UNIVERSIDAD TECNOLÓGICA DEL CENTRO DE VERACRUZ

REPORTE FINAL DE ESTADÍA

Osdair Jaimes Nava

Diseño de APIs para la lectura de variables empleando dispositivos IoT

EDUCACIÓN SECRETARÍA DE EDUCACIÓN PÚBLICA
UTP COORDINACIÓN GENERAL DE UNIVERSIDADES TECNOLÓGICAS Y POLITÉCNICAS
VERACRUZ GOBIERNO DEL ESTADO
SEV Secretaría de Educación
DET Dirección de Educación Tecnológica del Estado de Veracruz
SEMSyS Subsecretaría de Educación Media Superior y Superior

UNIVERSIDAD TECNOLÓGICA DEL CENTRO DE VERACRUZ

INGENIERIA EN REDES INTELIGENTES Y CIBERSEGURIDAD

Diseño de APIs para la lectura de variables empleando dispositivos de IoT

REPORTE FINAL DE ESTADÍA

QUE PARA OBTENER EL GRADO ACADÉMICO DE

INGENIERO EN REDES INTELIGENTES Y CIBERSEGURIDAD

OSDAIR JAIMES NAVA

ASESOR ACADÉMICO: DR. LUIS ROLANDO GUARNEROS NOLASCO

ASESOR INDUSTRIAL: ING. HUGO RODRÍGUEZ ASTORGA

CUITLÁHUAC, VER. ABRIL, 2023

UNIVERSIDAD TECNOLÓGICA DEL CENTRO DE VERACRUZ

CONTENIDO

Capítulo I: Introducción 7

1.1 Estado del Arte 8

1.1.1 ¿Qué es el internet de las cosas? 8

1.1.2 Estrategias de IoT para Lograr Ciudades Digitales Seguras, más Inclusivas y Sustentables 9

1.1.3 ¿Qué es un SOC? 10

1.1.4 Diseño e implementación de una solución basada en IoT para la empresa Galeo Enrollables 11

1.2 Planteamiento del Problema 12

1.3 Objetivos 13

1.3.1 Objetivo General 13

1.3.2 Objetivos Específicos 13

1.4 Hipótesis 13

1.5 Justificación del Proyecto 14

1.6 Alcances y Limitaciones 15

1.6.1 Alcances 15

1.6.2 Limitaciones 15

1.6.3 Plan de contingencia ante las limitaciones 15

1.7 La empresa 16

1.7.1 Misión 16

1.7.2 Visión 16

1.7.3 Valores 16

Capitulo II: Metodología 17

2.1 Metodología SCRUM 17

2.2 Fases del ciclo de vida de SCRUM 17

- 2.2.1 Inicio 17
- 2.2.2 Planificación y estimación 17
- 2.2.3 Implementación 18
- 2.2.4 Revisión y retrospectiva 18
- 2.2.5 Lanzamiento 18

Capítulo III: Desarrollo del Proyecto 19
3.1 Aplicación de la metodología 19
- 3.1.1 Inicio 19
- 3.1.2 Planificación y estimación 20
- 3.1.3 Implementación 24

3.1.3.1 Selección del software para la programación 24
3.1.3.2 Selección del hardware 26
3.1.3.3 Pruebas de comunicación serial para muestreo de información en un monitor serial 37
3.1.3.4 Pruebas de control para servomotores a través de un monitor serial 42
3.1.3.5 Desarrollo del código para la lectura de los sensores de IoT 44
3.1.3.6 Pruebas de comunicación del protocolo I2C para la comunicación del sensor de luminosidad y la pantalla LCD 48
3.1.3.7 Creación y configuración de un punto de acceso para la configuración de los dispositivos IoT 51
3.1.3.8 Implementación y configuración de un servidor web para pruebas del muestreo de información 53
3.1.3.9 Diseño y programación de las APIs de lectura 55
3.1.3.10 Pruebas de la integración de las APIs con los sensores seleccionados 70

- 3.1.4 Revisión y retrospectiva 74
- 3.1.5 Lanzamiento 75

Capítulo IV: Resultados y conclusiones 76
4.1 Resultados 76
4.2 Trabajos Futuros 76
4.3 Recomendaciones 77
Bibliografía 78

Tabla de ilustraciones:

Ilustración 1: Simbolismo del internet de las cosas. 8
Ilustración 2: Ilustración de una ciudad empleando totalmente IoT. 9
Ilustración 3: Ejemplo de referencia de la composición de un SoC. 10
Ilustración 4: Diversos tipos de Arduino empleados. 11
Ilustración 5: Entorno del software IDE de Arduino. 24
Ilustración 6: Pines del NodeMCU. 27
Ilustración 7: Esquema de la Tarjeta Integradora Mecatrónica. 28
Ilustración 8: LEDs 28
Ilustración 9: Valores de las resistencias. 29
Ilustración 10: Resistencias de diversos valores. 30
Ilustración 11: Capacitores de diversos voltajes. 30
Ilustración 12: Transistor. 31
Ilustración 13: Diodos. 31
Ilustración 14: Potenciómetro 32
Ilustración 15: Botón de pulsación. 32
Ilustración 16: Relevador. 33
Ilustración 17: Composición del relevador. 33
Ilustración 18: Servomotor. 34
Ilustración 19: Microcontrolador SoC. 35
Ilustración 20: Componentes del monitor serial. 38
Ilustración 21: Código para pruebas de comunicación. 40
Ilustración 22: Resultado de las pruebas de comunicación. 41
Ilustración 23: Resultado de las pruebas de comunicación, parte 2. 42
Ilustración 24: Código empleado para pruebas de servomotores. 43
Ilustración 25: Sensores de temperatura y humedad DHT11 y DHT22. 44
Ilustración 26: Partes del sensor DHT11. 45
Ilustración 27: Gestor de librerías para el sensor DHT11. 45
Ilustración 28: Código para el funcionamiento del sensor DHT22. 47
Ilustración 29: Captura de las variables del sensor DHT22. 48
Ilustración 30: Código para pruebas del sensor de luminosidad. 49
Ilustración 31: Código para prueba de la pantalla LCD. 50
Ilustración 32: Captura de variables del sensor de luminosidad. 51
Ilustración 33: Código para la creación de un punto de acceso. 52
Ilustración 34: Conexión al punto de acceso creado. 53
Ilustración 35: Verificación la dirección IP asignada en por el punto de acceso. 53
Ilustración 36: Código para la creación de un servidor y página web. 54
Ilustración 37: Conexión con la página web creada. 55
Ilustración 38: Desarrollo del código de la primera API de lectura. 56
Ilustración 39: Desarrollo del código de la primera API de lectura, parte 2. 57
Ilustración 40: Desarrollo del código de la segunda API de lectura. 58

Ilustración 41: Desarrollo del código de la segunda API de lectura, parte 2. 59
Ilustración 42: Desarrollo del código de la segunda API de lectura, parte 3. 60
Ilustración 43: Desarrollo del código de la segunda API de lectura, parte 4. 61
Ilustración 44: Desarrollo del código de la segunda API de lectura, parte 5. 62
Ilustración 45: Desarrollo del código de la segunda API de lectura, parte 6. 63
Ilustración 46: Integración de ambos códigos. 64
Ilustración 47: Integración de ambos códigos, parte 2. 65
Ilustración 48: Integración de ambos códigos, parte 3. 66
Ilustración 49: Integración de ambos códigos, parte 4. 67
Ilustración 50: Integración de ambos códigos, parte 5. 68
Ilustración 51: Integración de ambos códigos, parte 6. 69
Ilustración 52: Página del menú de interacción con los dispositivos. 72
Ilustración 53: Encendido del LED desde la página web. 72
Ilustración 54: Apagado del LED desde la página web. 72
Ilustración 55: Funcionalidad del potenciometro desde la página web. 73
Ilustración 56: Funcionamiento del servomotor desde la página web. 73
Ilustración 57: Microcontrolador ESP-8266 dañado. 74

Índice de tablas:

Tabla 1: Primera historia del usuario. 20
Tabla 2: Segunda historia del usuario. 20
Tabla 3: Tercera historia del usuario. 21
Tabla 4: Cuarta historia del usuario. 21
Tabla 5: Quinta historia del usuario. 21
Tabla 6: Sexta historia del usuario. 22
Tabla 7: Séptima historia del usuario. 22
Tabla 8: Octava historia del usuario. 22
Tabla 9: Cronograma de actividades a realizar. 23
Tabla 10: Dispositivos empleados en el desarrollo del proyecto. 37
Tabla 11: Diferencias entre los sensores DHT11 y DHT22. 44

Capítulo I: Introducción

En los últimos veinte años, internet ha tenido un avance sumamente rápido, así mismo, ha llegado a impactar en diferentes ámbitos humanos, como sociales, académicos, laborales, entre otros. Esto es debido a que las personas nos solemos beneficiar de ello, ya que nos suele facilitar diversas tareas que cuentan con diferentes grados de complejidad. En las empresas más grandes, medianas o inclusive en las empresas locales de nuestras ciudades, ofrecen diferentes y variados servicios, que van desde la búsqueda de una determinada información, hasta el respaldo y guardado de una copia de seguridad, entre algunas otras funciones.

Al día de hoy, se ha aprovechado y sobreexplotado esta tecnología, pues gracias a ella, se han automatizado diversos procesos y se han creado proyectos que realicen tareas de manera mucho más sencilla gracias a la unión de protocolos de comunicación, sensores, dispositivos y controladores. Por todo lo anterior mencionado, la empresa Rodas, tiene el objetivo de implementar APIs para la lectura de variables empleando dispositivos IoT ya que de esa manera podrían realizar procesos mucho más rápidos y sencillos, teniendo un mejor control de la información adquirida.

El presente proyecto la metodología SCRUM, la cual se divide en 5 etapas; Inicio, Planificación y estimación, Implementación, Revisión y retrospectiva y Lanzamiento, ya que de esta manera se podrá realizar un informe mucho más detallado de las tareas a realizar para el desarrollo e implementación de las APIs.

Es por ello, que en este documento se lleva a cabo el registro de la realización de un proyecto, el cual será, implementar una solución a una problemática que enfrentan diversas empresas, en este caso Rodas, tomando en cuenta y justificando cada uno de esos aspectos que conformarán este proyecto, comprendiendo la importancia que desempeñan en este.

1.1 Estado del Arte

1.1.1 ¿Qué es el internet de las cosas?

[1] La Internet de las cosas (IoT) describe la red de objetos físicos ("cosas") que llevan incorporados sensores, software y otras tecnologías con el fin de conectarse e intercambiar datos con otros dispositivos y sistemas a través de Internet. Estos dispositivos van desde objetos domésticos comunes hasta herramientas industriales sofisticadas. IoT se ha convertido en una de las tecnologías más importantes del siglo XXI. Ahora que podemos conectar objetos cotidianos, electrodomésticos, coches, termostatos, monitores de bebés, a Internet a través de dispositivos integrados, es posible una comunicación fluida entre personas, procesos y cosas. Mediante la informática de bajo costo, la nube, big data, analítica y tecnologías móviles, las cosas físicas pueden compartir y recopilar datos con una mínima intervención humana. En este mundo hiperconectado, los sistemas digitales pueden grabar, supervisar y ajustar cada interacción entre las cosas conectadas. El mundo físico y el digital van de la mano y cooperan entre sí.

Ilustración 1: Simbolismo del internet de las cosas.

[1]

1.1.2 Estrategias de IoT para Lograr Ciudades Digitales Seguras, más Inclusivas y Sustentables

[2] El presente proyecto realizado en la Universidad Nacional de La Plata, se basa en la explicación del término "Smart cities" -ciudades inteligentes-, que es más que una frase de moda: es una de las tendencias más desafiantes vinculada a otro concepto actual "Internet de las cosas" o simplemente IoT. El término IoT se refiere generalmente a escenarios donde la capacidad de cómputo y la conectividad de las redes se extienden a objetos, sensores y elementos cotidianos - no computadoras personales-, permitiendo que estos dispositivos generen, intercambien y consuman datos. Así mismo, En todo el mundo, más y más ciudades se están convirtiendo en "inteligentes" mediante el uso de tecnología avanzada para mejorar sus servicios y la calidad de vida de sus ciudadanos, y esta revolución liderada por IoT está ayudando a llevar a cabo ciudades inteligentes para mejorar los servicios y la calidad de vida de los ciudadanos, de esta manera se podrá ayudar tanto a las grandes metrópolis, como a las pequeñas ciudades a transformarse en ciudades inteligentes y verdaderamente conectadas.

Ilustración 2: Ilustración de una ciudad empleando totalmente IoT.

[2, P. 800]

1.1.3 ¿Qué es un SOC?

[3] Un Sistema en chip o SoC integra casi todos los componentes de un chipset (características del chipset) en un solo chip de silicio. Junto con un procesador, el sistema integrado en chip suele contener una GPU (procesador de gráficos), memoria, controlador USB, circuitos de administración de energía y radios inalámbricas. Debido a que un sistema integrado en chip incluye tanto el hardware como el software, utiliza menos energía, tiene un mejor desempeño, requiere menos espacio y es más confiable que los sistemas multichip. Puede parecer que prácticamente todas las CPUs modernas son de este tipo, ya que al final muchas integran gráficos, tienen conexiones e interfaces directas, caches y sub procesadores criptográficos e infinidad de elementos más, pero estas no son SoC. La principal diferencia entre el SoC y procesadores o APUs que se pueden parecer es el propósito, el SoC está preparado para funcionar sin ningún otro chip de apoyo mientras que las CPUs normales, que aunque hagan muchas funciones, están pensadas para depender de un chipset externo y otros muchos chips y piezas para funcionar como deberían y para ampliar sus funcionalidades, no como un SoC que no necesita chips externos para funcionar correctamente, aunque si pueden venir acompañados de chips para ampliar su funcionalidad como por ejemplo el almacenamiento.

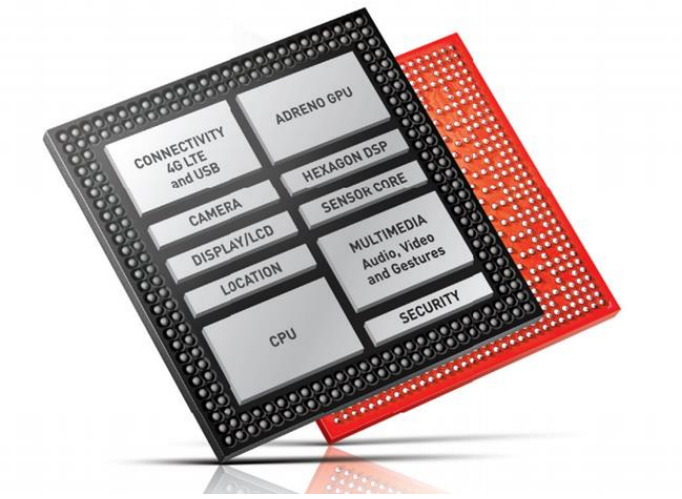


Ilustración 3: Ejemplo de referencia de la composición de un SoC.

[3]

1.1.4 Diseño e implementación de una solución basada en IoT para la empresa Galeo Enrollables

[4] El presente proyecto realizado en la Universidad Pública de Navarra surge por la necesidad de la empresa Galeo Enrollables (dedicada a la fabricación de cortina técnica) en conocer los tiempos de fabricación de sus productos y la situación en tiempo real de su proceso productivo. Con los datos obtenidos se analizarán las ineficiencias, se optimizarán los tiempos y costes de producción y, en consecuencia, aumentará la productividad. Además, la empresa tendrá la posibilidad de analizar y tomar decisiones en tiempo real. El proyecto se basa en la implantación de una red de Arduinos y sensores para la captación de datos del proceso de producción de una nave industrial. Esto se llevará a cabo en el contexto de la Industria 4.0 y el IoT (internet de las cosas). Los datos extraídos por los sensores se procesarán mediante Arduino, convirtiéndolos en información valiosa para la empresa. El trabajo incluirá el planteamiento, desarrollo y la implantación de la red. De igual forma se mostrará la comparativa de las diferentes opciones para la realización de la red. Para conseguir los objetivos se seleccionarán los sensores necesarios para cada puesto del proceso productivo y se programarán con microcontroladores Arduino. De esta manera, se recolectarán los datos obtenidos, para después mediante un nodo coordinador conectado a un ordenador donde se podrán visualizar gráficamente en tiempo real. Además, se determinará la configuración óptima de la red para este caso individual

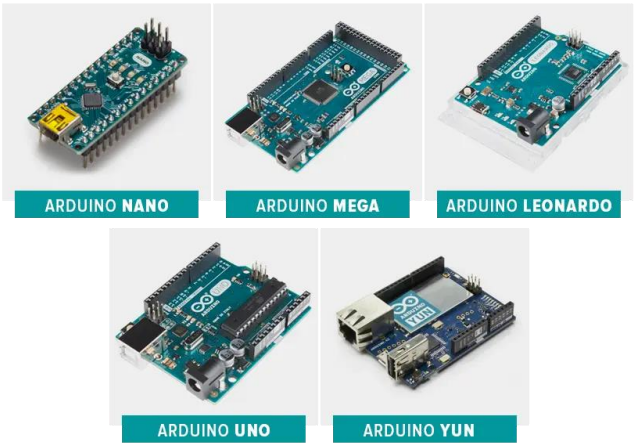


Ilustración 4: Diversos tipos de Arduino empleados.

[4, p. 13]

1.2 Planteamiento del Problema

Conforme pasan los años, el internet cada vez va teniendo más funciones y capacidades, inicialmente creado como un medio de comunicación y actualmente empleado para diversas cuestiones debido a sus amplias áreas de oportunidad y de usos en diferentes situaciones, como la oferta de servicios a través de este medio, control de dispositivos, almacenamiento, entre otras cuestiones. Es por ello que se han visto en la necesidad de crear e implementar nuevos protocolos y servicios, tanto de comunicación como de funcionamiento que perfeccionen o aseguren un adecuado desempeño de este, así mismo, comenzaron a emplear este medio para manejar y monitorizar cuestiones tales como temperatura, humedad, sensores, luces, entre otros datos, gracias al Internet de las cosas y los dispositivos que se encuentran dentro de diversas empresas, instituciones, departamentos gubernamentales, etcétera.

Por lo anterior mencionado, la empresa RODAS Computación S.A. DE C.V. que se encuentra ubicada en la ciudad de Orizaba, Veracruz, requiere de API's que sean capaces de realizar lecturas a las variables que estarán capturando los diversos dispositivos de IoT, empleando un microcontrolador tipo SoC, un servidor web y peticiones HTTP, para de esta manera lograr controlar y adquirir variables por medio de una página web.

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar e implementar API's para controlar y adquirir variables por medio de una página web empleando dispositivos de IoT para el envío y la comunicación de datos.

1.3.2 Objetivos Específicos

- Seleccionar la metodología que será empleada en el desarrollo del proyecto.
- Elegir el software que será aplicado para la programación de las API’s.
- Seleccionar los sensores de IoT a implementar en API’s.
- Realizar pruebas de comunicación serial a través de un monitor serial.
- Desarrollar el código para el funcionamiento de la lectura de los sensores de IoT.
- Realizar pruebas de comunicación del protocolo I2C para la configuración del sensor de luminosidad y la pantalla LCD.
- Implementación de un servidor web para pruebas del muestreo de información.
- Diseñar y programar las API’s que darán lectura a los sensores empleados.
- Controlar y adquirir variables por medio de una página web.

1.4 Hipótesis

Se espera que con la implementación de las API's se puedan adquirir y controlar las variables que los dispositivos de IoT y el microcontrolador tipo SoC vayan capturando, por medio de una página y servidor web para el envío y la comunicación de los datos recabados.

1.5 Justificación del Proyecto

El proyecto que se presenta, surge de acuerdo a la investigación realizada en el 2020 en donde se requería el diseño e implementación de una solución basada en IoT para la empresa Galeo Enrollables [4] para llevar un mejor registro de las actividades que se llevaban a cabo y de esta manera volverse mucho más eficientes.

Requiere el desarrollo de APIs que sean capaces de solicitar, adquirir y controlar variables de los dispositivos y sensores de IoT a través de una página web, empleando un microcontrolador System on a Chip que sea capaz de realizar estas funciones.

Para llevarlo a cabo, se iniciará desde lo más simple y sencillo, lo cual es comprender qué es y el funcionamiento que tiene cada dispositivo que será empleado, comprobando si existe compatibilidad entre los componentes y sensores del Internet de las cosas o si será necesario implementar algún protocolo o librería para su correcta funcionalidad.

Cabe mencionar que no únicamente serán empleados dispositivos de IoT, sino también, diversos protocolos de comunicación para que estos puedan interconectarse entre sí, como WiFi e I2C, de esta manera, se podrán capturar la temperatura, humedad, luminosidad, realizar la transferencia de datos, entre diversas cuestiones más.

Con esta implementación, la empresa se verá beneficiada, ya que se podrá dar solución al objetivo planteado anteriormente y de la misma manera, los empleados se verán beneficiados al contar con un sistema que es capaz de realizar diversas tareas a través de una página web empleando solamente el NodeMCU.

1.6 Alcances y Limitaciones

1.6.1 Alcances

Se pretende el desarrollo de APIs que sean capaces de solicitar, adquirir y controlar variables de los sensores de luminosidad, temperatura, humedad, proximidad y dispositivos como pantalla LCD, potenciómetros, servomotores, entre otros. (componentes de IoT) A través de una página web generada, empleando un microcontrolador System on a Chip que sea capaz de realizar estas funciones.

1.6.2 Limitaciones

- Variaciones entre las diferentes versiones del software de programación empleado y las librerías.
- Detectar posibles inconvenientes o errores al emplear el microcontrolador ESP-8266 para el desarrollo de las APIs.
- Incompatibilidades entre las librerías empleadas con el uso de los dispositivos del internet de las cosas.
- Procesos largos de configuración, debido a todos los dispositivos que serán empleados y las pruebas necesarias.

1.6.3 Plan de contingencia ante las limitaciones

- Investigar información acerca de los errores más frecuentes al emplear el chip ESP-8266 y la manera de resolverlos.
- Llevar un margen entre la fecha de entrega del proyecto y la configuración de los dispositivos.
- Tener en cuenta las documentaciones oficiales de las librerías en caso de que se presente una incompatibilidad.

1.7 La empresa

La empresa RODAS COMPUTACION, S.A. DE C.V. se creó en el año 2000 contando actualmente con 22 años de vida, en la cual ha alcanzado solidez y prestigio, ante sus clientes y proveedores, como una de las empresas de cómputo más fuertes de la zona. Han construido una imagen muy sólida y la han mantenido fuertemente apoyada por el correcto posicionamiento de su marca, la cual es sinónimo de calidad, servicio y buenos precios.

Está compuesta por una serie de principios, que se basa en saber quiénes son y qué creen; sus ideales y sus valores. Así como sus compromisos y responsabilidades con su público interno y externo.

1.7.1 Misión

Desarrollar y comercializar productos y servicios de Tecnología de Información, basados en estándares de calidad de procesos y dentro de un marco legal, que nos permita entender y satisfacer la necesidad de nuestros colaboradores y clientes.

1.7.2 Visión

Ser una de las 10 empresas más importantes de tecnología de información a nivel estatal. La visión en RODAS es el resultado de nuestro proceso de búsqueda, nuestro impulso para actuar día con día, resultado de nuestra experiencia y acumulación de información.

1.7.3 Valores

- Honestidad
- Calidad
- Productividad
- Capacitación
- Trabajo en equipo

Capitulo II: Metodología

2.1 Metodología SCRUM

La metodología Scrum es un proceso para llevar a cabo un conjunto de tareas de forma regular. Con este método de trabajo lo que se pretende es alcanzar el mejor resultado de un proyecto determinado. Las prácticas que se aplican con la metodología Scrum se retroalimentan unas con otras y la integración de las mismas.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

En esta ocasión se tomó en consideración su uso por la comodidad que ofrece sobre la estructuración de las tareas, de esa manera cumpliendo el objetivo de organizar y facilitar el trabajo a realizar, documentando todo lo que sea hecho en cada etapa involucrada.

2.2 Fases del ciclo de vida de SCRUM

En Scrum un proyecto se ejecuta en ciclos temporales cortos y de duración fija. Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

- ### 2.2.1 Inicio

La primera fase se encarga de estudiar y analizar el proyecto identificando las necesidades básicas del sprint. En el contexto de las metodologías ágiles, un sprint es un mini-proyecto que se interconecta con otros mini-proyectos para dirigirnos a los objetivos generales y específicos del proyecto general.

Las preguntas a hacer en la fase de inicio son:

1. ¿Qué quiero?
2. ¿Cómo lo quiero?
3. ¿Cuándo lo quiero?

- ### 2.2.2 Planificación y estimación

La segunda fase de Scrum incluye normalmente los siguientes pasos:

1. Crear, estimar y comprometer historias de usuario.
2. Identificar y estimar tareas.

3. Crear el sprint backlog o iteración de tareas.

La clave para llevar una buena administración de los proyectos es hacer una planificación y estimación del sprint, lo que te ayudará a establecer metas fijas y a cumplir con los plazos.

Tal vez esta sea la fase más importante del proyecto, ya que se tendrán que delegar las tareas correspondientes y hacer las estimaciones de tiempos de entrega, así como crear una lista ordenada para clasificar el trabajo según su prioridad.

- **2.2.3 Implementación**

Al llegar a la tercera de las 5 fases de Scrum, nos topamos con la implementación del proyecto. En la implementación se cumple con los siguientes procesos:

1. Crear entregables.
2. Realizar daily stand-up.
3. Refinanciamiento del backlog priorizado del producto.

En la fase de implementación o desarrollo no deberían hacerse cambios innecesarios de última hora (se supone que para evitarlo existe una fase de planificación).

- **2.2.4 Revisión y retrospectiva**

Una vez que ya todo está realizado e implementado, se deberá hacer la revisión del proceso, que no es más que la autocrítica o evaluación interna respecto al propio trabajo. Es importante sumar opiniones constructivas y aportar soluciones viables.

Entre los pasos más importantes para realizar en esta fase tenemos:

1. Demostrar y validar el sprint.
2. Retrospectiva del sprint.

- **2.2.5 Lanzamiento**

La última de las fases del método Scrum es el lanzamiento.

Con esto nos referimos al desenlace del proyecto y entrega del producto, donde deberías cumplir con 2 únicas tareas que son:

1. Enviar entregables.
2. Enviar retrospectiva del proyecto.

Capítulo III: Desarrollo del Proyecto

3.1 Aplicación de la metodología

- ### 3.1.1 Inicio

Se realizó una investigación respecto a qué necesidades se presentaban en la empresa.

Para ello, se tuvo una reunión con el asesor industrial, en dónde se llegó a la conclusión de que se necesita desarrollar APIs que sean capaces de realizar lecturas de las variables que estarán capturando los diversos dispositivos de IoT, empleando un microcontrolador tipo SoC, un servidor web y peticiones HTTP, para de esta manera lograr controlar y adquirir variables por medio de una página web.

Para llevar a cabo la meta, será necesario separar el proyecto en sprints, que serían "mini-proyectos" que vayan desglosando el proyecto principal, de esta manera se irán integrando durante la etapa de la implementación, creando así el proyecto final.

El objetivo a desempeñar es claro, y para ello, dentro de las necesidades que se identificaron se encontraron los siguientes requisitos:

1. Desarrollar e implementar APIs empleando dispositivos de IoT para la comunicación de datos.
   Durante este punto se tomarán en cuenta dos conceptos importantes para el cumplimiento de la tarea;

   - Evidencia gráfica (fotografías o videos), de cómo se fue desarrollando el proyecto.
     De esta manera se podrá tener un mayor entendimiento de los pasos que se fueron siguiendo para desarrollar el proyecto, así mismo, funcionará como evidencia de su realización que podrá ser adjuntada a la documentación.

   - Documentación del desarrollo del proyecto.
     Se irá documentando el funcionamiento de los dispositivos empleados, ya que se emplearán diversos dispositivos de IoT como controladores, así como el papel que desempeñarán dentro del proyecto principal.

Todo esto será necesario para lograr, controlar y adquirir variables como lo serían la temperatura y la humedad a través de peticiones HTTP, por medio de una página web, que se encontrará alojada en un servidor web.

- **3.1.2 Planificación y estimación**

Durante esta fase se realizó una lista como planeación del seguimiento del desarrollo del proyecto, así mismo, se realizaron algunas historias del usuario que van de la mano con la metodología, se pondrán a grandes rasgos las actividades o tareas que se deberán realizar para completarlo, para ello, se elaboró un cronograma de actividades a seguir, el cual consta de diversos pasos repartidos en un lapso de 13 semanas.

Historias del Usuario.	
Número: 1	**Usuario**: Hugo Rodríguez Astorga.
Nombre de la historia: Creación de APIs.	
Prioridad: Alta**.**	**Riesgo en desarrollo**: Media.
Persona responsable: Osdair Jaimes Nava	
Descripción: Requiero APIs que sean capaces de realizar lecturas de las variables que estarán capturando los diversos dispositivos de IoT, empleando un microcontrolador tipo SoC, un servidor web y peticiones HTTP, para de esta manera lograr controlar y adquirir variables por medio de una página web.	

Tabla 1: Primera historia del usuario.

Historias del Usuario.	
Número: 1.1	**Usuario**: Hugo Rodríguez Astorga.
Nombre de historia: Selección de software.	
Prioridad: Media**.**	**Riesgo en desarrollo**: Baja.
Persona responsable: Osdair Jaimes Nava	
Descripción: Seleccionar adecuadamente el software que será empleado para la programación de los dispositivos.	

Tabla 2: Segunda historia del usuario.

Historias del Usuario.	
Número: 1.2	**Usuario**: Hugo Rodríguez Astorga.
Nombre de la historia: Selección de hardware.	
Prioridad: Media**.**	**Riesgo en desarrollo**: Baja.
Persona responsable: Osdair Jaimes Nava	
Descripción: Seleccionar correctamente el hardware que se ocupará para la implementación en las API's (llámese, sensores de IoT, Microcontroladores, dispositivos de IoT, entre otros).	

Tabla 3: Tercera historia del usuario.

Historias del Usuario.	
Número: 1.3	**Usuario**: Hugo Rodríguez Astorga.
Nombre de la historia: Comunicación serial.	
Prioridad: Alta**.**	**Riesgo en desarrollo**: Media.
Persona responsable: Osdair Jaimes Nava	
Descripción: Se realizarán pruebas de que exista una comunicación serial y que la información que sea enviada se pueda mostrar dentro de un monitor serial con el que contará el prototipo.	

Tabla 4: Cuarta historia del usuario.

Historias del Usuario.	
Número: 1.4	**Usuario**: Hugo Rodríguez Astorga.
Nombre de la historia: Controlar servomotores.	
Prioridad: Baja**.**	**Riesgo en desarrollo**: Baja.
Persona responsable: Osdair Jaimes Nava	
Descripción: Se requiere poder controlar diversos servomotores a través de un solo prototipo.	

Tabla 5: Quinta historia del usuario.

Historias del Usuario.	
Número: 1.5	**Usuario**: Hugo Rodríguez Astorga.
Nombre de la historia: Código empleado.	
Prioridad: Baja**.**	**Riesgo en desarrollo**: Baja.
Persona responsable: Osdair Jaimes Nava	
Descripción: Para el funcionamiento de los sensores que serán empleados va a ser necesario desarrollar un código que hagan que estos puedan realizar las lecturas adecuadas, para ello se producirá un código adecuado.	

Tabla 6: Sexta historia del usuario.

Historias del Usuario.	
Número: 1.6	**Usuario**: Hugo Rodríguez Astorga.
Nombre de la historia: Protocolo I2C.	
Prioridad: Alta**.**	**Riesgo en desarrollo**: Alta.
Persona responsable: Osdair Jaimes Nava	
Descripción: Uno de los métodos de comunicación será a través del protocolo I2C, para los sensores de luminosidad y la pantalla LCD, por ello, será necesario realizar pruebas de funcionamiento y evaluar su rendimiento.	

Tabla 7: Séptima historia del usuario.

Historias del Usuario.	
Número: 1.7	**Usuario**: Hugo Rodríguez Astorga.
Nombre de la historia: Servidor web.	
Prioridad: Medio**.**	**Riesgo en desarrollo**: Medio.
Persona responsable: Osdair Jaimes Nava	
Descripción: Para poder solicitar las variables se empleará una página web, para ello, será necesario configurar un servidor web que sea capaz de realizar esa tarea, así mismo que sea capaz de mostrar esa información.	

Tabla 8: Octava historia del usuario.

Al recolectar las historias del usuario, se decidió realizar un cronograma de actividades que ayudara a facilitar el identificar las tareas que serán necesarias completar.

Número:	Actividades a realizar:	Duración:
1	Selección del software que será empleado para la programación.	09/01/23 – 15/01/23
2	Selección del hardware que se utilizará.	16/01/23 – 22/01/23
3	Pruebas de comunicación serial para muestreo de información en un monitor serial.	23/01/23 – 29/01/23
4	Pruebas de control para varios servomotores a través de un monitor serial.	30/01/23 – 05/02/23
5	Desarrollo del código para la lectura de los sensores de IoT.	06/02/23 – 12/02/23
6	Pruebas de comunicación del protocolo I2C para la comunicación del sensor de luminosidad y la pantalla LCD.	13/02/23 – 19/02/23
7	Creación y configuración de un punto de acceso para la configuración de los dispositivos IoT.	20/02/23 – 26/02/23
8	Implementación y configuración de un servidor web para pruebas del muestreo de información.	27/02/23 – 05/03/23
9	Diseño y programación de las APIs de lectura.	06/03/23 – 19/03/23
10	Pruebas de la integración de las APIs con los sensores seleccionados.	20/03/23 – 26/03/23
11	Documentación del proyecto.	27/03/23 – 02/04/23

Tabla 9: Cronograma de actividades a realizar.

- **3.1.3 Implementación**

Teniendo en cuenta los puntos planteados anteriormente en la etapa de planeación, se realizó lo siguiente:

3.1.3.1 Selección del software para la programación

Se utilizó el IDE Arduino. IDE proviene de Integrated Development Environment (Entorno de desarrollo integrado); la forma de escribir el programa está basada en el lenguaje de programación C y se decidió ocupar este debido a los diversos usos y aplicaciones que se le puede dar.

El IDE Arduino se ve como se muestra a continuación:

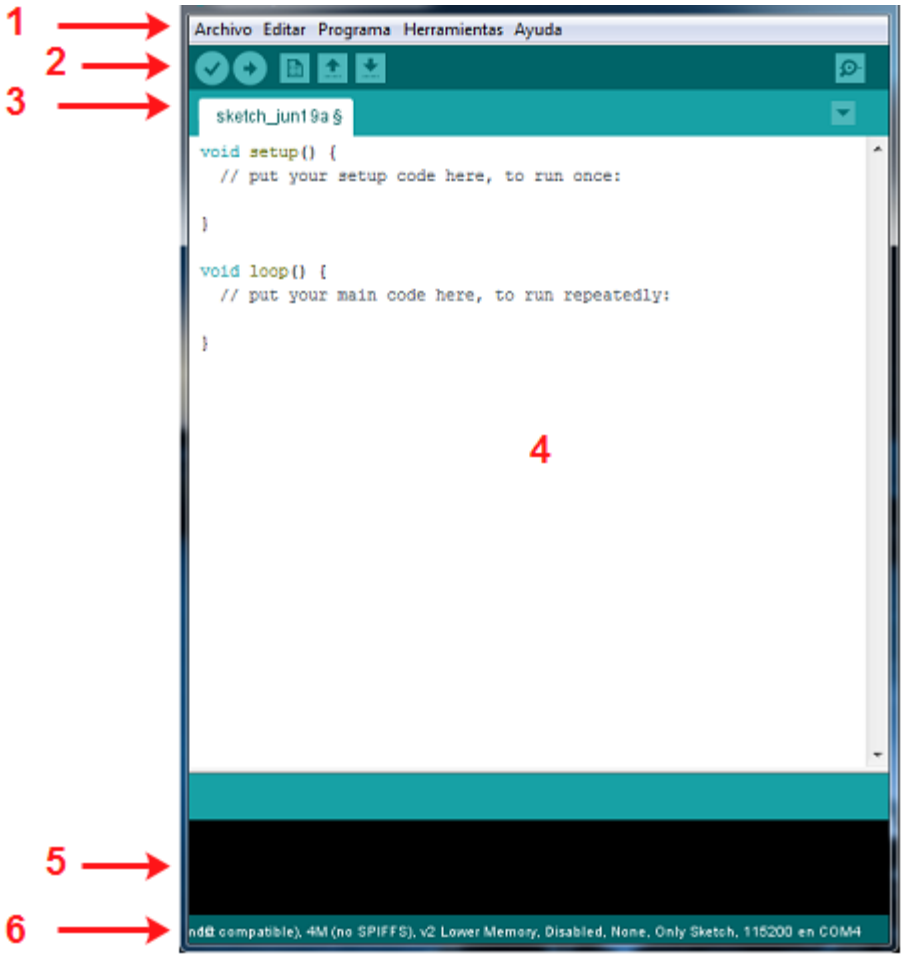


Ilustración 5: Entorno del software IDE de Arduino.

El área horizontal marcada como 1, contiene las siguientes opciones:

- **Archivo**: Contiene opciones para generar, guardar o cargar programas de Arduino, entre otras opciones.
- **Editar**: Opciones para modificar el programa o sketch actual.
- **Programa**: Opciones de verificación del sketch actual o subirlo (programar) a la placa conectada, así como un gestor de librerías, etcétera.
- **Herramientas**: Las opciones más destacadas son el monitor serie, la selección de la placa conectada y el puerto en donde se encuentre conectada.
- **Ayuda**: Opciones para la solución de problemas.

El área marcada como 2, tiene las siguientes opciones:

- **Verificar**: Analiza el sketch actual para buscar errores de ejecución. Si los hay, se imprimen en la consola () con letra de color naranja. Si no los hay, sólo se muestran letras blancas que especifican el tamaño del sketch.
- **Subir**: Programa la placa conectada.
- **Nuevo**: Abre una nueva ventana de IDE Arduino.
- **Abrir**: Abre una ventana para buscar y abrir archivos de Arduino.
- **Salvar**: Guarda los cambios del sketch en el archivo de Arduino actual.
- **Monitor serie**: Abre el monitor serie.

El área 3 contiene las pestañas que conforman al sketch. En la orilla a la derecha, esta una flecha con opciones para más pestañas.

El área 4 es toda esa pantalla blanca, en la cual se escribe el programa también conocido como sketch.

El área 5 es una consola que muestra mensajes según lo que se esté realizando. Se mostrará errores por ejemplo en una compilación si es que los hay, o cuando se intente subir el sketch a la placa. También escribe mensajes del proceso de programación de la placa, entre otros.

Por último, el área 6 muestra el nombre de la placa conectada, y a que puerto se encuentra conectada según como se haya seleccionado en la opción Herramientas.

3.1.3.2 Selección del hardware

Para cubrir el segundo punto, se seleccionará el hardware que será empleado para llevar a cabo el cumplimiento del proyecto. Recordemos que hardware es el término que se utiliza para todas partes físicas, tangibles, de un sistema informático, sus componentes eléctricos, electrónicos, electromecánicos. Los cables, así como los muebles o cajas, los periféricos de todo tipo, y cualquier otro elemento físico involucrado.

Las partes físicas que serán empleadas son las siguientes:

- El NodeMCU.

  Es un dispositivo basado en el sistema de comunicación WiFi ESP8266. Existen diferentes versiones de placas que incluyen al ESP, pero en el caso del NodeMCU (y en algunos más), se da mayor acceso a los GPIO (General Purpose Input/Output) con los que cuenta el ESP.

  El NodeMCU facilita en gran manera tanto el uso como la programación del ESP, convirtiéndolo en un módulo versátil para el desarrollo de proyectos enfocados a la conexión WiFi.

  El ESP8266, es un sistema que funciona con una alimentación de 3.3V, por ende, el NodeMCU incluye un regulador de voltaje para la correcta energización del ESP, pero sólo se encuentra protegido en este aspecto. En el caso de voltajes de entrada en los pines del Node, estos no deben superar los 3.3V, o se dañará al ESP.

  También cuenta con 2 botones: Reset (RST) y Flash. De estos 2 lo más probable es que sólo se utilice el botón de Reset, el cual ejecuta de nueva cuenta el programa que tenga guardado.

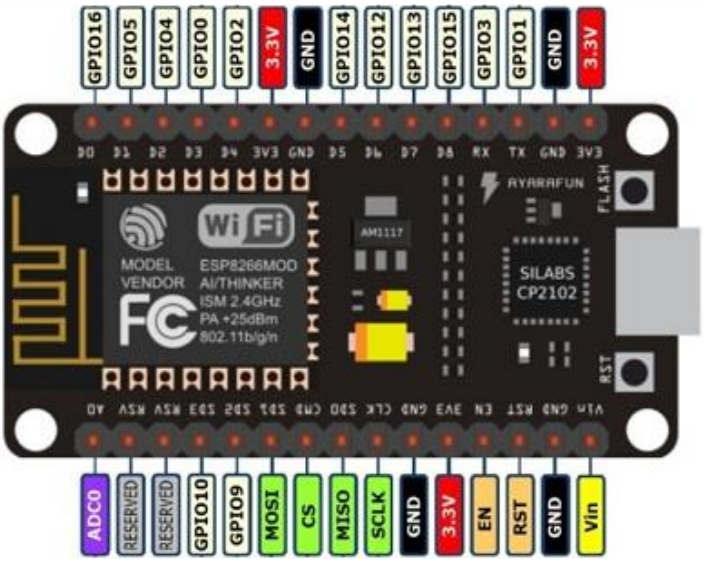


Ilustración 6: Pines del NodeMCU.

- Tarjeta Integradora Mecatrónica.

La tarjeta integradora mecatrónica, es una tarjeta hecha para que el NodeMCU trabaje con diferentes sensores y actuadores, esta tarjeta es muy versátil y se describe a sí misma, de manera que será fácil adaptarse a ella y conocer sus puertos para conexiones.

Cuando requiera de usar las entradas de CONTROL o 127V, deberá usar desarmadores de cruz o planos. La conexión de 127V, debe ir conectada a un tomacorriente a través de un cable con clavija, y sirve para 2 cosas: Alimentar los dispositivos de la tarjeta a través de la mini fuente 127VCA – 5V y para que algún dispositivo de 127Vca se alimente a través de la conexión CONTROL, cuando se use el Relé. El Relé es el cuadro grande que dice K1.

Para lograr alimentar el NodeMCU usando la mini fuente, debe conectar entre sí los pines que dicen EXTERNA (refiriéndose a la alimentación externa) mediante un cable o jumper, aunque esto también permite que los dispositivos que usen 5V se alimenten de los 5V del puerto USB, cuando el Node se encuentre conectado a la computadora. Por esa razón, se recomienda no conectar el cable USB a la computadora y al Node y alimentar con la mini fuente al mismo tiempo, es decir, manteniendo conectados los pines que dicen EXTERNA. La única forma de tener conectado el cable USB y alimentar algunos dispositivos de forma externa, sería desconectando los pines de EXTERNA.

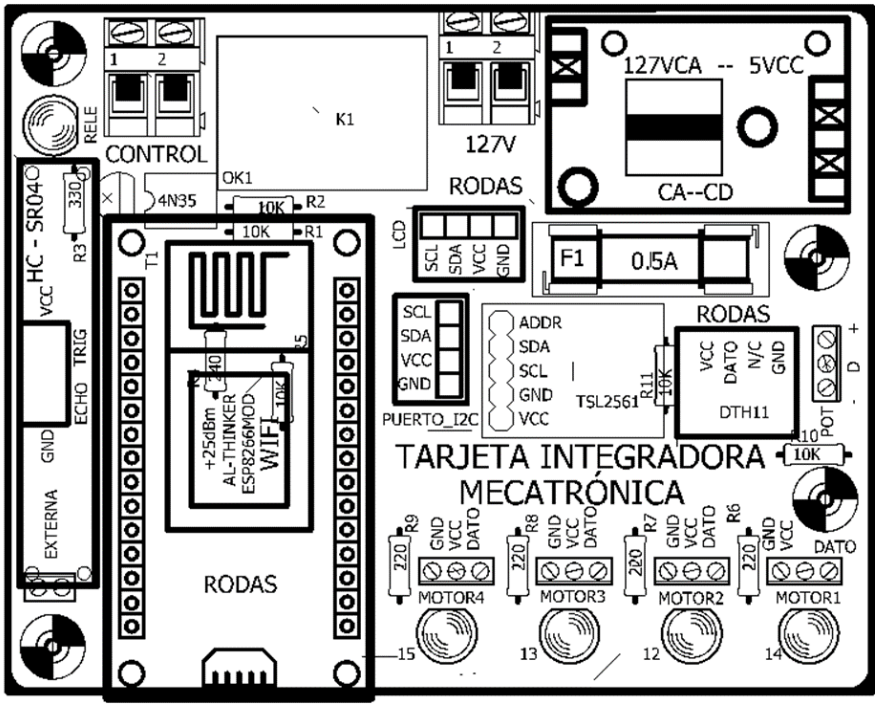


Ilustración 7: Esquema de la Tarjeta Integradora Mecatrónica.

- LEDs.

LED es el acrónimo de Light Emitting Diode (Diodo emisor de luz). Existen LED de diferentes colores y tamaño, siendo los más comunes los de 5mm; por lo regular, su forma es redondeada en la punta, aunque también los hay con formas rectangulares. Al ser un diodo, solo conduce electricidad en un sentido, esto quiere decir que se debe observar cual terminal se conecta al positivo de la fuente y cual al negativo (tierra o masa). Sus terminales se identifican porque una es más larga, siendo esta la que va conectada al positivo de la fuente, y la otra a tierra (representado como GND).

Ilustración 8: LEDs

- Resistencias:

Componente que limita la corriente que fluye en un circuito. La unidad en la que se mide la resistencia es Ohm, y se representa con la letra griega omega: Ω. Se usan los prefijos de medición Kilo (K) y Mega (M). El valor de una resistencia se representa con 4 o 5 bandas de colores, y se leen de izquierda a derecha teniendo los siguientes valores:

- La primera banda equivale a su valor x 10.
- La segunda banda equivale a su valor x 1. La primera y la segunda se suman.
- La tercera banda es el multiplicador. Su valor es la potencia de una base 10 (10potencia) y el resultado multiplica a la suma de las 2 bandas ya mencionadas; o también se podría decir que su valor es la cantidad de ceros que se añaden a dicha suma.
- La última banda es la tolerancia de la resistencia, es decir, su valor podría ser un poco más grande o pequeño que lo calculado anteriormente.

Para entender mejor los puntos anteriores, en la siguiente tabla se muestran los diferentes colores, junto con sus valores.

Color	1ª banda x 10	2ª banda x 1	3ª banda Multiplicador	4ª banda Tolerancia
Negro	0	0	1Ω	
Café	1	1	10Ω	±1%
Rojo	2	2	100Ω	±2%
Naranja	3	3	1000 o 1KΩ	
Amarillo	4	4	10KΩ	
Verde	5	5	100KΩ	±0.5%
Azul	6	6	1MΩ	±0.25%
Violeta	7	7	10MΩ	±0.1%
Gris	8	8		±0.05%
Blanco	9	9		
Oro			0.1Ω	±5%
Plata			0.01Ω	±10%

Ilustración 9: Valores de las resistencias.

Ilustración 10: Resistencias de diversos valores.

- Capacitores

Estos componentes almacenan y liberan energía eléctrica en un circuito. Se comporta como una pequeña batería recargable. Los capacitores se usan en filtros. La unidad para medir la capacitancia es Faradio simplificado como F. Es una unidad muy grande, y a menudo se le encuentra con prefijos como pico (p), nano (n) y micro (µ). A menudo se les coloca entre el voltaje y tierra (ground) de un sensor o motor para ayudar con la fluctuación del voltaje.

Existen diferentes tipos de capacitores. Algunos de ellos no están polarizados (por ejemplo, los cerámicos) y otros si (por ejemplo, los electrolíticos). El que estén polarizados significa que una de sus terminales debe conectarse obligatoriamente al voltaje positivo, mientras que la otra terminal se conecta a tierra (GND); de este tipo de capacitores, es común que su polaridad esté escrita sobre el componente, con el símbolo – en la terminal a tierra.

Ilustración 11: Capacitores de diversos voltajes.

- Transistor

Es un dispositivo semiconductor, usado para conmutar o amplificar una señal. Se le puede imaginar como un interruptor que funciona con una corriente muy pequeña, un switch controlado por corriente. Tiene 3 terminales, conocidas como: base (B), colector (C) y emisor (E).

El emisor es el que emite los electrones, mientras que el colector es el que los recolecta; la base controla ese flujo de electrones. Si una corriente pequeña fluye de la base al emisor, una corriente mucho más grande fluirá del colector al emisor. La cantidad de la corriente C-E depende de una constante específica del tipo de transistor. A esta constante se le conoce como ganancia de corriente directa y su símbolo es la letra griega beta (β), ó HFE. Sin embargo, llega un punto en el que el transistor ya no amplifica, y se comporta sólo como un switch.

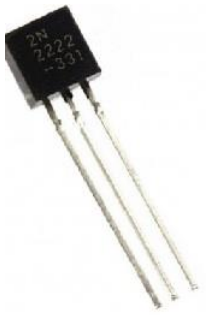


Ilustración 12: Transistor.

- Diodo

Es un dispositivo semiconductor. Una de sus propiedades, es que conduce la electricidad en una sola dirección. Tiene 2 terminales conocidas como ánodo y cátodo, y para lograr que el diodo conduzca la energía eléctrica, el ánodo debe estar conectado a positivo, mientras que el cátodo a tierra o negativo.

Los diodos tienen un consumo de voltaje de 0.5 a 0.7V. Esto significa que, si se realiza una medición de voltaje, esta será alrededor de 0.6V más pequeña después del diodo. También tiene sus límites, pues si el voltaje en polaridad inversa es demasiado grande, causará que la corriente fluya en el sentido equivocado.

Ilustración 13: Diodos.

- Resistencia variable

Al igual que las resistencias antes mencionadas, este componente limita el flujo de la corriente, pero tiene la característica de modificar su valor. Estos componentes tienen 3 terminales: 2 de ellos están conectados a una resistencia de un valor fijo, mientras que la otra terminal es móvil mediante una perilla, dividiendo así en 2 partes a la resistencia fija y logrando de esta forma, que esas 2 partes tengan diferente valor de Ohm. También es conocido como potenciómetro.

Las terminales a ambos lados se conectan al voltaje, y la del centro dejará notar el cambio de voltaje en ella cuando se mueva la perilla. También existen diferentes valores de potenciómetros.

Ilustración 14: Potenciómetro

.

- Push button

También llamado botón momentáneo, es un componente usado para realizar conexiones de forma momentánea. Cuenta con 4 terminales, pero internamente es un par de terminales, estando divididos en 2 pines cada una.

Mientras se mantenga pulsado el botón, se realizará una conexión entre las terminales. A este tipo de push button se le clasifica como “normalmente abierto”. Por el contrario, existen push button que se pulsan para desconectar las terminales, conocidos como “normalmente cerrado”.

Ilustración 15: Botón de pulsación.

- Relé o relevador

Un relé es un componente que funciona como un interruptor, pero activado por energía eléctrica. Internamente posee un electroimán, que cuando es energizado, moverá una pieza metálica que hará la función de un switch.

Ilustración 16: Relevador.

Regularmente tiene 5 terminales, las cuáles se podrían dividir en 2 partes: En una parte se tienen 2 y en la otra los 3 restantes. Las 3 terminales corresponden al interruptor, siendo una de las terminales la común, y las restantes a donde se conectará la común.

Las 2 terminales de la otra parte, activarán un electroimán cuando se les suministre corriente eléctrica, cambiando de estado el switch interno.

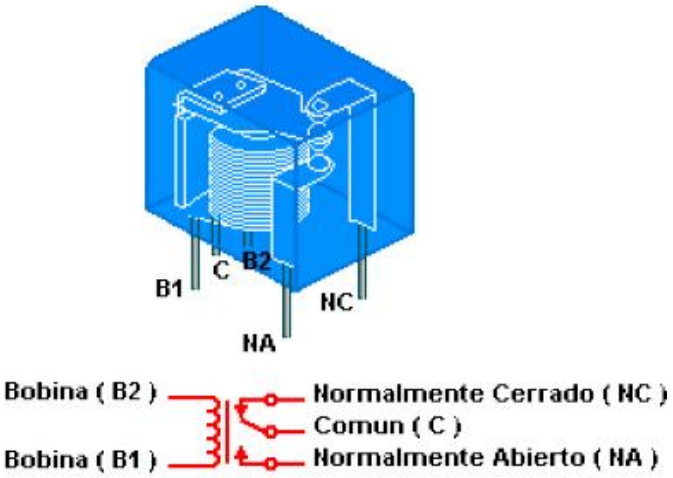


Ilustración 17: Composición del relevador.

Regresando a los pines del interruptor, el común estará conectado hacia uno de los otros 2 (normalmente cerrado), pero cuando se activa el electroimán, éstos se desconectan, pero ahora se conectará con la otra terminal, siempre y cuando se mantenga energizado

el electroimán; si se deja de suministrarle energía, entonces el interruptor volverá a su estado normal.

Los pines para activar el electroimán suelen funcionar a voltajes bajos, pero con corriente directa. Por otro lado, las terminales para la conexión pueden funcionar con corriente directa o alterna, y el voltaje y corriente que soporten dependerá de las especificaciones que tenga, usualmente escritas en la parte superior del relevador.

- Servomotor.

Un servomotor, es un dispositivo basado en un motor, pero con la capacidad de controlar la posición de una flecha o eje. Este control se da en grados. Existen servomotores de diferentes grados, es decir, el giro máximo que pueden dar, pero los más comunes son los de 180°.

Ilustración 18: Servomotor.

Su conexión consta de 3 cables:

- VCC: Voltaje de alimentación (5V). Cable Rojo o Naranja
- GND: Tierra. Cable Negro o Café
- Señal PWM: Conectado al pin PWM. Recibe la posición en grados que debe tomar. Cable Naranja o Amarillo (si VCC es Naranja)

Para controlar este componente desde el IDE Arduino, se necesita una librería llamada Servo, que afortunadamente viene incluida en la instalación del IDE Arduino.

- Microcontrolador SoC.

Un microcontrolador SoC integra todas o la mayor parte de componentes necesarios para el funcionamiento de un ordenador. Entre ellas se incluirá casi siempre una CPU, por lo que podemos hablar de un procesador que incluye más componentes en su interior que normalmente estarían relegados a chips externos a este.

Circuito integrado que incorpora gran parte de los componentes de un ordenador o cualquier otro sistema informático o electrónico. Habitualmente integra núcleos de procesador, el sistema de gráficos, memoria RAM y, posiblemente, la ROM también, controladores de interfaz para USB, tecnología inalámbrica, reguladores de voltaje, etc.

La diferencia principal de un SoC con un microcontrolador tradicional es que estos rara vez disponen de más de 100 Kilobytes de memoria RAM, y gran parte de estos son estructuras mono-chip, mientras que el término SoC es usado para procesadores más potentes y complejos, que dependen de chips o módulos de memoria externos para ser eficaces

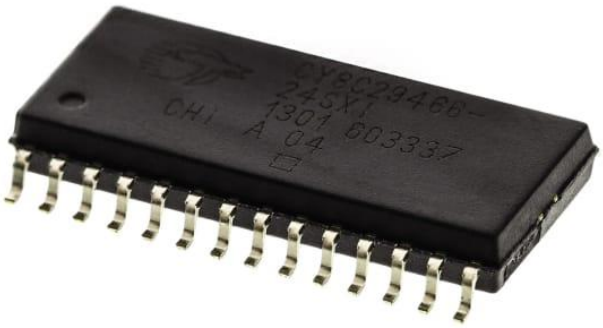
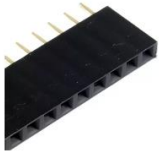


Ilustración 19: Microcontrolador SoC.

A continuación, se mostrará una lista de todos los dispositivos que serán empleados en el desarrollo del proyecto:

Componente	Cantidad	Imagen de referencia:
Porta fusible	1	
Clema de presión 2 terminales	2	
LED 5 mm	5	

Resistencia 220Ω 1/4W	4	
Resistencia 240Ω 1/4W	1	
Resistencia 330Ω 1/4W	1	
Resistencia 10KΩ 1/4W	3	
Tira de Headers	1	
Tira de pines	1	
Transistor 2N2222	1	
Optoacoplador	1	
Relevador de 5Vcc	1	
Placa fenólica 10x10 una cara	1	

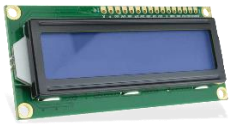
Sensor TSL2561	1	
Sensor HC-SR04	1	
Sensor DHT22	1	
Módulo I2C para LCD	1	
Pantalla LCD	1	
NodeMCU	1	

Tabla 10: Dispositivos empleados en el desarrollo del proyecto.

3.1.3.3 Pruebas de comunicación serial para muestreo de información en un monitor serial

Para realizar la tercera actividad, se realizaron pruebas de la comunicación serial, pero para ello, es importante saber qué es el monitor serial. El monitor serial es una interfaz gráfica entre el NodeMCU y el usuario, que permite desplegar información del NodeMCU. Es una gran ayuda porque puede ser usado, por ejemplo, para mostrar el proceso de la ejecución del código, o cómo se van modificando algunos valores y también puede ser usado para enviar datos al NodeMCU.

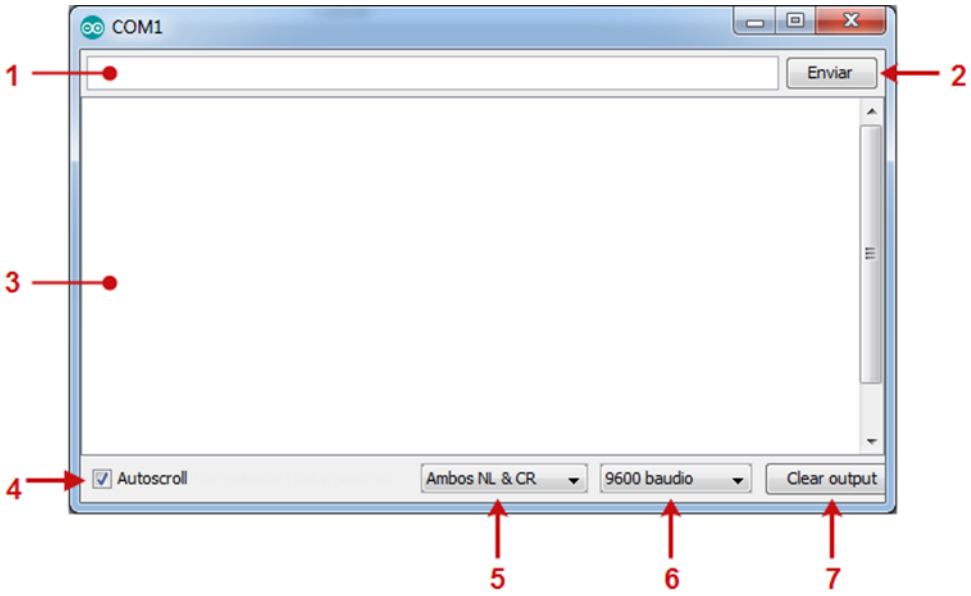


Ilustración 20: Componentes del monitor serial.

En la imagen anterior se muestra como se ve el monitor una vez abierto. También se da una explicación de sus herramientas.

1. Es el área donde el usuario puede escribir los datos que desee enviar al Arduino o en este caso, al NodeMCU.
2. Una vez escritos los datos, con este botón se envían tales datos.
3. Área donde se imprime la información que envía el Arduino o Node.
4. Al marcar la casilla, el área 3 se “mueve” siguiendo los últimos datos impresos.
5. Es el modo en el que se imprimen los datos.
6. Velocidad de transmisión serial. Su valor debe coincidir con el que se inicializó la comunicación serial en el código, en la función setup.
7. Limpia los datos del área 3.

Algunas funciones que se suelen ocupar mucho en la comunicación serial son las siguientes:

Inicializar comunicación serie

Cuando se hace uso de la comunicación serie, se debe inicializar la misma en el código y establecer los baudios. Por ejemplo, para el NodeMCU, se escribe la siguiente línea en la función setup.

```
Serial.begin(115200);
```

Al ser la línea que inicializa la comunicación serie, debe ser escrita antes que cualquier otra línea que haga uso de esta comunicación.

Enviar datos

El envío de datos requiere que estos sean de tipo String, ya que la comunicación serie lee los datos como caracteres; por ejemplo, si se envía un número, aun cuando se escriba como tal, será transmitido como caracteres. Por ejemplo, escribir hola en el puerto serie.

```
Serial.write("hola");
```

Imprimir datos en el monitor serie

La siguiente línea imprime los datos, pero manteniendo el cursor sobre la misma línea, es decir, la próxima vez que se impriman datos, se escribirán a la derecha de los primeros. Ejemplo: Escribir Hola en el monitor serie.

```
Serial.print("Hola");
```

Para imprimir datos y después realizar un salto de línea (o simplemente realizar un salto de línea), para que la siguiente vez que se impriman datos sea en un nuevo renglón, se usa la siguiente línea. Por ejemplo, si se escribe Adios en el monitor serie.

```
Serial.println("Adios");
```

En caso de solo hacer un salto de línea, se dejan vacíos los paréntesis.

Datos disponibles

La siguiente línea, obtiene la cantidad de caracteres que existen en el puerto serie, pero no los lee.

```
Serial.available();
```

Si ese dato se desea guardar en una variable, se puede escribir como sigue:

```
int datos = Serial.available();
```

Leer datos: Como ya se mencionó, la comunicación serie transmite uno a uno los caracteres, por lo tanto, la lectura se realiza recibiendo uno a uno los caracteres. Básicamente, la siguiente línea sirve para leer un carácter:

```
Serial.read();
```

El dato se debe guardar en una variable de tipo char, por lo tanto, la línea puede quedar así:

```
char dato = Serial.read();
```

Sin embargo, usarlo por sí solo no sirve de mucho cuando hay varios caracteres entrantes. En este caso, es necesario usar esta línea en combinación con un ciclo para leer varios datos, esto se explicará más adelante durante las prácticas.

Cuando se lee un carácter de esta forma, ese carácter es borrado del puerto serial. La manera en que se leen los caracteres es, desde nuestra perspectiva, como cuando leemos un texto de izquierda a derecha.

Teniendo todo esto en cuenta, para realizar las pruebas de comunicación se empleó el siguiente sencillo código:

```
void setup()
{
  Serial.begin(115200);
  Serial.println();
  Serial.print("Hola ");
  Serial.println("mundo");
}

void loop()
{
  while(Serial.available())
  {
    char d = Serial.read();
    Serial.println(d);
  }

  delay(50);
}
```

Ilustración 21: Código para pruebas de comunicación.

Primero, la frase “Hola mundo” está dividida en 2 partes, ya que el primero no tiene salto de línea, causa que lo siguiente que se imprima lo haga sobre la misma línea que la anterior.

En este caso, el NodeMCU estará revisando que existan datos disponibles en el bus serial, y cuando encuentra datos, se inicia el ciclo while. Este ciclo se ejecutará una y otra vez hasta que la condición deje de cumplirse, en este caso, hasta que ya no existan datos en el bus serial.

Se leen uno a uno los caracteres, y se guardan en una variable tipo char, e inmediatamente se muestran en el monitor serial realizando saltos de línea cada que se lee un carácter.

El ciclo loop, también tiene un pequeño retraso de 50 milisegundos, es decir, cada 50 milisegundos se está revisando que existan datos en el bus serial.

La consecuencia de esto, es que cada palabra o número enviado por el monitor serial, será mostrado en el mismo, pero en carácter por carácter. Si abre el monitor serial después de subir el programa, es posible que no se muestren caracteres, pulse el botón reset del Node, deberán aparecer símbolos y debajo de ellos, la frase Hola mundo. Para enviar los datos, estos se escriben en la barra de texto blanca, y se pulsa Enviar o se pulsa la tecla Enter.

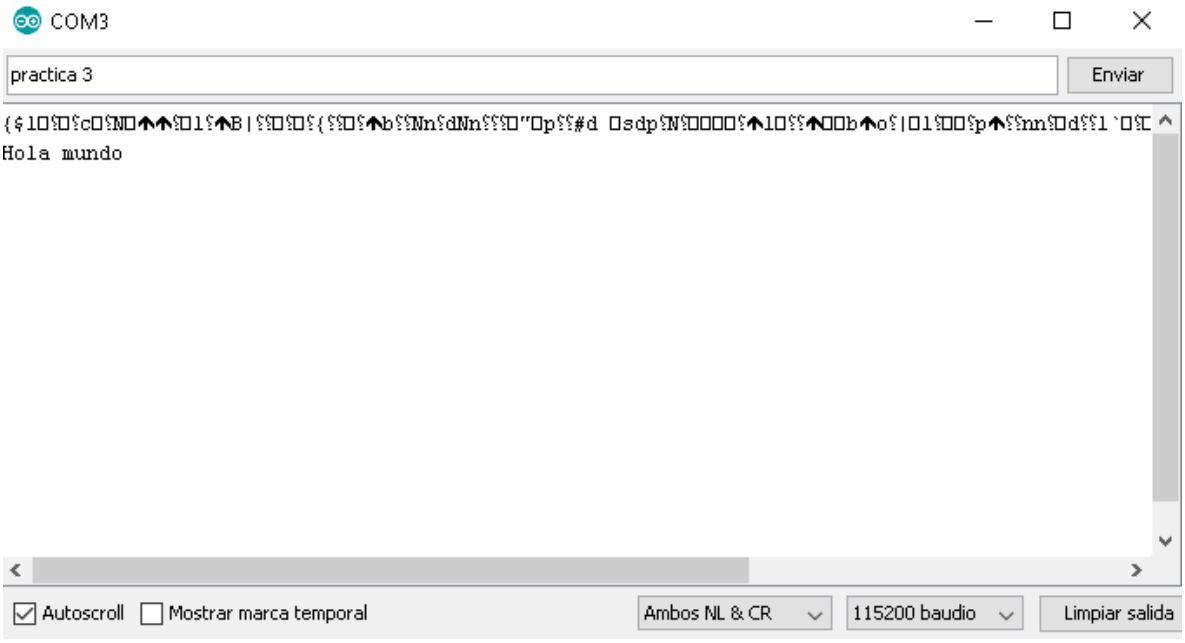


Ilustración 22: Resultado de las pruebas de comunicación.

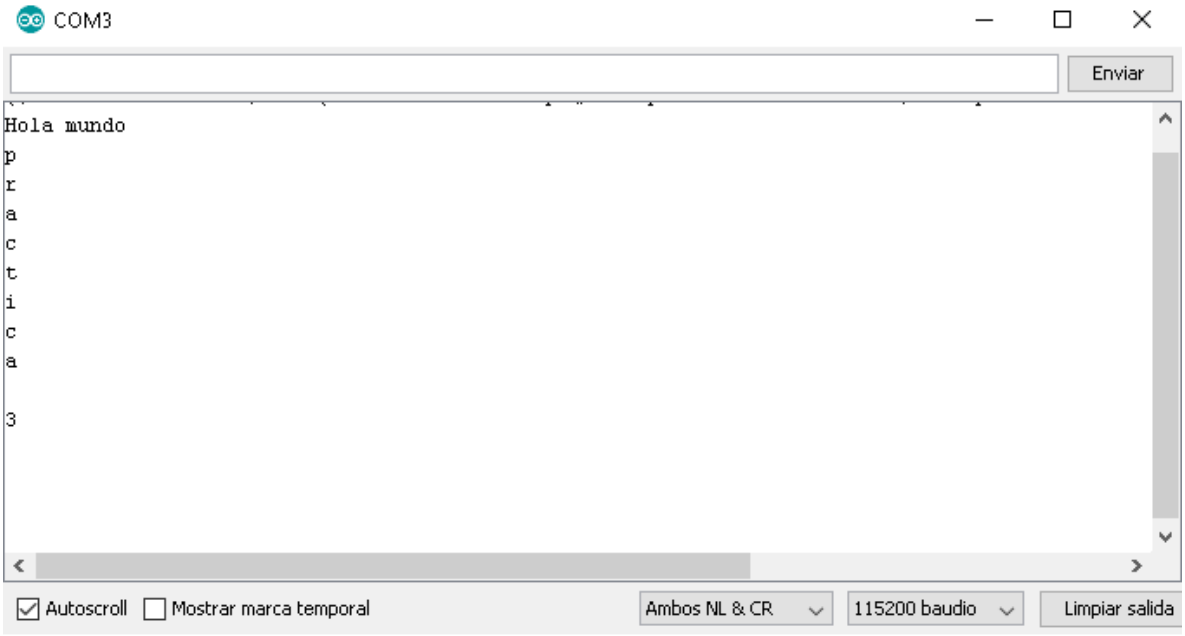


Ilustración 23: Resultado de las pruebas de comunicación, parte 2.

Algunos símbolos o signos ortográficos no son reconocidos, y se imprimen con signos de interrogación, por ejemplo, la letra ñ, las vocales con tilde (á, é, í, ó, ú), entre otros.

3.1.3.4 Pruebas de control para servomotores a través de un monitor serial

Continuando con las pruebas de control para servomotores a través de un monitor serial, se debe de tener en cuenta que para alimentar el servomotor se necesitan 5V, los cuales se pueden obtener del cable USB conectado a la computadora.

El código para controlar el servomotor hará uso del monitor serial para el control del servomotor, por lo tanto, el servomotor deberá conectarse a la terminal que dice MOTOR3, ya que el siguiente código hace uso del pin 13. Asegúrese de conectar correctamente el servomotor a los pines de la tarjeta, recuerde que el cable oscuro en el servo (café o negro) es la conexión a GND, y le siguen Vcc y finalmente, Dato.

Posteriormente, el código que fue empleado para llevar a cabo las pruebas fue el siguiente:

```
#include <Servo.h>

Servo myServo;
int angulo;

void setup()
{
  Serial.begin(115200);
  myServo.attach(13);
}

void loop()
{
  if(Serial.available())
  {
    String var;
    while(Serial.available())
    {
      char c = Serial.read();
      if(c > 13){var += c;}
    }
    angulo = var.toInt();
    if(angulo >= 0 && angulo <= 180)
    {
      myServo.write(angulo);
      Serial.println(angulo);
    }
    else
    {
      Serial.println("Escriba un numero entero de 0 a 180");
    }

  }

  delay(50);
}
```

Ilustración 24: Código empleado para pruebas de servomotores.

Primero se incluye la biblioteca y luego se declara un objeto Servo, en este caso se llama myServo, dicho nombre puede ser diferente. También se ha declarado una variable de tipo int.

En setup, se asigna el pin PWM conectado al cable PWM del servo. Para esta práctica, se usa uno de los pines usados en la práctica 5, así que es normal que el LED se encienda durante esta práctica.

En loop, se envía el ángulo al servomotor. Nótese que los datos obtenidos del Monitor Serial primero deben convertirse a int, ya que los datos de la comunicación serie se leen como caracteres. Los números enviados por el Monitor Serial, siempre deben ser enteros, además de tener un valor de entre 0 y 180, ya que esos son los grados a los que se puede mover el servomotor. La ventaja de convertir a entero, es que si se envía un numero con fracciones, es redondeado.

Y para asegurar que el valor respete el rango de 0 hasta 180, se usa una condición if donde se escriben tales límites en forma de 2 condiciones unidas. Si se cumple la condición, entonces se envía la instrucción al servomotor, además se imprime en el Monitor serial el dato recibido. Si no se cumple, se imprime un aviso en el Monitor serial.

3.1.3.5 Desarrollo del código para la lectura de los sensores de IoT

Para llevar a cabo este punto, se pueden emplear los sensores DHT11 o DHT22, estos son sensores básicos y de bajo costo, que miden la humedad y la temperatura, usando un sensor de humedad capacitivo y un termistor para medir el aire a su alrededor. Aunque el sensor por si solo tiene 4 terminales, en realidad solo se usan 3.

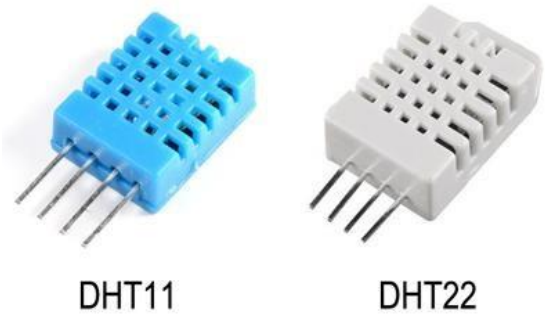


Ilustración 25: Sensores de temperatura y humedad DHT11 y DHT22.

La diferencia física entre el DHT11 y el DHT22 es su color, pero lo importante son sus características de medición, ya que el segundo es más preciso y con un rango de medición mayor. Esto a su vez causa que el DHT22 sea un poco más caro que el DHT11.

Descripción	DHT11	DHT22
Rango de medición de temperatura	0 a 50°C	-40 a 80°C
Precisión de medición de temperatura	±1 a ±2°C	±0.5°C
Resolución o sensibilidad en temperatura	1°C	0.1°C
Rango de medición de humedad	20 a 80%	0 a 100%
Precisión de medición de humedad	4 a 5%	2 a 5%
Resolución o sensibilidad en humedad	1%	0.1%
Frecuencia de muestras (muestras por segundo)	1 Hz	2 Hz

Tabla 11: Diferencias entre los sensores DHT11 y DHT22.

Las terminales en ambos sensores están situadas de la misma forma, como se puede ver a continuación:

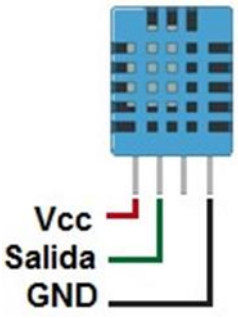


Ilustración 26: Partes del sensor DHT11.

- **Vcc**: Voltaje de alimentación. DHT11: 3 a 5.5V. DHT22: 3.3 a 6V.
- **Salida**: Datos. Obviamente se conecta a un pin digital.
- **GND**: Tierra.

Así mismo, para emplearlo dentro del código, es necesario hacer uso de la librería indicada; las librerías son software que ayuda a facilitar el uso de un dispositivo en específico, interpretando también sus datos. Cada dispositivo puede tener su propia librería, o también hay librerías que cubren más de uno, en este caso la librería en cuestión es la siguiente:

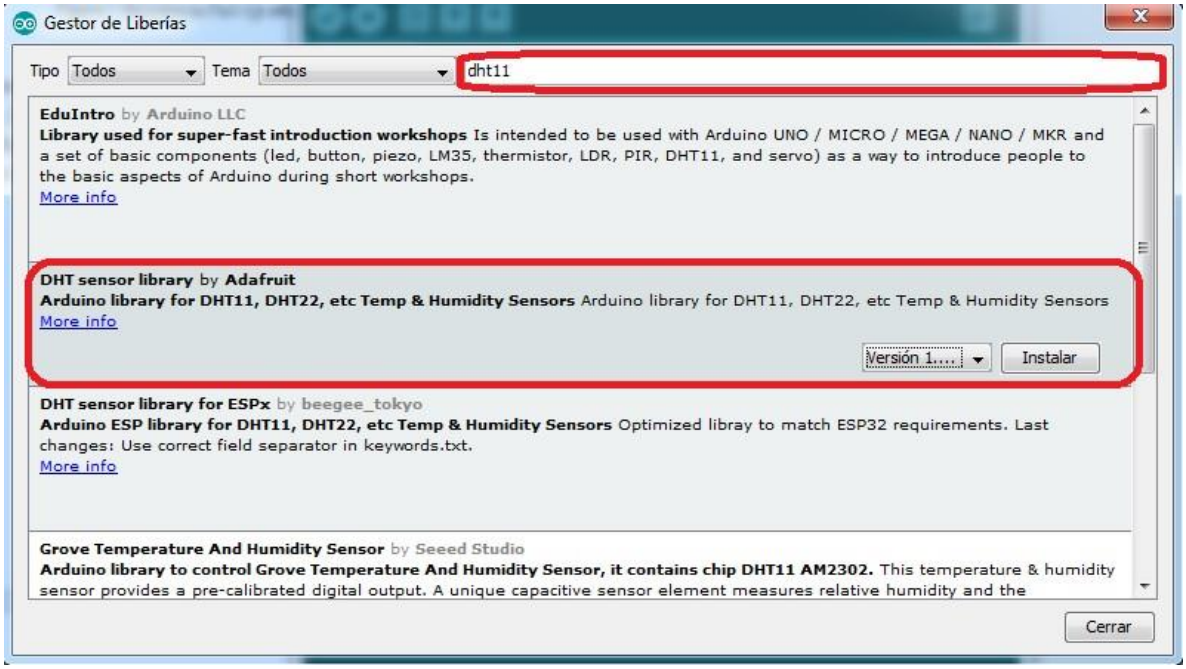


Ilustración 27: Gestor de librerías para el sensor DHT11.

Y para incluir la librería se utiliza la siguiente línea:

```
#include "DHT.h"
```

Algunos comandos que se emplean con este sensor son los siguientes: Primero, se recomienda definir el pin a usarse en una variable, por ejemplo, si se usará el GPIO 12 (D6), se escribe como sigue.

```
#define pinDHT 12
```

También se define el tipo de sensor DHT que se usará. Si se usa el DHT22.

```
#define tipoDHT DHT22
```

Y si se usa el DHT11.

```
#define tipoDHT DHT11
```

La siguiente línea inicializa el sensor con el pin y el tipo de sensor definidos.

```
DHT dht(pinDHT, tipoDHT);
```

Todas las líneas anteriores se escriben antes de la función *setup*.

La siguiente línea, se escribe en *setup*, empezando con la comunicación con el sensor.

```
dht.begin();
```

Las siguientes líneas se usan para obtener los datos de temperatura o la humedad. Devuelven un dato numérico con fracciones, así que, si se desea guardar dicho dato, debe ser en variables *double* o *float*.

```
dht.readHumidity();
dht.readTemperature();
dht.readTemperature(true);
```

La segunda línea para obtener la temperatura, da el valor en grados Celsius. Mientras que la línea que incluye el argumento *true*, da la temperatura en grados Fahrenheit.

Calcular el índice de calor. El resultado es en grados Fahrenheit.

```
dht.computeHeatIndex(F, H);
```

F y **H** son variables que contienen los valores de la temperatura en grados Fahrenheit y de la humedad, respectivamente.

Una vez que sabemos eso, se puede pasar al código que se empleará para la lectura de las variables:

```
#include <DHT.h>

#define pinDHT 0
#define tipoDHT DHT22
DHT dht(pinDHT, tipoDHT, 11);

void setup()
{
  Serial.begin(115200);
  dht.begin();
  Serial.println();
  Serial.println("Iniciando sensor DHT11 o DHT22");
}

void loop()
{
  delay(2000);
  float H = dht.readHumidity();
  float Tc = dht.readTemperature();
  float Tf = dht.readTemperature(true);

  Serial.print("Humedad: ");
  Serial.print(H); Serial.println("%");

  Serial.print("Temperatura Celsius: ");
  Serial.print(Tc); Serial.println("*C");

  Serial.print("Temperatura Fahrenheit: ");
  Serial.print(Tf); Serial.println("*F");
}
```

Ilustración 28: Código para el funcionamiento del sensor DHT22.

En primer lugar, se llama a la biblioteca que se usará para el sensor. Después se declara cual pin será usado para leer los datos, en este caso será el pin 0 (D3). En la siguiente línea se escribe cuál sensor será usado. Si está usando el DHT11, reemplácelo en el código.

Luego, se crea un objeto para el sensor, y toma como argumentos el pin, el tipo de sensor antes definidos y un número (para el NodeMCU siempre usar el 11).

En setup, se establecen los baudios, se inicia el objeto del sensor y se imprime una frase para indicar que el sensor está listo.

En loop, primero se espera 2 segundos, antes de pedir datos al sensor, la razón es por la frecuencia a la que obtienen las mediciones, son sensores muy lentos. Posteriormente, las siguientes 3 líneas son para adquirir los valores de: Humedad, temperatura en grados Celsius y temperatura en grados Fahrenheit. Y son almacenados los datos en las variables declaradas en esa misma línea.

Después, se van mostrando en el monitor serial cada una de las mediciones obtenidas. Se usan 3 líneas para cada medición

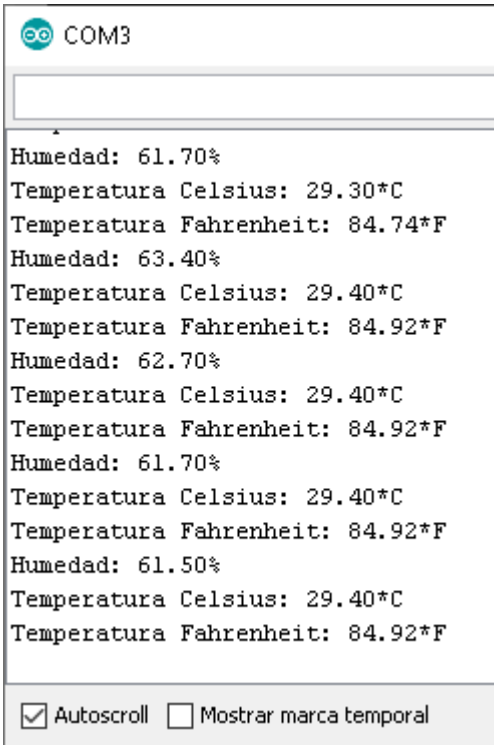


Ilustración 29: Captura de las variables del sensor DHT22.

3.1.3.6 Pruebas de comunicación del protocolo I2C para la comunicación del sensor de luminosidad y la pantalla LCD

El microcontrolador ESP8266 puede realizar la comunicación a través del protocolo I2C, ya sea para comunicarse con otros microcontroladores que lo manejen, o para obtener información de sensores que también lo usen.

Los GPIO destinados para esta comunicación, son el 4 como SDA, y el 5 como SCL. Sin embargo, se pueden usar otros pines según como se configure en la programación. La librería que se usa para incluir este protocolo de comunicación, es *Wire*, y se incluye como se muestra a continuación:

```
#include <Wire.h>
```

Ahora, el código que emplearemos para realizar la prueba de comunicación será la siguiente:

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_TSL2561_U.h>

Adafruit_TSL2561_Unified TSL = Adafruit_TSL2561_Unified(TSL2561_ADDR_FLOAT, 12345);

float luz;

void setup()
{
  Serial.begin(115200);
  Serial.println();
  Serial.println("Medición de luminosidad");
}

void loop()
{
  sensors_event_t event;
  TSL.getEvent(&event);
  luz = event.light;
  Serial.print(luz);
  Serial.println(" lux");

  delay(500);
}
```

Ilustración 30: Código para pruebas del sensor de luminosidad.

Primero se declaran las 3 librerías que serán usadas.

Después, se crea un objeto para el sensor, y con 2 argumentos se le asignan la dirección para el protocolo I2C y una identificación única. Posteriormente se declara una variable float para almacenar la medición.

En loop, se necesitan 2 líneas para preparar el objeto del que se obtendrá el dato de la medición, en la tercera línea ese dato se guarda en la variable para después ser mostrada en el Monitor Serial. Este proceso se realizará cada medio segundo.

Recuerde que en la comunicación I^2C, por defecto se usan los pines 4 y 5, para SDA y SCL respectivamente.

Anteriormente, se empleó el código para realizar pruebas de la comunicación del sensor y a continuación se hará uso del protocolo I2C para la comunicación con la pantalla LCD, probando cada uno por separado para posteriormente crear uno capaz de combinar ambos y juntar sus cualidades y funcionamientos.

En esta ocasión, el código empleado es el siguiente:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C LCD(0x27, 16, 2);

void setup()
{
  LCD.begin();
  LCD.backlight();
}

void loop()
{
  LCD.setCursor(0,0);
  LCD.print("Hola ");
  LCD.print("mundo");
  delay(1500);

  LCD.clear();
  LCD.setCursor(8,1);
  LCD.print(537);
  delay(1500);

  LCD.clear();
}
```

Ilustración 31: Código para prueba de la pantalla LCD.

Primero, se llaman a las librerías necesarias. Después, se construye un objeto para la pantalla LCD con el módulo I2C, especificando primero su dirección, y después el tamaño de la pantalla LCD, escribiendo primero las columnas y después las filas (en este caso, un LCD de 16x2).

En setup se inicia el objeto, y se enciende la luz del LCD.

En loop, primero se coloca el cursor en la posición 0,0, lo que equivale a colocarlo justo al principio del LCD, y desde ahí escribe primero la palabra “Hola”, después escribe la palabra “mundo” y espera un segundo y medio.

Después, limpia el texto del LCD y coloca el cursor en la posición (8,1), lo que significa que estará a la mitad de la segunda fila, y ahí escribirá el número 537, luego espera un segundo y medio y limpia la pantalla antes de repetir el ciclo loop.

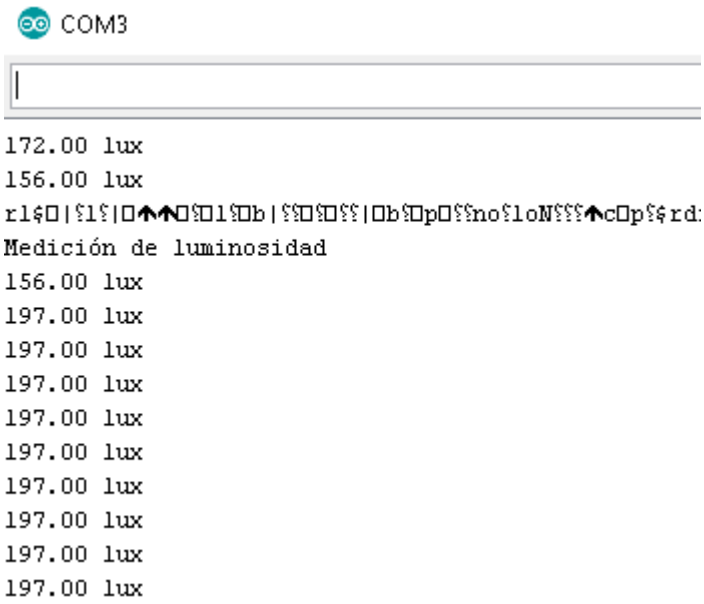


Ilustración 32: Captura de variables del sensor de luminosidad.

3.1.3.7 Creación y configuración de un punto de acceso para la configuración de los dispositivos IoT

El microcontrolador ESP8266 es capaz de generar y configurar un punto de acceso en el cual se pueden conectar diversos dispositivos, en esta ocasión, se hará uso de la pantalla LCD, para mostrar la dirección IP que se utilizará.

El código que se empleará será el siguiente, así mismo, se harán uso de diversas librerías, como wire.h, LiquidCrystal_I2C y el ESP8266Wifi.h.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <ESP8266WiFi.h>

LiquidCrystal_I2C LCD(0x27, 16, 2);

const char *ssid = "NodeMCU-AP";
const char *password = "12345678";

void setup()
{
  WiFi.mode(WIFI_OFF);
  WiFi.mode(WIFI_AP);
  delay(50);

  LCD.begin();
  LCD.backlight();
  LCD.setCursor(0,0);
  if(WiFi.softAP(ssid, password);)
  {
    LCD.print(ssid);
    LCD.setCursor(0,1);
    LCD.print(WiFi.softAPIP());
  }
  else
  {
    LCD.print("ERROR");
  }
}

void loop()
{

}
```

Ilustración 33: Código para la creación de un punto de acceso.

primero se establecen el nombre y la contraseña del Punto de Acceso, la contraseña es opcional.

Ahora, el modo WiFi se establece como Punto de Acceso, y se prepara la pantalla LCD.

Se construye el Punto de Acceso pasando el nombre y la contraseña de la red, pero dicho comando se escribe en la condición de un *if*. Esto es porque, cuando se construye un Punto de Acceso, tal comando devuelve un valor de **true** si fue construido exitosamente, y en caso contrario será un **false**; entonces, si se logra construir, la pantalla LCD imprimirá el nombre y la IP del Punto de Acceso, si no, imprimirá la palabra **ERROR**.

Una vez que se haya cargado el código, nos podremos conectar al punto de acceso que se acaba de crear.

Ilustración 34: Conexión al punto de acceso creado.

Para comprobar que se haya realizado la conexión, podemos entrar al símbolo del sistema y con el comando ifconfig, comprobar la dirección IP que se nos asigna y de la puerta de enlace.

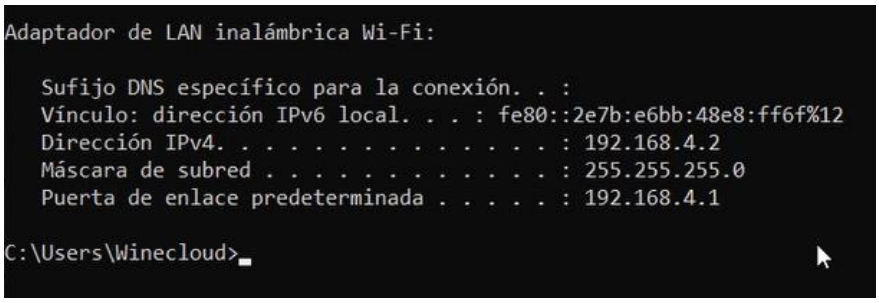


Ilustración 35: Verificación la dirección IP asignada en por el punto de acceso.

3.1.3.8 Implementación y configuración de un servidor web para pruebas del muestreo de información

Un servidor web, se caracteriza por recibir y responder peticiones HTTP almacenando y gestionando archivos, al emplear un microcontrolador ESP8266 la diferencia radica en que el cliente es quien empieza la comunicación, enviando solicitudes HTTP al Servidor, luego el servidor responderá de acuerdo a como se encuentre programado, para responder a una petición HTTP.Programar como servidor web al NodeMCU no requiere de un modo WiFi en específico, es decir, puede ser un servidor web como Punto de Acceso o como Estación, o ambos a la vez; sin embargo, se ha comprobado que en algunos dispositivos WiFi falla la conexión cuando se usa el Node exclusivamente como Punto de Acceso.

En este caso, se construirá el Servidor web mientras el Node construye su propia red, usando el modo WiFi Punto de Acceso y Estación a la vez. El código se muestra a continuación.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <LiquidCrystal_I2C.h>

const char *ssid = "NodeMCU-AP";
const char *password = "12345678";

ESP8266WebServer server(80);
LiquidCrystal_I2C LCD(0x27, 16, 2);

void setup()
{
  WiFi.mode(WIFI_OFF);
  WiFi.mode(WIFI_AP_STA);
  delay(20);

  LCD.begin();
  LCD.backlight();
  LCD.setCursor(0,0);
  if(WiFi.softAP(ssid, password))
  {
    LCD.print(ssid);
    LCD.setCursor(0,1);
    LCD.print(WiFi.softAPIP());
  }
  else
  {
    LCD.print("ERROR");
  }

  server.on("/", bienvenida);
  server.onNotFound(NotFound);
  server.begin();
}

void bienvenida()
{
  server.send(200, "text/plain", "Hola desde el NodeMCU");
}

void NotFound()
{
  server.send(404, "text/plain", "404: Not Found");
}

void loop()
{
  server.handleClient();
}
```

Ilustración 36: Código para la creación de un servidor y página web.

Una vez que se haya cargado el código en el ESP8266 nos conectaremos al punto de acceso que se creó, y posteriormente, en el navegador escribiremos la dirección IP de la puerta de enlace, que en este caso es 192.168.4.1 como se puede ver a continuación.

Ilustración 37: Conexión con la página web creada.

De esta manera, se puede comprobar que el prototipo del servidor web funciona de una manera adecuada, logrando modificar posteriormente el código será posible solicitar la información solicitada a través de los sensores.

3.1.3.9 Diseño y programación de las APIs de lectura

Las API permiten que los productos y servicios se comuniquen con otros sin necesidad de saber cómo están implementados, estas otorgan flexibilidad; ya que simplifican el diseño, la administración y el uso de las aplicaciones, ofreciendo oportunidades de innovación.

En esta cuestión, enfocado al proyecto se realizaron dos códigos que fueran capaz de realizar la lectura de los dispositivos, en el primero se puede solicitar temperatura y humedad, mientras que en el segundo se pueden realizar diversas funciones.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>

#define pinDHT 0
#define tipoDHT DHT11

const char *ssid = "NodeMCU-AP";
const char *password = "12345678";

ESP8266WebServer server(80);
LiquidCrystal_I2C LCD(0x27, 16, 2);
DHT dht(pinDHT, tipoDHT, 11);

float H, Tc, Tf;

void setup()
{
  WiFi.mode(WIFI_OFF);
  WiFi.mode(WIFI_AP_STA);
  delay(20);

  dht.begin();
  LCD.begin();
  LCD.backlight();
  LCD.setCursor(0,0);
  bool AP = WiFi.softAP(ssid, password);
  server.on("/", bienvenida);
  server.on("/hume", humedad);
  server.on("/temc", temperaturaC);
  server.on("/temf", temperaturaF);
  server.onNotFound(NotFound);
  server.begin();
  delay(1000);

  if(AP == true)
  {
    LCD.print(WiFi.softAPIP());
  }
  else
  {
    LCD.print("ERROR");
  }
}
```

Ilustración 38: Desarrollo del código de la primera API de lectura.

```
void bienvenida()
{
  server.send(200, "text/plain", "Hola desde el NodeMCU");
}

void humedad()
{
  H = dht.readHumidity();
  LCD.clear();
  LCD.setCursor(0,0);
  LCD.print(WiFi.softAPIP());
  LCD.setCursor(0,1);
  LCD.print("H: "); LCD.print(H); LCD.print("%");
  String cont = "Humedad: " + String(H) + "%";
  server.send(200, "text/plain", cont);
}

void temperaturaC()
{
  Tc = dht.readTemperature();
  LCD.clear();
  LCD.setCursor(0,0);
  LCD.print(WiFi.softAPIP());
  LCD.setCursor(0,1);
  LCD.print("T: "); LCD.print(Tc); LCD.print("*C");
  String cont = "Temperatura: " + String(Tc) + "*C";
  server.send(200, "text/plain", cont);
}

void temperaturaF()
{
  Tf = dht.readTemperature(true);
  LCD.clear();
  LCD.setCursor(0,0);
  LCD.print(WiFi.softAPIP());
  LCD.setCursor(0,1);
  LCD.print("T: "); LCD.print(Tf); LCD.print("*F");
  String cont = "Temperatura: " + String(Tf) + "*F";
  server.send(200, "text/plain", cont);
}

void NotFound()
{
  server.send(404, "text/plain", "404: Not Found");
}
```

Ilustración 39: Desarrollo del código de la primera API de lectura, parte 2.

El segundo código empleado para realizar peticiones es el siguiente;

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <LiquidCrystal_I2C.h>
#include <Servo.h>
#include <Wire.h>
#include <DHT.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_TSL2561_U.h>

#define pinDHT 0
#define tipoDHT DHT11
const char *ssid = "NodeMCU-AP";
const char *password = "12345678";
const int foco = 14;
const int trig = 16;
const int echo = 10;

ESP8266WebServer server(80);
LiquidCrystal_I2C LCD(0x27, 16, 2);
DHT dht(pinDHT, tipoDHT, 11);
Adafruit_TSL2561_Unified TSL = Adafruit_TSL2561_Unified(TSL2561_ADDR_FLOAT, 12345);
Servo myServo1;
Servo myServo2;
Servo myServo3;
Servo myServo4;

int cm, An, H, Tc, Tf, L;
```

Ilustración 40: Desarrollo del código de la segunda API de lectura.

```
void bienvenida()
{
  server.send(200, "text/plain", "Hola desde el NodeMCU");
}

void paginaWeb()
{
  server.send(200, "text/html", construirPag());
}

String construirPag()
{
  String html = "<HTML>"
                "<HEAD> <TITLE> Tarjeta Integradora Mecatronica </TITLE> </HEAD>"
                "<BODY>"
                "<FONT> TARJETA INTEGRADORA MECATRONICA</FONT><BR><BR>"
                "<A HREF='/leer'><BUTTON> Analogico </BUTTON></A><BR><BR>"
                "<A HREF='/ON'><BUTTON> Rele ON </BUTTON></A>"
                "<A HREF='/OFF'><BUTTON> Rele OFF </BUTTON></A><BR><BR>"
                "<A HREF='/varDHT'><BUTTON> Valores DHT </BUTTON></A><BR><BR>"
                "<A HREF='/dist'><BUTTON> Distancia </BUTTON></A><BR><BR>"
                "<A HREF='/luz'><BUTTON> Luminosidad </BUTTON></A><BR><BR>"
                "<FORM action='/servo1' method='GET'>"
                "<LABEL for='sm1'> Servomotor 1: </LABEL><BR>"
                "<INPUT type='text' id='sm1' name='ang1' />"
                "<INPUT type='submit' value='Enviar' /><BR><BR>"
                "</FORM>"
                "<FORM action='/servo2' method='GET'>"
                "<LABEL for='sm2'> Servomotor 2: </LABEL><BR>"
                "<INPUT type='text' id='sm2' name='ang2' />"
                "<INPUT type='submit' value='Enviar' />"
                "</FORM>"
                "<FORM action='/servo3' method='GET'>"
                "<LABEL for='sm3'> Servomotor 3: </LABEL><BR>"
                "<INPUT type='text' id='sm3' name='ang3' />"
                "<INPUT type='submit' value='Enviar' /><BR><BR>"
                "</FORM>"
                "<FORM action='/servo4' method='GET'>"
                "<LABEL for='sm4'> Servomotor 4: </LABEL><BR>"
                "<INPUT type='text' id='sm4' name='ang4' />"
                "<INPUT type='submit' value='Enviar' />"
                "</FORM>"
                "</BODY>"
                "</HTML>";
  return html;
}
```

Ilustración 41: Desarrollo del código de la segunda API de lectura, parte 2.

```
void setup()
{
  pinMode(foco, OUTPUT);
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  digitalWrite(foco, LOW);
  digitalWrite(trig, LOW);
  dht.begin();
  myServo1.attach(2);
  myServo2.attach(12);
  myServo3.attach(13);
  myServo4.attach(15);
  delay(20);

  WiFi.mode(WIFI_OFF);
  WiFi.mode(WIFI_AP_STA);
  bool AP = WiFi.softAP(ssid, password);
  delay(20);

  server.on("/", bienvenida);
  server.on("/menu", paginaWeb);
  server.on("/leer", analogico);
  server.on("/ON", releON);
  server.on("/OFF", releOFF);
  server.on("/varDHT", sensorDHT);
  server.on("/dist", sensorHCSR04);
  server.on("/luz", sensorTSL2561);
  server.on("/servo1", servo_1);
  server.on("/servo2", servo_2);
  server.on("/servo3", servo_3);
  server.on("/servo4", servo_4);
  server.onNotFound(NotFound);
  server.begin();
  delay(20);

  LCD.begin();
  LCD.backlight();
  LCD.setCursor(0,0);
  if(AP == true){LCD.print(WiFi.softAPIP());}
  else{LCD.print("ERROR");}
  delay(2000);
}
```

Ilustración 42: Desarrollo del código de la segunda API de lectura, parte 3.

```
void analogico()
{
  String pA = String(An * (3.3/1023)) + "V";
  server.send(200, "text/plain", pA);
}

void releON()
{
  digitalWrite(foco, HIGH);
  server.send(200, "text/plain", "Foco encendido");
}

void releOFF()
{
  digitalWrite(foco, LOW);
  server.send(200, "text/plain", "Foco apagado");
}

void sensorDHT()
{
  String datos = "Humedad: " + String(H) + " %\n"
                 "Temperatura: " + String(Tc) + " *C\n"
                 "             " + String(Tf) + " *F";
  server.send(200, "text/plain", datos);
}

void sensorHCSR04()
{
  String med = String(cm) + " cm";
  server.send(200, "text/plain", med);
}

void sensorTSL2561()
{
  String luz = String(L) + " lux";
  server.send(200, "text/plain", luz);
}
```

Ilustración 43: Desarrollo del código de la segunda API de lectura, parte 4.

```
void desplegadoLCD()
{
  digitalWrite(trig, LOW); delayMicroseconds(10);
  digitalWrite(trig, HIGH); delayMicroseconds(10);
  digitalWrite(trig, LOW);
  long esp = (pulseIn(echo, HIGH)/2);
  cm = int(esp * 0.0343);
  An = analogRead(A0);
  H = dht.readHumidity(); Tc = dht.readTemperature(); Tf = dht.readTemperature(true);
  sensors_event_t event; TSL.getEvent(&event); L = event.light;
  bool F = digitalRead(foco);
  String Fc;
  if(F){Fc = "On";}
  else{Fc = "Off";}

  LCD.clear();
  if(An <= 566)
  {
    LCD.setCursor(0,0);
    LCD.print("Foco:"); LCD.print(Fc);LCD.print(" D:");
    LCD.print(cm); LCD.print("cm");
    LCD.setCursor(0,1);
    LCD.print("Luz:"); LCD.print(L); LCD.print("Lux");
  }
  else if (An > 566)
  {
    LCD.setCursor(0,0);
    LCD.print("DHT: "); LCD.print(H); LCD.print("%");
    LCD.setCursor(0,1);
    LCD.print(Tc); LCD.print("*C  ");
    LCD.print(Tf); LCD.print("*F");
  }
}
```

Ilustración 44: Desarrollo del código de la segunda API de lectura, parte 5.

```
void servo_1()
{
  servomotores(1);
}

void servo_2()
{
  servomotores(2);
}

void servo_3()
{
  servomotores(3);
}

void servo_4()
{
  servomotores(4);
}

void servomotores(int Nserv)
{
  String a = server.arg(0);
  int ang = a.toInt();
  if(ang >= 0 && ang <= 180)
  {
    if(Nserv == 1){myServo1.write(ang);}
    else if(Nserv == 2){myServo2.write(ang);}
    else if(Nserv == 3){myServo3.write(ang);}
    else if(Nserv == 4){myServo4.write(ang);}
    server.send(200, "text/plain", ("Angulo enviado: " + a));
  }
  else{server.send(200, "text/plain", "Angulo fuera de rango");}
}

void NotFound()
{
  server.send(404, "text/plain", "404: Not Found");
}

void loop()
{
  desplegadoLCD();
  server.handleClient();
  delay(50);
}
```

Ilustración 45: Desarrollo del código de la segunda API de lectura, parte 6.

Por último, se integraron ambos códigos para realizar uno solo que fuera capaz de ejecutar todas las peticiones sin necesidad de cambiarlo y subirlo al microcontrolador ESP-8266, el resultado fue el siguiente:

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <LiquidCrystal_I2C.h>
#include <Servo.h>
#include <Wire.h>
#include <DHT.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_TSL2561_U.h>

#define pinDHT 0
#define tipoDHT DHT11
const char *ssid = "NodeMCU-AP";
const char *password = "12345678";
const int foco = 14;
const int trig = 16;
const int echo = 10;

ESP8266WebServer server(80);
LiquidCrystal_I2C LCD(0x27, 16, 2);
DHT dht(pinDHT, tipoDHT, 11);
Adafruit_TSL2561_Unified TSL = Adafruit_TSL2561_Unified(TSL2561_ADDR_FLOAT, 12345);
Servo myServo1;
Servo myServo2;
Servo myServo3;
Servo myServo4;

int cm, An, H, Tc, Tf, L;
```

Ilustración 46: Integración de ambos códigos.

```
void setup()
{
  pinMode(foco, OUTPUT);
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  digitalWrite(foco, LOW);
  digitalWrite(trig, LOW);
  dht.begin();
  myServo1.attach(2);
  myServo2.attach(12);
  myServo3.attach(13);
  myServo4.attach(15);
  delay(20);

  WiFi.mode(WIFI_OFF);
  WiFi.mode(WIFI_AP_STA);
  bool AP = WiFi.softAP(ssid, password);
  delay(20);

  server.on("/", bienvenida);
  server.on("/menu", paginaWeb);
  server.on("/leer", analogico);
  server.on("/ON", releON);
  server.on("/OFF", releOFF);
  server.on("/varDHT", sensorDHT);
  server.on("/dist", sensorHCSR04);
  server.on("/luz", sensorTSL2561);
  server.on("/servo1", servo_1);
  server.on("/servo2", servo_2);
  server.on("/servo3", servo_3);
  server.on("/servo4", servo_4);
  server.onNotFound(NotFound);
  server.begin();
  delay(20);

  LCD.begin();
  LCD.backlight();
  LCD.setCursor(0,0);
  if(AP == true){LCD.print(WiFi.softAPIP());}
  else{LCD.print("ERROR");}
  delay(2000);
}
```

Ilustración 47: Integración de ambos códigos, parte 2.

```
void bienvenida()
{
  server.send(200, "text/plain", "Hola desde el NodeMCU");
}

void paginaWeb()
{
  server.send(200, "text/html", construirPag());
}

String construirPag()
{
  String html = "<HTML>"
                "<HEAD> <TITLE> Tarjeta Integradora Mecatronica </TITLE> </HEAD>"
                "<BODY>"
                "<FONT> TARJETA INTEGRADORA MECATRONICA</FONT><BR><BR>"
                "<A HREF='/leer'><BUTTON> Analogico </BUTTON></A><BR><BR>"
                "<A HREF='/ON'><BUTTON> Rele ON </BUTTON></A>"
                "<A HREF='/OFF'><BUTTON> Rele OFF </BUTTON></A><BR><BR>"
                "<A HREF='/varDHT'><BUTTON> Valores DHT </BUTTON></A><BR><BR>"
                "<A HREF='/dist'><BUTTON> Distancia </BUTTON></A><BR><BR>"
                "<A HREF='/luz'><BUTTON> Luminosidad </BUTTON></A><BR><BR>"
                "<FORM action='/servo1' method='GET'>"
                "<LABEL for='sm1'> Servomotor 1: </LABEL><BR>"
                "<INPUT type='text' id='sm1' name='ang1' />"
                "<INPUT type='submit' value='Enviar' /><BR><BR>"
                "</FORM>"
                "<FORM action='/servo2' method='GET'>"
                "<LABEL for='sm2'> Servomotor 2: </LABEL><BR>"
                "<INPUT type='text' id='sm2' name='ang2' />"
                "<INPUT type='submit' value='Enviar' />"
                "</FORM>"
                "<FORM action='/servo3' method='GET'>"
                "<LABEL for='sm3'> Servomotor 3: </LABEL><BR>"
                "<INPUT type='text' id='sm3' name='ang3' />"
                "<INPUT type='submit' value='Enviar' /><BR><BR>"
                "</FORM>"
                "<FORM action='/servo4' method='GET'>"
                "<LABEL for='sm4'> Servomotor 4: </LABEL><BR>"
                "<INPUT type='text' id='sm4' name='ang4' />"
                "<INPUT type='submit' value='Enviar' />"
                "</FORM>"
                "</BODY>"
                "</HTML>";
  return html;
}
```

Ilustración 48: Integración de ambos códigos, parte 3.

```
void desplegadoLCD()
{
  digitalWrite(trig, LOW); delayMicroseconds(10);
  digitalWrite(trig, HIGH); delayMicroseconds(10);
  digitalWrite(trig, LOW);
  long esp = (pulseIn(echo, HIGH)/2);
  cm = int(esp * 0.0343);
  An = analogRead(A0);
  H = dht.readHumidity(); Tc = dht.readTemperature(); Tf = dht.readTemperature(true);
  sensors_event_t event; TSL.getEvent(&event); L = event.light;
  bool F = digitalRead(foco);
  String Fc;
  if(F){Fc = "On";}
  else{Fc = "Off";}

  LCD.clear();
  if(An <= 566)
  {
    LCD.setCursor(0,0);
    LCD.print("Foco:"); LCD.print(Fc);LCD.print(" D:");
    LCD.print(cm); LCD.print("cm");
    LCD.setCursor(0,1);
    LCD.print("Luz:"); LCD.print(L); LCD.print("Lux");
  }
  else if (An > 566)
  {
    LCD.setCursor(0,0);
    LCD.print("DHT: "); LCD.print(H); LCD.print("%");
    LCD.setCursor(0,1);
    LCD.print(Tc); LCD.print("*C  ");
    LCD.print(Tf); LCD.print("*F");
  }
}
```

Ilustración 49: Integración de ambos códigos, parte 4.

```
void analogico()
{
  String pA = String(An * (3.3/1023)) + "V";
  server.send(200, "text/plain", pA);
}

void releON()
{
  digitalWrite(foco, HIGH);
  server.send(200, "text/plain", "Foco encendido");
}

void releOFF()
{
  digitalWrite(foco, LOW);
  server.send(200, "text/plain", "Foco apagado");
}

void sensorDHT()
{
  String datos = "Humedad: " + String(H) + " %\n"
                 "Temperatura: " + String(Tc) + " *C\n"
                 "             " + String(Tf) + " *F";
  server.send(200, "text/plain", datos);
}

void sensorHCSR04()
{
  String med = String(cm) + " cm";
  server.send(200, "text/plain", med);
}

void sensorTSL2561()
{
  String luz = String(L) + " lux";
  server.send(200, "text/plain", luz);
}
```

Ilustración 50: Integración de ambos códigos, parte 5.

```
void servo_1()
{
  servomotores(1);
}

void servo_2()
{
  servomotores(2);
}

void servo_3()
{
  servomotores(3);
}

void servo_4()
{
  servomotores(4);
}

void servomotores(int Nserv)
{
  String a = server.arg(0);
  int ang = a.toInt();
  if(ang >= 0 && ang <= 180)
  {
    if(Nserv == 1){myServo1.write(ang);}
    else if(Nserv == 2){myServo2.write(ang);}
    else if(Nserv == 3){myServo3.write(ang);}
    else if(Nserv == 4){myServo4.write(ang);}
    server.send(200, "text/plain", ("Angulo enviado: " + a));
  }
  else{server.send(200, "text/plain", "Angulo fuera de rango");}
}

void NotFound()
{
  server.send(404, "text/plain", "404: Not Found");
}

void loop()
{
  desplegadoLCD();
  server.handleClient();
  delay(50);
}
```

Ilustración 51: Integración de ambos códigos, parte 6.

3.1.3.10 Pruebas de la integración de las APIs con los sensores seleccionados

En el funcionamiento de este primer código se tienen las bibliotecas, seguidas del pin a usarse para el DHT y que tipo de DHT se usará. Se definen el nombre y contraseña para el Punto de Acceso y se crean los objetos para el servidor, la pantalla LCD y el sensor DHT, después se declaran 3 variables float.

En setup se establece el modo WiFi y se inician el sensor DHT y el LCD, además se construye el Punto de Acceso, pero en esta ocasión, su valor se guarda en una variable tipo bool. Seguido de eso, se crean las URIs para el servidor, y se han añadido 3 URIs más que en la práctica anterior, siendo cada una para los 3 diferentes datos que se pedirán del sensor DHT. Después se inicia el servidor, y espera 1 segundo antes de avisar por la pantalla LCD si el Punto de Acceso está listo.

Luego están las funciones que corresponden a cada URI. Las 3 funciones añadidas tienen la misma estructura: Solicitan el dato y lo guardan en la variable, limpian la pantalla y vuelven a imprimir la IP y también el dato solicitado por HTTP, usando 3 líneas para imprimir una letra (H para humedad y T para temperatura), la cifra, y el tipo de medición; aunque están escritas sobre la misma línea, esto no importa, pueden estar escritas de arriba hacia abajo como las demás.

También se construye un dato de tipo String, y por último se envía al cliente.

De acuerdo a las URIs construidas, las solicitudes HTTP que se deben enviar al servidor son las siguientes:

- HTTP://192.168.4.1/ ó 192.168.4.1: URI de bienvenida.
- HTTP://192.168.4.1/hume ó 192.168.4.1/hume: Solicitar humedad.
- 192.168.4.1/temc: Solicitar temperatura en Celsius.
- 192.168.4.1/temf: Solicitar temperatura en Farenheit.

En el segundo código, primero se incluyen las librerías, se definen pines, definen características de la red, se crean objetos para los dispositivos y crear variables que contengan la información de los dispositivos. Aquí hay 6 variables de tipo *int*, y la razón por la que son globales (es decir, están declaradas en una posición donde pueden ser usadas en cualquier parte del código) es porque se tendrán diferentes funciones que harán uso de los mismos datos. No olvide cambiar el tipo de sensor DHT si es necesario.

En la función *setup*, es muy similar al primer código, realmente no hay algo nuevo solo que la URI antes llamada **/botones** para la página Web, ahora se llama **/menu**.

Posteriormente, contiene las funciones de bienvenida (URI */*) y paginaWeb (URI ***/menu***), así como la función encargada de contener el código HTML, esta vez más grande debido a que se incluyen todos los dispositivos, y se construirán 4 conjuntos de **etiqueta, entrada de texto** y **botón**, uno para cada servomotor.

Después, contiene una función llamada ***desplegadoLCD***. Como lo dice su nombre, se encargará de desplegar la información de los sensores, en la pantalla LCD. Primero hace uso de las líneas adecuadas para cada dispositivo, y va guardando en las variables globales la información.

En el caso de la lectura analógica, sólo obtiene el dato de tipo *int* sin convertirlo todavía a voltaje. También se hace una lectura del pin del Relé (**foco**) para obtener su estado actual. Con eso se ayuda para crear un *String* que diga **On** u **Off**, según el estado del Relé, y mostrarlo en la LCD.

Para mostrar la información, se usa una condición *if else if*, y usando la lectura del pin analógico, mostrará diferente información en la pantalla, es decir, el potenciómetro controlará la información mostrada en la pantalla.

La función ***desplegadoLCD*** se ejecutará desde *loop*, así que se estará ejecutando una y otra vez, y como desde esta función se obtienen los datos, es por eso que se declaran las variables como globales, así solo se usarán tales datos para enviar la información al cliente web.

Luego, contiene las funciones en el siguiente orden: Obtener lectura analógica de voltaje en el potenciómetro, encender relé, apagar relé, obtener datos del sensor DHT, obtener la distancia del sensor HC-SR04 y obtener la luminosidad del sensor TSL2561.

Todas las funciones que obtienes datos de los sensores, hacen uso de las variables globales, ya que la función ***desplegadoLCD*** ya ha obtenido dichos datos, y no tendría caso volver a solicitarlos a los sensores. Si la diferencia de tiempo en *loop* fuese muy grande, entonces si se necesitaría volver a solicitar información desde cada función, pero usted puede notar que *loop* tiene un *delay* de 50 milisegundos (20 ejecuciones de ***desplegadoLCD*** cada segundo).

Finalmente, contiene primero las funciones para enviar el ángulo a los servomotores, pero además muestra una función llamada servomotores, la razón de esta función es porque las 4 funciones para cada servo son exactamente iguales, siendo la única diferencia el servo al que van dirigidas, por esta razón se puede hacer uso de una misma función.

Primero, cada función de los servos utiliza la función servomotores, escribiéndole como argumento un número, según el servomotor al que va dirigido. Ahora, la función recibe un argumento de tipo int, el cual representa al servo de destino; aquí el código es casi igual al de la práctica anterior, excepto porque se incluyen las condiciones if else if, para decidir a qué servomotor va dirigido el ángulo.

Finalmente, se tienen las funciones para URIs no existentes y loop, la cual ahora, primero ejecuta la función desplegadoLCD, después tiene la línea para escuchar las peticiones del cliente y ejecuta estas 2 líneas cada 50 milisegundos.

Ilustración 52: Página del menú de interacción con los dispositivos.

Ilustración 53: Encendido del LED desde la página web.

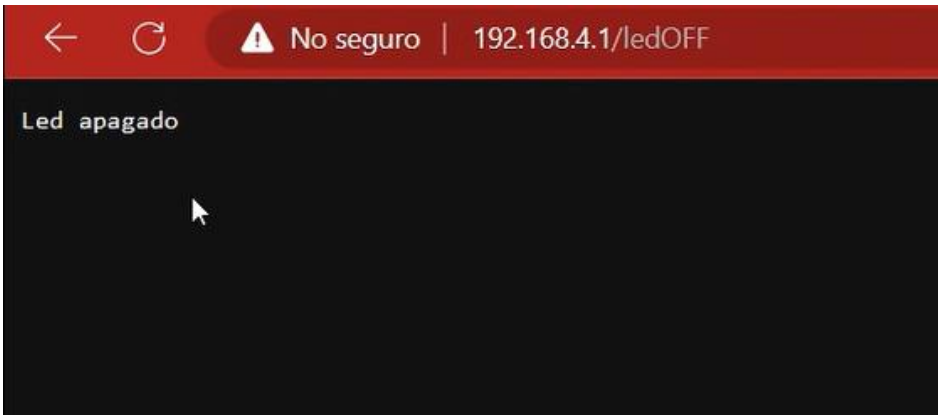


Ilustración 54: Apagado del LED desde la página web.

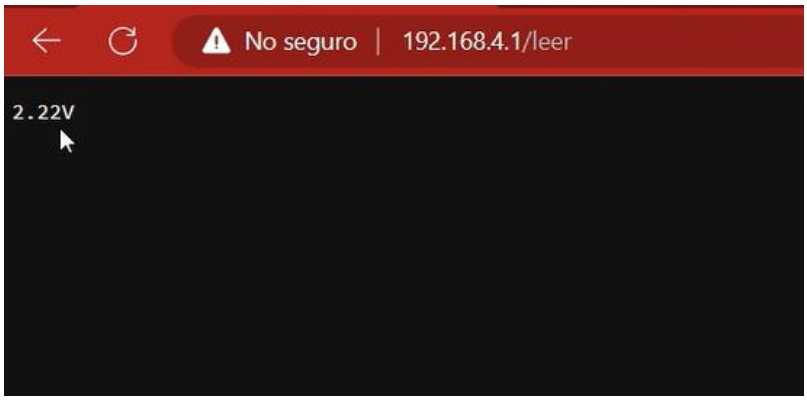


Ilustración 55: Funcionalidad del potenciometro desde la página web.

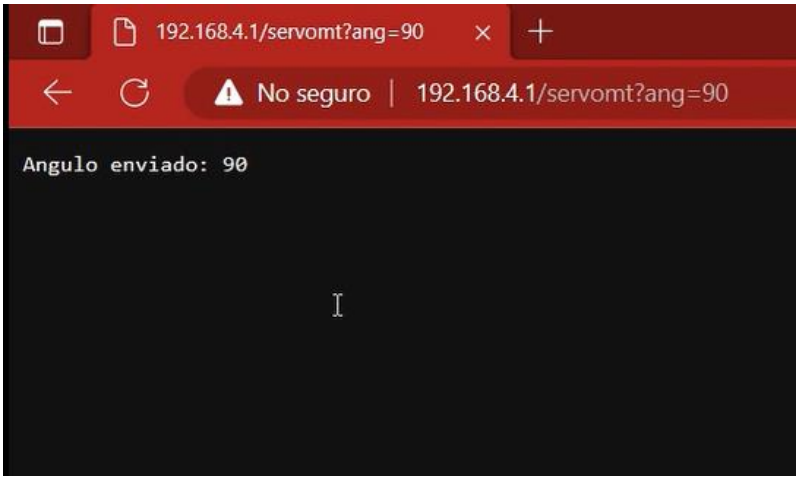


Ilustración 56: Funcionamiento del servomotor desde la página web.

- **3.1.4 Revisión y retrospectiva**

Durante el proceso de creación e implementación, en los códigos se detectaron y corrigieron algunos errores que se fueron presentando, fueron situaciones muy variadas pero que, de igual forma, impedían un funcionamiento adecuado.

Entre ellos, uno de los imprevistos más frecuentes eran las librerías, pues como bien se sabe, las bibliotecas son un conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrecen funciones bien definidas al momento de emplearlas en un código, para que estas se mantengan se tienen que actualizar de una manera constante o recurrente, pero al hacerlo, se van modificando algunos comandos utilizados, en este caso, ese fue uno de las situaciones que se presentaron, pues en la librería de; “LiquidCrystal_I2C” anteriormente y durante mucho tiempo, para el funcionamiento de una pantalla LCD era necesario utilizar la función “LCD.begin” pero al momento de subir el código, se presentaba un error, marcando a la función utilizada como no útil o adecuada.

El método empleado con el que se resolvió el problema fue buscando en la documentación de la librería, en ella se explica que en las versiones anteriores era empleado la función “lcd.begin” pero que esta fue sustituida por “LCD.init”, empleando este nuevo comando, ya se logró subir el código, trabajando con normalidad.

Otro de los problemas que también tuvo un gran impacto fue la placa ESP-8266 que se empleaba, pues esta al intentarla conectar a la computadora no era reconocida, en un principio se creía que el causante era el cable, pues se pensaba que se estaba empleando uno que solo transfería corriente, pero no datos, posteriormente, se comprobó que era la placa la que se encontraba dañada y que por más que se conectara con diferentes cables ninguno iba a funcionar, por lo que se tuvo que cambiar la placa por otra igual.

Ilustración 57: Microcontrolador ESP-8266 dañado.

De esta manera se lograron corregir los errores que se fueron presentando, consiguiendo, seguir adelante con la creación del proyecto.

- **3.1.5 Lanzamiento**

Con los códigos que fueron desarrollados e implementados en las fases anteriores se lograron realizar diferentes actividades, entre ellas, las más destacables fueron las siguientes:

- Controlar sensores como; DHT11 o DHT22, ultrasónico, luminosidad y dispositivos como una pantalla LCD, servomotores, potenciómetros, entre otros.
- Comprender y aplicar el protocolo Inter-Integrated Circuid (I2C) que es un protocolo de comunicación serial empleado para comunicar chips con dispositivos del internet de las cosas.
- La creación de un punto de acceso en el NodeMCU para establecer una conexión inalámbrica entre diferentes dispositivos.
- La configuración de un servidor web empleando el microcontrolador ESP-8266 capaz de generar una página web empleando únicamente el software de IDE de Arduino.
- Y mezclando las anteriores actividades; controlar los dispositivos de IoT y adquirir variables por medio de una página web a través de un punto de acceso para comunicarse con la placa ESP-8266 y esta a su vez, hacerlo a través del protocolo I2C con los dispositivos y sensores.

Con todas las actividades descritas anteriormente, se puede llegar a la conclusión de que la API cumple con su cometido, el cual es; comunicar diferentes aplicaciones y dispositivos entre sí, compartiendo información y funcionalidades, ya que esta se comporta como un intermediario entre dos o más sistemas, permitiendo que alguna aplicación solicite datos o alguna acción en específico, como es en este caso.

Ya que el Usuario está interactuando con la página web, que esta, se comunica con el microcontrolador ESP-8266 a través de una red inalámbrica creada en el NodeMCU gracias a un punto de acceso, y la placa, se conecta con los dispositivos y sensores mediante el protocolo Inter-Circuitos Integrados (I2C) repitiendo el mismo proceso, pero a la inversa para que estos dispositivos envíen una respuesta y que se vea reflejada en la página web.

Capítulo IV: Resultados y conclusiones

4.1 Resultados

Se evaluarán los resultados de acuerdo a los objetivos y alcances esperados que se expusieron anteriormente. Respecto al objetivo general, el cual era: Desarrollar e implementar API's para controlar y adquirir variables por medio de una página web empleando dispositivos de IoT para el envío y la comunicación de datos.

Se logró desarrollar API's utilizando el software IDE de Arduino, subiéndolos al microcontrolador ESP-8266 que genera un servidor y página web en donde se puede solicitar, controlar y adquirir diferentes variables de los dispositivos y sensores de IoT, gracias a los protocolos de comunicación que fueron empleados, los cuales son Wi-Fi e I2C.

Respecto al alcance que se pretendía llegar, era: "El desarrollo de APIs que sean capaces de solicitar, adquirir y controlar variables de los sensores de luminosidad, temperatura, humedad, proximidad y dispositivos como pantalla LCD, potenciómetros, servomotores, entre otros. (componentes de IoT) A través de una página web generada, empleando un microcontrolador System on a Chip que sea capaz de realizar estas funciones."

Realmente, se llegó al alcance esperado, ya que a través de una página web generada en el microcontrolador se pueden solicitar y controlar diversas variables de los diferentes dispositivos empleados, logrando así que las APIs cumplan con la función que fueran hechas.

En conclusión, se cumple con la hipótesis que fue planteada al inicio de este documento el cual decía; "Se espera que con la implementación de las API's se puedan adquirir y controlar las variables que los dispositivos de IoT y el microcontrolador tipo SoC vayan capturando, por medio de una página y servidor web para el envío y la comunicación de los datos recabados."

4.2 Trabajos Futuros

En este apartado, se expondrá un trabajo futuro que se puede llegar a implementar y que daría un plus a este proyecto. El cuál sería la configuración e implementación de una página web más estética, que sea más intuitiva y amigable con el usuario para solicitar los datos de los dispositivos, desarrollada en el NodeMCU que, a su vez, emplea el software IDE de Arduino para su creación.

4.3 Recomendaciones

Con base a los errores expuestos anteriormente, sugiero verificar las versiones de las librerías que van saliendo y realizar una revisión de las que son necesarias para los dispositivos que se emplean en el proyecto, pues algunas sentencias o comandos pueden variar en un futuro y hacer que los códigos que se emplearon para las APIs ya no sean funcionales en versiones futuras, de esta manera, si se presenta algún cambio se podría actualizar el código para retomar un normal funcionamiento.

Bibliografía

[1] Oracle. "¿Qué es el Internet de las cosas (IoT)?" Oracle | Cloud Applications and Cloud Platform. https://www.oracle.com/mx/internet-of-things/what-is-iot/ (accedido el 3 de febrero de 2023).

[2] J. Díaz, P. Venosa, N. Castro, D. Vilches y F. López. SEDICI - Repositorio de la Universidad Nacional de La Plata. http://sedici.unlp.edu.ar/bitstream/handle/10915/62410/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y (accedido el 15 de febrero de 2023).

[3] Intel. "Referencia de chipset y sistema en un chip (SoC) para Intel". Intel. https://www.intel.la/content/www/xl/es/support/articles/000056236/intel-nuc.html (accedido el 26 de febrero de 2023).

[4] A. Birrium. "Diseño e implementación de una solución basada en IoT para la empresa Galeo Enrollables". Academica-e. https://academica-e.unavarra.es/handle/2454/38707 (accedido el 3 de febrero de 2023).

[5] Y. Fernández. "API: Qué es y para qué sirve". Xataka - Tecnología y gadgets, móviles, informática, electrónica. https://www.xataka.com/basics/api-que-sirve (accedido el 15 de febrero de 2023).

[6] D. C. Ledesma Mera, "Reestructuración de la infraestructura de red LAN.", tesis de ingeniería, Universidad Politécnica Salesiana, Guayaquil, 2018. Accedido el 8 de marzo de 2023. [En línea]. Disponible: https://dspace.ups.edu.ec/bitstream/123456789/17336/1/UPS-GT002618.pdf

[7] J. M. Yanangómez Zambrano, "Análisis de seguridad de las redes inalámbricas", Tesis de ingeniería, Universidad Estatal del Sur de Manabí, Jipijapa, 2020. Accedido el 9 de marzo de 2023. [En línea]. Disponible: http://repositorio.unesum.edu.ec/bitstream/53000/2884/1/TESIS%20-%20YANANGOMEZ%20ZAMBRANO%20JAMILETH%20MONSERRATE.pdf

[8] "¿Qué es la tecnología wifi? Definición y tipos". Cisco. https://www.cisco.com/c/es_mx/products/wireless/what-is-wifi.html (accedido el 22 de marzo de 2023).

[9] "¿Qué es una API y cómo funciona?" Red Hat - We make open source technologies for the enterprise. https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces (accedido el 22 de marzo de 2023).