor2, time2],
    "Sensor 3": [sensor3, time3],
    "Sensor 4": [sensor4, time4],
    "Sensor 5": [sensor5, time5],
    "Sensor 6": [sensor6, time6],
    "Sensor 7": [sensor7, time7],
    "Sensor 8": [sensor8, time8],
    "Sensor 9": [sensor9, time9],
    "Sensor 10": [sensor10, time10]
}


def update_obd_values(sensor1, sensor2, sensor3, sensor4, sensor5, sensor6, sensor7, sensor8, sensor9, sensor10, time1, time2, time3, time4, time5, time6, time7, time8, time9, time10):
    connection = psycopg2.connect(conn_string)
    cursor = connection.cursor()
    sensor_query = "Select * from public.sensor_distance_data where sensorid='{}' ORDER BY date_n_time DESC limit 1"
    sensors = [[sensor1, time1], [sensor2, time2], [sensor3, time3], [sensor4, time4], [sensor5, time5], [
```

Continuación apéndice 3.

```
            sensor6, time6], [sensor7, time7], [sensor8, time8], [sensor9, time9], [sensor10,
time10]]
        for i in range(len(sensors)):
            cursor.execute(sensor_query.format(i+1))
            sensor_records = cursor.fetchone()
            if sensor_records != None:
                if len(sensors[i][1]) > 0:
                    if sensors[i][1][-1] != sensor_records[2]:
                        sensors[i][0].append(sensor_records[1])
                        sensors[i][1].append(sensor_records[2])
                else:
                    sensors[i][0].append(sensor_records[1])
                    sensors[i][1].append(sensor_records[2])

        return sensor1, sensor2, sensor3, sensor4, sensor5, sensor6, sensor7, sensor8,
sensor9, sensor10, time1, time2, time3, time4, time5, time6, time7, time8, time9, time10


    sensor1, sensor2, sensor3, sensor4, sensor5, sensor6, sensor7, sensor8, sensor9,
sensor10, time1, time2, time3, time4, time5, time6, time7, time8, time9, time10 =
update_obd_values(
        sensor1, sensor2, sensor3, sensor4, sensor5, sensor6, sensor7, sensor8, sensor9,
sensor10, time1, time2, time3, time4, time5, time6, time7, time8, time9, time10)


    app.layout = html.Div([
        html.Div([
            html.H2('Sistema de sensores de deformación',
                    style={'float': 'left',
                           }),
        ]),
        dcc.Dropdown(id='sensor-data-name',
                     options=[{'label': s, 'value': s}
                              for s in data_dict.keys()],
```

Continuación apéndice 3.

```
                value=['Sensor 1'],
                multi=True
                ),
        html.Div(children=html.Div(id='graphs'), className='row'),
        dcc.Interval(
            id='graph-update',
            interval=1000),
        html.H4(children='Datos obtenidos de los sensores de deformación'),
        dtable.DataTable(id='sensor-data-table',
                         rows=[{}],
                         row_selectable=False,
                         filterable=True,
                         sortable=False,
                         editable=False),
        html.A(
            'Descargar datos',
            id='download-link',
            download="rawdata.csv",
            href="",
            target="_blank"
        )
    ], className="container", style={'width': '98%', 'margin-left': 10, 'margin-right': 10, 'max-
width': 50000})


    @app.callback(
        dash.dependencies.Output('graphs', 'children'),
        [dash.dependencies.Input('sensor-data-name', 'value')],
        events=[dash.dependencies.Event('graph-update', 'interval')]
    )
    def update_graph(data_names):
        graphs = []
```

Continuación apéndice 3.

```
        update_obd_values(sensor1, sensor2, sensor3, sensor4, sensor5, sensor6, sensor7, sensor8,
                          sensor9, sensor10, time1, time2, time3, time4, time5, time6, time7, time8, time9, time10)
        if len(data_names) > 2:
            class_choice = 'col s12 m6 l4'
        elif len(data_names) == 2:
            class_choice = 'col s12 m6 l6'
        else:
            class_choice = 'col s12'

        for data_name in data_names:

            data = go.Scatter(
                x=list(data_dict[data_name][0]),
                y=list(data_dict[data_name][1]),
                name='Scatter',
                fill="tozeroy",
                fillcolor="#6897bb"
            )

            graphs.append(html.Div(dcc.Graph(
                id=data_name,
                animate=False,
                figure={'data': [data], 'layout': {'xaxis': {'title': 'Timestamp', 'range': [min(data_dict[data_name][0]), max(data_dict[data_name][0])]},
                                                   'yaxis': {'title': 'Distancia (mm)', 'range': [min(data_dict[data_name][1]), max(data_dict[data_name][1])]},
                                                   'title': '{}'.format(data_name)
                                                   }}
            )))

        return graphs
```

Continuación apéndice 3.

```
@app.callback(
    dash.dependencies.Output('sensor-data-table', 'rows'),
    [dash.dependencies.Input('sensor-data-name', 'value')],
    events=[dash.dependencies.Event('graph-update', 'interval')]
)
def generate_table(max_rows=10):
    connection = psycopg2.connect(conn_string)
    dataframe = pd.read_sql_query(
        'Select sensorid as "ID Sensor", date_n_time as "Timestamp", distance as "Distancia"
from public.sensor_distance_data ORDER BY date_n_time DESC limit 10', con=connection)
    return dataframe.to_dict('records')


if __name__ == '__main__':
    app.run_server(debug=True)
```

Fuente: elaboración propia.