Universidad de Guadalajara

centro universitario de ciencias exactas e ingenierías

Sistema IoT para el encendido remoto de bombas de agua en comunidades rurales

Omar Sánchez Gudiño.

Mtra en computo aplicado | Internet de las cosas.

27 de abril de 2025

Resumen

El presente proyecto propone el diseño e implementación de un sistema basado en IoT para el encendido y apagado remoto de bombas de agua en zonas rurales, utilizando como plataforma principal una ESP32 programada en lenguaje C. En muchas comunidades rurales, el acceso al sistema de bombeo de agua requiere desplazamientos largos a pie, lo cual representa una pérdida considerable de tiempo y esfuerzo, especialmente en actividades relacionadas con la agricultura y el uso doméstico del agua. La solución desarrollada permite a los usuarios activar la bomba desde un dispositivo móvil o plataforma web, eliminando la necesidad de presencia física en el lugar donde se encuentra el equipo. Esto se logra mediante el uso de conectividad inalámbrica, una interfaz sencilla y un sistema de control con relevador, que permite operar la bomba de manera eficiente y segura. Además, se consideraron aspectos de bajo consumo energético y facilidad de instalación para garantizar que el sistema sea viable en contextos con recursos limitados. El proyecto demuestra cómo el uso de tecnologías de bajo costo y fácil acceso puede mejorar significativamente la calidad de vida de los habitantes rurales, optimizando el uso del agua y reduciendo el esfuerzo físico necesario para su gestión. Esta solución no solo representa un avance en términos de automatización, sino también una herramienta que aporta al desarrollo sustentable y a la modernización del campo a través de la tecnología

Índice

1. Introducción.

En las zonas rurales, donde el manejo manual de las bombas de agua implica recorrer distancias considerables y afrontar condiciones adversas, se hace imprescindible una solución accesible, modular y de bajo coste que permita optimizar el abastecimiento hídrico. Este proyecto propone un sistema IoT basado en una placa ESP32, programada íntegramente en C desde el IDE de Arduino, que se integra con la plataforma Blynk para ofrecer un control remoto fiable a través de una aplicación móvil o un panel web. La comunicación Wi-Fi de la ESP32 se encarga de suscribirse a los eventos de Blynk, traduciendo los comandos del usuario en señales de activación digital que conmuten el relé de potencia XN13, capaz de manejar cargas de hasta 10A a 125VAC. Para proporcionar retroalimentación inmediata sobre el estado del sistema, se incorpora un módulo LED RGB XN007 controlado por salidas PWM de la ESP32: verde indica operación correcta, rojo señala un error de comunicación o sobrecarga, y azul refleja el modo de espera o reposo. Cada componente —ESP32, XN13 y XN007— se conecta mediante pines y cajas tipo "ladrillos de Lego", eliminando la necesidad de soldaduras y facilitando el montaje o la expansión en campo. El firmware incluye rutinas de detección de fallo (como pérdida de Wi-Fi o fallo en la conmutación del relé), gestión de reconexiones automáticas y optimización de consumo en modo deep-sleep para maximizar la autonomía en entornos con suministro limitado. Con este diseño, no solo se reduce drásticamente el esfuerzo físico y el tiempo invertido por los usuarios rurales, sino que también se promueve la modernización del campo, fomentando un uso más eficiente y sostenible de los recursos hídricos a través de tecnología IoT de fácil implementación.

2. Definición del problema y toma de requerimientos.

En muchas zonas rurales, el proceso de encender o apagar una bomba de agua representa un desafío logístico importante para los usuarios, quienes deben recorrer largas distancias a pie para acceder al punto de control, lo cual implica una inversión considerable de tiempo y esfuerzo físico. Esta situación no solo retrasa las actividades relacionadas con el riego o el abastecimiento de agua, sino que también dificulta la implementación de horarios precisos y una gestión eficiente de los recursos hídricos, afectando directamente a la productividad y sostenibilidad de estas comunidades. A partir de este escenario, se identificó la necesidad de desarrollar una solución tecnológica que permitiera controlar de manera remota el sistema de bombeo, brindando una interfaz accesible para cualquier usuario, sin requerimientos técnicos avanzados ni infraestructura compleja. El sistema debía ser modular, de bajo costo, energéticamente eficiente y fácil de instalar, considerando las limitaciones comunes en entornos rurales como la falta de herramientas especializadas, condiciones climáticas adversas y conectividad inestable. En función de lo anterior, se establecieron algunos requisitos para cumplir con las necesidades del proyecto a desarrollar.

- Control remoto de la bomba de agua mediante una interfaz accesible desde dispositivos móviles, usando una plataforma accesible tanto en sistemas operativos android y IOS.
- Conectividad inalámbrica estable utilizando Wi-Fi para la comunicación entre el sistema embebido y la nube.
- Uso de una placa ESP32, programada en C desde el IDE de Arduino, como núcleo del sistema de control.
- Integración de un módulo de relé XN13, capaz de conmutar cargas de hasta 10 A a 125 VAC, para el accionamiento de la bomba.
- Retroalimentación visual inmediata mediante un módulo LED RGB XN007, que indique el estado del sistema (funcionando, error, espera).
- Diseño modular que facilite la conexión de los componentes sin necesidad de soldaduras, permitiendo el ensamblaje y mantenimiento por parte de usuarios no especializados.
- Implementación de rutinas de ahorro energético, como el uso del modo deep-sleep

en la ESP32, para optimizar el consumo en contextos donde el acceso a la energía puede ser limitado.

- Tolerancia a fallos, mediante rutinas que gestionen la reconexión automática y la recuperación tras pérdidas de red o errores en el proceso de conmutación.

Esta definición del problema y la recopilación de requerimientos permitieron estructurar una solución concreta, ajustada al contexto rural, con un enfoque práctico que busca empoderar a los usuarios mediante el acceso a tecnologías de automatización confiables, sostenibles y fácilmente replicables.

3. Diseño de la arquitectura IoT.

El diseño de la arquitectura IoT consistió en estructurar los componentes físicos, de comunicación y de software que conforman el sistema de control remoto para la bomba de agua. La arquitectura se basó en una placa ESP32 como dispositivo principal, encargada de ejecutar el código en C y gestionar la conectividad Wi-Fi con la plataforma Blynk. Se integraron módulos periféricos como el relé XN13 para la conmutación de la bomba y el LED RGB XN007 para la señalización de estados. La comunicación entre el usuario y el sistema se estableció mediante la nube de Blynk, permitiendo el envío de comandos y la visualización de estados en tiempo real a través de una aplicación móvil. Esta arquitectura modular y escalable garantizó una solución confiable, de bajo consumo y fácil implementación en contextos rurales, cada una de las capas implementadas seran explicadas a detalle en los siguientes puntos.

I. Capa de dispositivo.

La capa de dispositivo, también conocida como capa física o "thing" dentro del modelo IoT, representa el conjunto de elementos que interactúan directamente con el entorno físico y constituyen la base funcional del sistema. En el presente proyecto, esta capa está protagonizada por una placa de desarrollo ESP32,

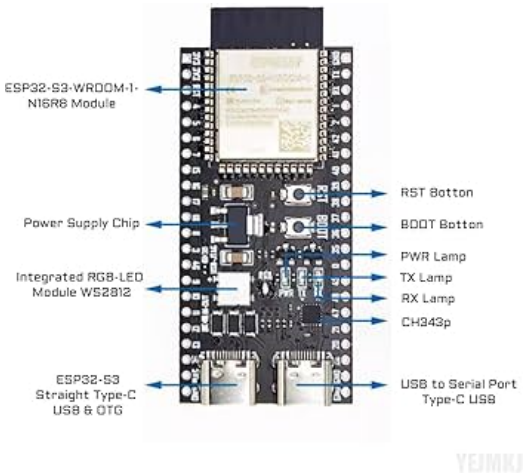

Figura 1: EPS32 placa de desarrollo.

seleccionada por su bajo consumo energético, su capacidad de procesamiento, sus múltiples periféricos integrados y su conectividad inalámbrica a través de Wi-Fi. Esta placa fue programada íntegramente en lenguaje C utilizando el entorno de desarrollo Arduino IDE, permitiendo el control eficiente de los módulos periféricos conectados al sistema. Entre ellos, destaca el módulo de relé XN13,

Figura 2: Placa XN13 con relevador.

encargado de realizar la conmutación eléctrica que activa o desactiva la bomba de agua según las instrucciones recibidas desde la nube. Complementariamente, se incorporó el módulo

XN007, un LED RGB que opera como indicador visual del estado del sistema: verde para funcionamiento correcto, rojo para error o fallo, y azul para estado de espera o reposo. El diseño físico de esta capa apostó por un enfoque modular, donde los componentes se conectan como bloques de construcción tipo "Lego", eliminando la necesidad de soldaduras y herramientas especializadas, lo que facilita su montaje, mantenimiento y reemplazo en campo, especialmente en zonas rurales. Adicionalmente, se implementaron mecanismos de ahorro energético mediante el uso de los modos de suspensión de la ESP32, así como rutinas de monitoreo interno para validar el estado del sistema, identificar posibles errores de red o funcionamiento, y ejecutar respuestas automáticas. Esta capa es fundamental, ya que es responsable tanto de ejecutar las acciones físicas como de mantener la conectividad activa con las capas superiores del ecosistema IoT.

II. Capa de conectividad.

La capa de conectividad es el componente intermedio entre el dispositivo físico (capa "thing") y los servicios en la nube. Su función principal es garantizar que los datos generados o solicitados por el sistema puedan ser transmitidos de forma confiable, segura y en tiempo real hacia una plataforma de gestión remota, en este caso Blynk.

Figura 3: Plataforma Blynk.

En este proyecto, la conectividad se estableció a través de una red Wi-Fi, permitiendo que la placa ESP32 mantuviera un canal de comunicación constante con los servidores en la nube. El protocolo de comunicación utilizado fue HTTP, una elección que ofrece simplicidad, amplia compatibilidad y facilidad de implementación en dispositivos embebidos. A través de este protocolo, la ESP32 realizó solicitudes y respuestas (GET y POST) para actualizar el estado del dispositivo, recibir comandos del usuario y sincronizar variables de control. La conexión Wi-Fi fue gestionada directamente desde el firmware del microcontrolador, incluyendo rutinas para la reconexión automática en caso de pérdida de señal, y validaciones periódicas del estado de la red. Esta capa también fue responsable de asegurar la entrega oportuna de los datos de entrada y salida, minimizando la latencia entre la acción del usuario y la respuesta física del sistema (por ejemplo, encender o apagar la bomba). Gracias a la integración efectiva de esta capa, fue posible establecer una comunicación remota y transparente entre el usuario y el sistema de bombeo, permitiendo el monitoreo y control desde cualquier ubicación con acceso a Internet, sin la necesidad de una infraestructura local compleja.

III. Capa de aplicación.

La capa de aplicación representa el nivel más alto dentro de la arquitectura IoT y es la encargada de ofrecer al usuario final una interfaz intuitiva, accesible y funcional para la supervisión y el control del sistema. En este proyecto, dicha capa fue implementada utilizando la plataforma Blynk,

Figura 4: Interfaz grafica creada en Blynk.

una herramienta que permite la creación de paneles de control visuales personalizados, accesibles desde dispositivos móviles o navegadores web. A través de Blynk, se diseñó una interfaz gráfica amigable que permite al usuario visualizar el estado actual del sistema (por ejemplo, si la

bomba está encendida o apagada), recibir alertas en caso de fallos, e interactuar directamente con el hardware mediante botones virtuales. Esta interfaz se comunica con la ESP32 mediante peticiones HTTP, enviando comandos y recibiendo respuestas en tiempo real. El entorno de Blynk también facilitó la integración de widgets dinámicos, como indicadores LED virtuales, terminales de texto y displays de estado, los cuales se actualizaron en función de los datos proporcionados por el microcontrolador. Esta visualización fue clave para garantizar una experiencia de usuario clara y eficaz, incluso para personas sin conocimientos técnicos avanzados. En conjunto, la capa de aplicación no solo centralizó el control del sistema, sino que además permitió su uso desde cualquier ubicación con acceso a Internet, eliminando por completo la necesidad de intervención física en campo para tareas rutinarias como el encendido o apagado de la bomba de agua.

4. Diagrama de arquitectura del sistema.

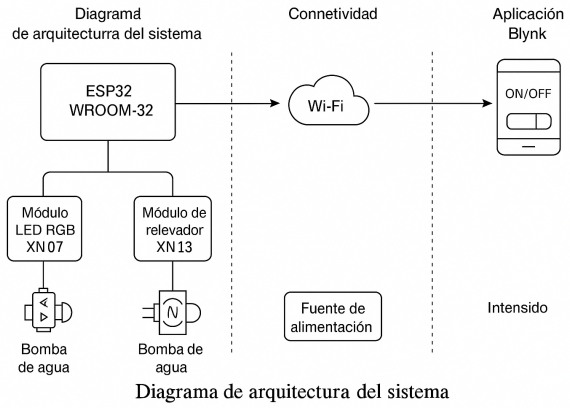

Figura 5: Esquema del funcionamiento de la aplicación.

I. Descripción del Diagrama de Arquitectura del Sistema

El diagrama de arquitectura del sistema representa la organización y funcionamiento de los diferentes componentes involucrados en el sistema IoT para el control remoto de una bomba de agua en zonas rurales. A continuación, se describen detalladamente cada una de las capas que conforman el sistema:

I.1. 1. Capa de Aplicación (Usuario Final)

Esta capa está representada por la **aplicación móvil Blynk**, la cual sirve como interfaz principal para el usuario. Desde ella, el usuario puede:

- Enviar comandos para encender o apagar la bomba de agua.
- Visualizar el estado del sistema mediante indicadores gráficos (LED virtual, botones, displays).
- Recibir retroalimentación en tiempo real.

La comunicación se realiza mediante el **protocolo HTTP** sobre una red **Wi-Fi**, lo que permite el control remoto del sistema desde cualquier ubicación con acceso a internet.

I.2. 2. Capa de Conectividad

Esta capa actúa como puente entre el dispositivo físico (ESP32) y la aplicación móvil. Está compuesta por:

- Una **red Wi-Fi local**, que permite la conexión del microcontrolador a internet.
- El **protocolo HTTP**, encargado de transmitir los comandos del usuario hacia el ESP32, así como de enviar retroalimentación desde el dispositivo hacia la aplicación.

I.3. 3. Capa de Dispositivo ("Thing")

Esta capa contiene los elementos físicos que ejecutan las instrucciones del sistema:

- **ESP32-WROOM-32**: microcontrolador principal que ejecuta la lógica del sistema, controlando periféricos y gestionando la conectividad. Fue programado en lenguaje C mediante el IDE de Arduino.
- **Módulo de relevador XN13**: actúa como interruptor electrónico que permite activar o desactivar la bomba de agua.
- **Bomba de agua**: actuador final del sistema.
- **Módulo LED RGB XN007**: proporciona retroalimentación visual del estado del sistema, mediante códigos de color:

- Verde: sistema activo.
- Azul: esperando comandos.
- Rojo: error de comunicación o sistema detenido.

- **Conectores JST y mikroBUS™** : permiten una conexión tipo "Lego" entre módulos, facilitando el montaje y mantenimiento del sistema.

- **Jumpers de configuración**: usados para seleccionar el modo de operación (maestro/esclavo) y el nivel de voltaje (3.3V o 5V).

- **Puerto micro USB tipo B**: utilizado tanto para alimentación del sistema como para la programación del ESP32.

I.4. Flujo de Operación

1. El usuario accede a la aplicación Blynk en su dispositivo móvil.
2. Envía un comando (por ejemplo, encender la bomba).
3. El comando viaja mediante HTTP a través de la red Wi-Fi hasta el ESP32.
4. El ESP32 activa el módulo de relevador que enciende la bomba.
5. Se actualiza el estado del LED RGB para reflejar la acción ejecutada.
6. La aplicación recibe la confirmación del cambio de estado.

Esta arquitectura modular permite una fácil implementación, mantenimiento y escalabilidad del sistema, resultando ideal para entornos con acceso limitado a infraestructura tecnológica, como lo son las zonas rurales.

5. Diagrama del hardware utilizado.

I. X-NODEC01.

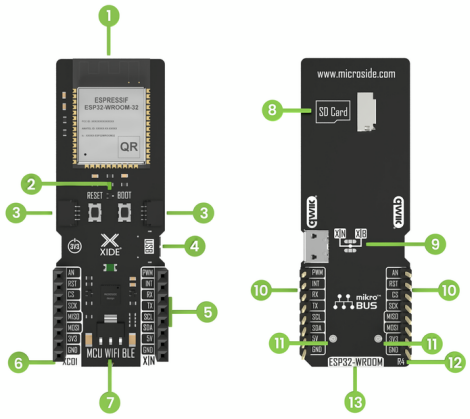

Figura 6: Tarjeta de desarrollo.

1. SoC ESP32-WROOM-32 de Espressif Systems
2. Push button para entrar en modo bootloader y push button conectado al pin RESET
3. Conectores JST compatibles con Qwiic
4. Conector micro USB tipo B para comunicación y alimentación
5. Puertos de comunicación UART ↔ I2C
6. Modelo de X-NODE
7. Tipo de X-NODE
8. Ranura para tarjeta microSD
9. Jumpers para selección de modo X-NODE (Esclavo) ↔ X-BOARD (Maestro)
10. Conectores estándar mikroBUS
11. Jumpers para brindar voltaje de 3.3V y 5V en los pines mikroBUS
12. Versión del hardware: R4
13. Número de parte del componente principal en el X-NODE

II. X-NODE13.

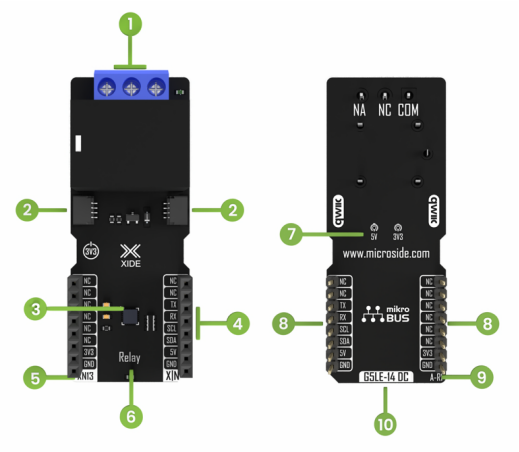

Figura 7: Placa XN13 diagrama.

1. Clemas de conexión a los contactos de salida (NA, NC, COM) del relevador G5LE-14 de OMRON
2. Conectores JST compatibles con Qwiic
3. Controlador en hardware
4. Puertos de comunicación UART <>I2C
5. Modelo de X-NODE
6. Tipo de X-NODE
7. Jumpers de selección para el pin que alimentará el relevador
8. Conectores estándar mikroBUS
9. Versión de hardware: R1
10. Componente principal en el X-NODE

III. X-NODE007.

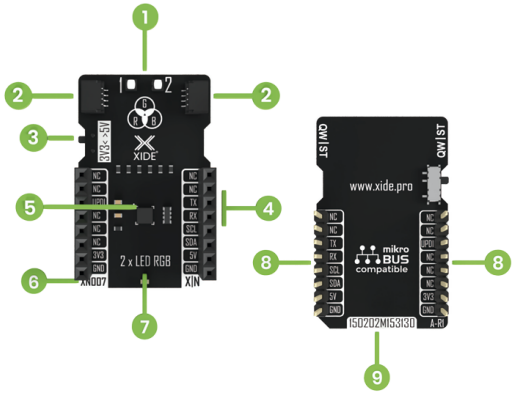

Figura 8: Placa XN007 LED RGB.

1. Indicadores LED RGB
2. Conectores JST compatibles con Qwiic y STEMMA QT
3. Selector de voltaje 3.3V ↔ 5V
4. Puertos de comunicación UART ↔ I2C (Conectados al controlador en hardware)
5. Controlador en hardware
6. Modelo de X-NODE
7. Tipo de X-NODE
8. Conectores estándar mikroBUS Compatible
9. Componente principal en el X-NODE

6. Código fuente utilizado en la solución.

El programa desarrollado tiene como objetivo controlar de forma remota una bomba de agua mediante la plataforma IoT **Blynk**, utilizando un microcontrolador **ESP32**, un módulo **X-NODE**, y comunicación vía **Wi-Fi**.
Código fuente: https://github.com/OmarGudi/Proyecto_Final
El funcionamiento se estructura en los siguientes componentes:

- **Definiciones iniciales:** Se definen el nombre del proyecto, el ID del template y el token de autenticación de Blynk necesarios para establecer la comunicación entre el

microcontrolador y la plataforma Blynk Cloud. También se incluyen las bibliotecas esenciales para el funcionamiento: `Wire.h` (I2C), `XNODE.h` (control del módulo), `WiFi.h`, `WiFiClient.h` y `BlynkSimpleEsp32.h`.

- **Configuración de pines y tiempos:** Se define el pin `CS_PIN`, y dos temporizadores:
  - `TIME_BEFORE_CHECKING`: intervalo entre verificaciones de estado.
  - `TIME_ON`: tiempo máximo que la bomba puede permanecer encendida antes de apagarse automáticamente, este tiempo va a variar para cada usuario segun las necesidades y requisitos que ellos tengan.

  Asimismo, se especifican los datos de la red Wi-Fi y se asignan los pines virtuales de la aplicación Blynk (`V0` para el botón de control y `V1` para mostrar el estado de la bomba).

- **Inicialización de objetos y variables:** Se instancia el objeto `XNODE`, utilizando `Serial2` para la comunicación UART, y se inicializan variables para manejar el estado de la bomba (`bomb_state`) y llevar el conteo del tiempo (`startTime`).

- **Control de la bomba mediante Blynk:** La función `BLYNK_WRITE(VIRTUAL_BOMB_BUTTON)` se ejecuta automáticamente cuando el usuario presiona el botón en la aplicación móvil. Si se activa (`pinValue == 1`) y la bomba está apagada, se enciende la bomba, se actualiza el estado y se registra el tiempo de activación.

- **Configuración en `setup()`:**
  - Se inicializan las comunicaciones `Serial`, `Serial2` y `Wire`.
  - Se conecta a la red Wi-Fi, mostrando progresivamente el estado de conexión. Durante el intento de conexión, se envían comandos visuales al módulo XNODE para encender ciertos LEDs como indicativo visual.
  - Una vez conectado a la red, se inicializa la sesión con Blynk.
  - Se establece el estado inicial de la bomba (encendida o apagada según la variable `bomb_state`).

- **Bucle principal en `loop()`:**
  - Se mantiene el servicio de Blynk activo mediante `Blynk.run()`.
  - Se monitorea si la bomba ha superado el tiempo máximo encendida (`TIME_ON`). De ser así, se apaga automáticamente.
  - Se actualiza constantemente el estado de la bomba en la aplicación móvil usando `virtualWrite` en el pin virtual `V1`.
  - Se envía un comando de control al módulo XNODE, actualizando el estado físico de la bomba según corresponda.
  - Se incluye un retardo (`delay(TIME_BEFORE_CHECKIN`) para espaciar la revisión del estado.

- **Funciones de control específicas:**
  - `encenderBomba()`: Enciende la bomba enviando comandos específicos al módulo XNODE para cambiar el estado de los canales 1 y 2.
  - `apagarBomba()`: Apaga la bomba enviando los comandos correspondientes para devolver los canales a su estado original.

I. Resumen de la funcionalidad

Gracias a la integración de Blynk, Wi-Fi y el control físico vía XNODE, el sistema permite que el usuario:

- Active o desactive la bomba remotamente desde su teléfono móvil.
- Reciba actualizaciones inmediatas del estado de la bomba.
- Garantice que la bomba no permanezca encendida más tiempo del permitido automáticamente, protegiendo así el sistema de posibles fallos o daños.

7. Interfaz realizada.

La construcción de la interfaz gráfica de usuario se llevó a cabo utilizando la plataforma Blynk, tal como se expuso anteriormente en este documento. Esta herramienta fue elegida debido a su facilidad de uso, su amplia compatibilidad con dispositivos móviles y su soporte para acceso vía navegador web. Gracias a Blynk, se logró diseñar un entorno visual intuitivo que permite al usuario final interactuar de manera sencilla y efectiva con el sistema de control de la bomba de agua. La aplicación desarrollada no sólo permite el monitoreo en tiempo real del estado del dispositivo, sino también su activación y desactivación remota. Asimismo, la plataforma web de Blynk ofrece una alternativa adicional para gestionar el sistema desde cualquier computadora con acceso a Internet, proporcionando flexibilidad y mejorando la accesibilidad de la solución tecnológica propuesta.

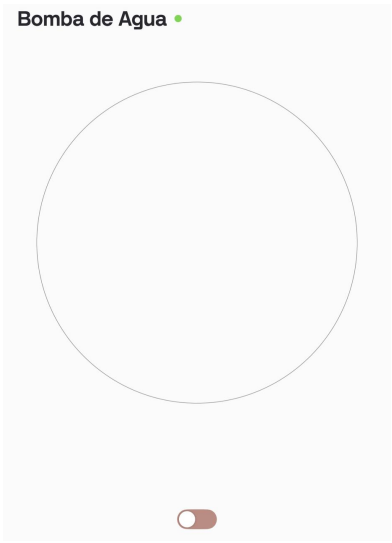

Figura 9: Interfaz de usuario para dispositivos móviles.

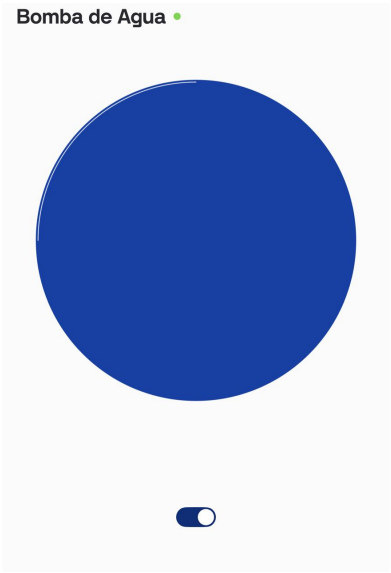

Figura 10: Interfaz de usuario para dispositivos móviles.

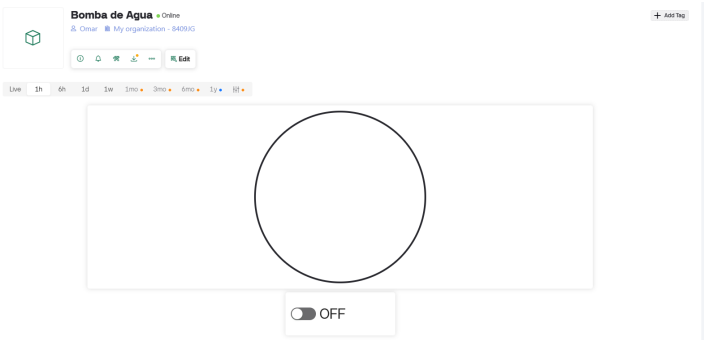

Figura 11: Interfaz de usuario en la pagina web.

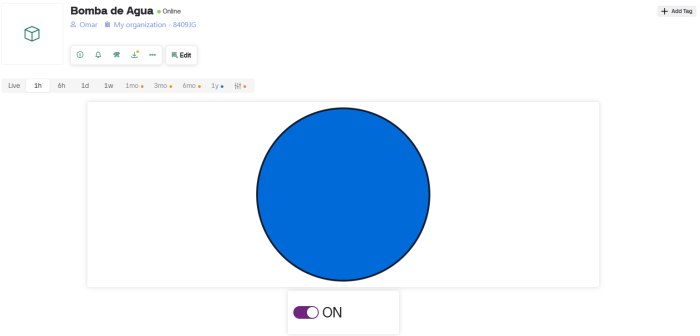

Figura 12: Interfaz de usuario en la pagina web.

8. Justificación técnica de las decisiones tomadas.

I. Selección del Módulo X-NODE XN13-1 (Relevador 10A)

El módulo **XN13-1** fue seleccionado por sus características técnicas y su integración nativa con la plataforma **XIDE IoT** de Microside. Este relevador de estado físico permite controlar cargas de hasta **10 amperios a 250V AC o 30V DC**, cumpliendo con los requerimientos de aplicaciones de automatización como encendido de luces, motores pequeños, válvulas solenoides, entre otros. La elección del XN13-1 también se basa en:

- Su encapsulado compacto tipo X-NODE, que facilita el montaje y mantenimiento.
- La protección óptica y aislamiento galvánico entre el microcontrolador y la carga, lo cual mejora la seguridad del sistema.
- Su LED indicador de estado, que proporciona retroalimentación visual inmediata del funcionamiento.

II. Plataforma de Desarrollo: Arduino (ESP32 o similar)

La decisión de utilizar un microcontrolador compatible con Arduino (como un ESP32) se fundamenta en:

- Facilidad de programación y prototipado.
- Compatibilidad con la biblioteca `XNODE`, que permite una interfaz rápida y clara con módulos X-NODE mediante comandos predefinidos.
- Soporte para múltiples puertos UART (`Serial`, `Serial1`, `Serial2`), ideal para mantener separados los canales de depuración y de control.

Arduino también reduce la curva de aprendizaje, facilita pruebas rápidas y es altamente documentado, lo cual es útil tanto en entornos académicos como profesionales.

III. Comunicación Serial UART

El protocolo UART fue utilizado para establecer comunicación entre el microcontrolador y el XN13-1. Esta decisión se basa en las siguientes ventajas:

- UART es un protocolo simple, eficiente y ampliamente soportado.
- Reduce el uso de pines al requerir solo RX y TX.
- Es ideal para comunicación punto a punto con dispositivos como sensores, actuadores o módulos de control.

Se utilizó `Serial2` específicamente en los pines `GPIO34` y `GPIO35` para mantener libre el puerto serial principal para depuración.

IV. Control del Relevador mediante Comandos de Texto

La decisión de controlar el relevador usando la función `xnode.SendCommand(char*, char*, char*)` responde a:

- Un diseño modular, donde cada módulo puede recibir y ejecutar comandos claros como `"S"`, `"0"` o `"S"`, `"1"`, donde `"S"` representa el switch o canal, y `"0/"1"` representan su estado.
- Permite escalabilidad: varios módulos pueden ser controlados desde un mismo microcontrolador cambiando únicamente el ID del dispositivo.

Se solucionó un problema de compilación causado por un desajuste de tipos (`const char*` vs `char*`) mediante el uso de un *casting* explícito `(char*)`, sin modificar la librería base. Esto mantuvo la integridad del código original.

V. Verificación del Funcionamiento del Relevador

Se llevaron a cabo varios métodos para validar que el relevador funciona correctamente:

- **Prueba de clic mecánico**: Se escucha un sonido cada vez que el relevador cambia de estado.

- **LED indicador de estado**: Se verifica si el LED del módulo cambia al enviar comandos.
- **Medición con multímetro**: Se revisa continuidad entre los terminales `COM`, `NO` y `NC` del relevador bajo diferentes estados.
- **Pruebas de carga**: Se conectó una carga real (como una lámpara) para confirmar que el relevador la conmuta correctamente.

VI. Validación del ID del Módulo (XN13A)

Cada módulo X-NODE posee un identificador único (por ejemplo, `"XN13A"`) que debe coincidir con el configurado en el código. Este ID es necesario para que el sistema pueda diferenciar entre varios módulos conectados. Se validó que el ID fuera correcto al:

- Confirmar que los comandos enviados al ID `"XN13A"` provocaban respuesta.
- Revisar el hardware para verificar configuraciones mediante jumpers o switches (cuando aplique).
- Probar respuesta en consola serial con retroalimentación `Çommand Sent"` y cambios de estado físicos.

VII. Modularidad y Escalabilidad del Sistema

La arquitectura propuesta permite que se puedan agregar más módulos X-NODE (relevadores, sensores, etc.) con facilidad. Cada módulo responde a un ID distinto, y el software puede ser fácilmente adaptado para incluir nuevas instrucciones o módulos. Esto favorece:

- **La escalabilidad horizontal**: más módulos en la misma red.
- **El mantenimiento**: los módulos defectuosos pueden ser reemplazados fácilmente.
- **La flexibilidad**: permite implementar lógica compleja como temporización, activación por eventos, automatización condicional, etc.

VIII. Integración con Plataforma IoT (XIDE IoT)

El módulo **XN13-1** es parte del ecosistema **XIDE IoT**, lo cual permite:

- Visualizar el estado del módulo en tiempo real desde una interfaz web.
- Controlarlo remotamente mediante dashboards o reglas automatizadas.
- Integrarlo a proyectos de domótica o sistemas industriales sin desarrollar software desde cero.

Esta decisión técnica permite crear una solución moderna, escalable y profesional desde el inicio del desarrollo.

IX. Conectividad WiFi

Se optó por utilizar conectividad WiFi como medio de enlace entre el microcontrolador y la plataforma IoT, permitiendo el control y monitoreo remoto del sistema sin necesidad de cables físicos. Las razones técnicas incluyen:

- Alta disponibilidad de redes WiFi en entornos domésticos, comerciales e industriales.
- Integración nativa del módulo WiFi en microcontroladores como el ESP32, eliminando hardware adicional.
- Ancho de banda suficiente para el envío y recepción de comandos o estados.
- Configuración sencilla utilizando SSID y contraseña, compatible con entornos dinámicos.

X. Uso de la Plataforma Blynk IoT

Se eligió Blynk como plataforma de monitoreo y control por su facilidad de uso, compatibilidad con microcontroladores y disponibilidad multiplataforma. Entre sus ventajas destacan:

- Interfaz gráfica accesible desde dispositivos móviles y navegadores web.
- Capacidad para enviar comandos remotos (ej. encender o apagar relevadores).

- Infraestructura en la nube segura y confiable.
- Herramientas integradas para automatización, temporizadores, alertas y visualización de datos.
- API sencilla para integrarse con sistemas embebidos vía HTTP o MQTT.

XI. Protocolo HTTP

Se utilizó el protocolo HTTP para la comunicación entre el microcontrolador y la nube, debido a sus múltiples beneficios técnicos:

- Protocolo estándar ampliamente soportado por plataformas y servidores.
- Permite realizar solicitudes RESTful (GET, POST, PUT), ideales para sistemas de control.
- Fácil de implementar mediante bibliotecas disponibles en el entorno Arduino (ej. `HTTPClient.h`).
- Requiere pocos recursos computacionales, siendo adecuado para microcontroladores con memoria y procesamiento limitados.
- Facilita la integración futura con otras APIs o plataformas IoT.

9. Comparativas de tecnologías.

I. Sensores.

A continuación, se presentan diferentes sensores o tecnologías que podrían haberse utilizado para lograr funcionalidades similares, junto con las razones por las cuales no fueron seleccionadas.

I.1. Sensor PIR (Detector de Movimiento)

- **Funcionalidad:** Detecta movimiento de personas.
- **Posible uso:** Encender o apagar cargas al detectar movimiento.
- **Desventaja:** No permite un control manual remoto; depende exclusivamente de la presencia física.

I.2. Sensor de Corriente ACS712

- **Funcionalidad:** Detecta el flujo de corriente eléctrica.
- **Posible uso:** Supervisar si una carga está encendida o apagada.
- **Desventaja:** Solo monitorea; no tiene capacidad de actuar sobre el estado de las cargas.

I.3. Sensor de Contacto Magnético (Reed Switch)

- **Funcionalidad:** Detecta apertura o cierre de puertas y ventanas.
- **Posible uso:** Activar cargas basado en eventos físicos de apertura o cierre.
- **Desventaja:** Limitado a un tipo de evento físico; no ofrece control programable o remoto.

I.4. Relé de Estado Sólido (SSR)

- **Funcionalidad:** Actúa como interruptor electrónico para cargas de CA.
- **Posible uso:** Encender/apagar cargas eléctricas de forma silenciosa y rápida.
- **Desventaja:** Es más costoso, sensible a sobretensiones, y carece de integración nativa con la plataforma XNODE.

I.5. Justificación de la Elección del Relevador XN13-1

- Permite control seguro de cargas de alta corriente (hasta 10A).
- Ofrece aislamiento galvánico mediante optoacopladores.
- Incluye indicador LED para confirmación visual del estado.
- Totalmente integrado con la biblioteca XNODE y la plataforma IoT XIDE.
- Facilita expansión y modularidad en el sistema.

- Proporciona control manual y remoto eficiente, seguro y documentado.

II. Módulos de procesamiento.

II.1. ESP32

- **Funcionalidad:** Microcontrolador de alto rendimiento con conectividad WiFi y Bluetooth integrada.
- **Ventajas:**
  - Doble núcleo a 240 MHz.
  - Múltiples interfaces de comunicación (UART, SPI, I2C, CAN).
  - Amplia comunidad de soporte.
  - Bajo consumo de energía.
- **Desventaja:** Puede ser más costoso que soluciones más básicas si no se aprovechan todas sus capacidades.

II.2. ESP8266

- **Funcionalidad:** Microcontrolador WiFi de bajo costo.
- **Ventajas:**
  - Bajo consumo de energía.
  - Amplio soporte para proyectos IoT simples.
  - Precio accesible.
- **Desventaja:** Tiene un solo núcleo y menor capacidad de procesamiento que el ESP32.

II.3. Arduino UNO (ATmega328P)

- **Funcionalidad:** Microcontrolador clásico para prototipado.
- **Ventajas:**
  - Gran facilidad de uso y aprendizaje.
  - Comunidad masiva y abundante documentación.
- **Desventaja:** No posee conectividad WiFi o Bluetooth integrada; se necesitarían módulos externos, aumentando la complejidad y los costos.

II.4. Raspberry Pi Pico

- **Funcionalidad:** Microcontrolador basado en RP2040 de alto rendimiento.
- **Ventajas:**
  - Bajo costo.
  - Procesador dual-core ARM Cortex-M0+.
- **Desventaja:** Carece de conectividad inalámbrica nativa; requiere módulos adicionales para WiFi/Bluetooth.

II.5. STM32 (familia ARM Cortex-M)

- **Funcionalidad:** Microcontroladores de 32 bits de alto rendimiento para aplicaciones industriales.
- **Ventajas:**
  - Alta velocidad y bajo consumo.
  - Soporte para interfaces avanzadas de comunicación (CAN, Ethernet, etc.).
- **Desventaja:** Mayor complejidad de programación y menor accesibilidad para prototipos rápidos.

II.6. Justificación de la Elección del ESP32

- **Integración total**: Cuenta con WiFi y Bluetooth, eliminando la necesidad de módulos externos.
- **Potencia de procesamiento**: Ideal para manejar comunicaciones UART, procesamiento local y conectividad IoT simultáneamente.
- **Compatibilidad**: Soporta la librería XNODE de Microside directamente.
- **Escalabilidad**: Permite el futuro crecimiento del sistema (adición de más nodos, sensores o funciones).
- **Optimización de costos**: Ofrece mejor relación costo/beneficio en comparación con agregar múltiples módulos a un Arduino o un Raspberry Pi Pico.

III. Protocolos de comunicación.

Se evaluaron distintos protocolos de comunicación para el proyecto, considerando facilidad de implementación, compatibilidad y eficiencia.

III.1. UART (Universal Asynchronous Receiver/Transmitter)

- **Funcionalidad:** Comunicación serial asíncrona punto a punto.
- **Ventajas:**
  - Muy simple de implementar.
  - Requiere pocos pines (RX y TX).
  - Ampliamente soportado en microcontroladores.
- **Desventaja:** Comunicación exclusiva entre dos dispositivos; no es ideal para redes grandes sin multiplexores o expansores.

III.2. I2C (Inter-Integrated Circuit)

- **Funcionalidad:** Comunicación serial síncrona para múltiples dispositivos.
- **Ventajas:**
  - Permite conectar múltiples dispositivos a través de dos pines (SCL y SDA).
  - Ideal para distancias cortas dentro de un mismo PCB o módulo.
- **Desventaja:** No tan robusto para distancias largas o ambientes ruidosos; velocidades más limitadas comparadas con otros protocolos.

III.3. SPI (Serial Peripheral Interface)

- **Funcionalidad:** Comunicación serial síncrona de alta velocidad.
- **Ventajas:**
  - Muy rápida (hasta decenas de Mbps).
  - Ideal para transferencias grandes de datos.
- **Desventaja:** Requiere más líneas físicas (MISO, MOSI, SCK, SS) y complejidad en el manejo de múltiples dispositivos esclavos.

III.4. CAN (Controller Area Network)

- **Funcionalidad:** Red robusta diseñada para comunicaciones en ambientes industriales o automotrices.
- **Ventajas:**
  - Alta fiabilidad y tolerancia a fallos.
  - Comunicación multi-maestro con control de errores integrado.
- **Desventaja:** Requiere transceptores adicionales y configuración más compleja.

III.5. HTTP sobre WiFi

- **Funcionalidad:** Comunicación basada en protocolo de internet a través de solicitudes HTTP.
- **Ventajas:**
  - Muy compatible con plataformas IoT y web.
  - Permite control remoto desde cualquier lugar con acceso a internet.
- **Desventaja:** Mayor consumo de energía comparado con protocolos seriales simples; depende de una red WiFi funcional.

III.6. Justificación de la Elección de UART y HTTP

- **UART:**
  - Es la mejor opción para comunicación directa entre el microcontrolador (ESP32) y el módulo X-NODE XN13-1, con implementación simple y rápida.
  - Ideal para control de hardware físico a corta distancia y baja complejidad.
- **HTTP sobre WiFi:**
  - Permite monitoreo y control remoto usando plataformas como Blynk y XIDE IoT.

- Facilita la visualización en tiempo real y la integración con interfaces gráficas amigables para el usuario final.
- Es ideal para aplicaciones modernas de IoT, donde el acceso remoto y la escalabilidad son cruciales.

IV. Plataformas IoT.

IV.1. Blynk IoT

- **Funcionalidad:** Plataforma IoT enfocada en la creación rápida de interfaces móviles para control y monitoreo.
- **Ventajas:**
  - Interfaz gráfica intuitiva para creación de dashboards sin necesidad de programar aplicaciones móviles.
  - Amplio soporte para microcontroladores como ESP32, ESP8266 y Arduino.
  - Comunicación segura mediante autenticación por tokens.
- **Desventaja:** Plan gratuito limitado en la cantidad de dispositivos y funciones avanzadas; requiere internet constante para el funcionamiento.

IV.2. XIDE IoT

- **Funcionalidad:** Plataforma diseñada específicamente para dispositivos Microside, como los módulos X-NODE.
- **Ventajas:**
  - Integración nativa y directa con módulos X-NODE, minimizando configuración inicial.
  - Permite visualización en tiempo real, programación de eventos y reglas automáticas.
  - Dashboard web amigable y adaptable.
- **Desventaja:** Ecosistema más cerrado, dependiente del hardware específico de Microside.

IV.3. TagoIO

- **Funcionalidad:** Plataforma IoT general para integración de dispositivos, visualización de datos y creación de aplicaciones inteligentes.
- **Ventajas:**
  - Alto grado de personalización en dashboards y reportes.
  - API REST potente para integraciones externas.
  - Funcionalidades de análisis de datos, alertas y control de usuarios.
- **Desventaja:** Curva de aprendizaje más alta; puede ser excesiva para proyectos simples.

IV.4. Ubidots

- **Funcionalidad:** Plataforma IoT enfocada en soluciones industriales y educativas.
- **Ventajas:**
  - Herramientas de visualización muy potentes.
  - Funciones de alerta avanzada, análisis y predicción de datos.
  - Alta compatibilidad con protocolos como HTTP, MQTT y TCP/UDP.
- **Desventaja:** El plan gratuito es bastante limitado; los planes pagos pueden ser costosos para proyectos pequeños.

IV.5. Justificación de la Elección de Blynk IoT

- **Blynk IoT:**
  - Permite un desarrollo rápido de interfaces gráficas de control y monitoreo.
  - Ideal para prototipos, demostraciones y proyectos educativos/profesionales de bajo a mediano alcance.
  - Compatible directamente con ESP32 y muy fácil de integrar.

V. ¿Cómo la solución impacta positivamente al usuario, cliente o sector?

La implementación de la solución propuesta tiene múltiples impactos positivos:

V.1. Automatización de Procesos

- Permite el encendido y apagado remoto de cargas eléctricas (luces, bombas, motores) de manera automática o bajo demanda.
- Reduce la necesidad de intervención manual, optimizando tiempos de operación.

V.2. Monitoreo Remoto en Tiempo Real

- A través de plataformas como Blynk IoT, los usuarios pueden visualizar en tiempo real el estado de sus dispositivos desde cualquier parte del mundo.
- Se incrementa la capacidad de supervisión y toma de decisiones basada en datos en vivo.

V.3. Reducción de Costos Operativos

- La automatización de tareas repetitivas minimiza errores humanos y reduce la necesidad de personal exclusivamente destinado a tareas de monitoreo o control manual.
- Se ahorra energía al activar dispositivos únicamente cuando es necesario.

V.4. Mejora de la Seguridad

- Gracias al aislamiento galvánico del módulo X-NODE XN13-1, se protege tanto a los usuarios como al microcontrolador frente a daños eléctricos.
- El monitoreo en tiempo real permite detectar fallas o condiciones anómalas de manera anticipada.

V.5. Escalabilidad y Adaptabilidad

- El sistema puede crecer fácilmente integrando nuevos módulos, sensores o actuadores sin necesidad de rediseñar toda la infraestructura.
- La flexibilidad de la arquitectura permite su aplicación en diversos sectores como agricultura, industria, domótica, entre otros.

V.6. Experiencia de Usuario Mejorada

- Interfaces gráficas intuitivas (dashboards) mejoran la experiencia de usuario, haciendo la operación del sistema accesible incluso a usuarios no técnicos.
- El control desde dispositivos móviles proporciona conveniencia y rapidez en las acciones remotas.

10. Conclusión.

El desarrollo de este sistema de control remoto para una bomba de agua mediante tecnologías IoT representa una solución efectiva y replicable ante una problemática recurrente en zonas rurales: la necesidad de desplazarse largas distancias para activar o desactivar manualmente dispositivos básicos, como lo es una bomba hidráulica. Este proyecto logró implementar una alternativa tecnológica accesible, económica y funcional que automatiza dicha tarea, mejorando significativamente la calidad de vida de los usuarios al ahorrar tiempo, esfuerzo y recursos. La arquitectura del sistema fue diseñada siguiendo el modelo clásico de tres capas del Internet de las Cosas: dispositivo, conectividad y aplicación. En la capa de dispositivo, se empleó la placa de desarrollo ESP32 como núcleo del sistema embebido, aprovechando su capacidad de procesamiento, conectividad nativa Wi-Fi y bajo consumo energético. Esta placa se programó en lenguaje C utilizando el IDE de Arduino, controlando un módulo de relevador XN13, encargado de accionar la bomba, y un módulo LED RGB XN007, destinado a la señalización del estado operativo del sistema. El diseño modular tipo "Lego" de los componentes facilitó su integración, ensamblaje y mantenimiento, elementos clave para su implementación en entornos con limitaciones técnicas. La capa de conectividad se estructuró mediante la conexión Wi-Fi de la ESP32 y la utilización del protocolo HTTP para el intercambio de datos entre el dispositivo físico y la plataforma de gestión. Esta capa incluyó rutinas de reconexión

automática, verificación del estado de red y tolerancia a fallos, asegurando una comunicación confiable y continua con la nube. En cuanto a la capa de aplicación, se utilizó la plataforma Blynk, que permitió construir una interfaz gráfica adaptable e intuitiva para el usuario final. Esta interfaz móvil facilitó la interacción remota con el sistema, mostrando en tiempo real el estado del equipo, permitiendo el control de la bomba y generando una experiencia de uso amigable yinclusopara usuarios sin conocimientos técnicos. Durante la ejecución del proyecto, también se aplicaron técnicas de optimización energética, como el uso de modos de bajo consumo del microcontrolador (por ejemplo, deep-sleep), lo cual resulta esencial en entornos donde la disponibilidad de energía puede ser limitada o intermitente. En resumen, el sistema desarrollado cumple con los requerimientos técnicos y funcionales planteados desde la etapa de análisis del problema. Se alcanzó una solución sólida, escalable y adaptable, con potencial de implementación en diversos escenarios del sector rural. Además, este trabajo refuerza el valor del Internet de las Cosas como una herramienta clave para democratizar el acceso a tecnologías de automatización, impulsando la eficiencia operativa, la sostenibilidad y el desarrollo en comunidades tradicionalmente rezagadas en términos de infraestructura tecnológica.

11. Recursos.

I. Repositorio en Github.

En el siguiente enlace pueden encontrar el script generado para este experimento: https://github.com/OmarGudi/Proyecto_Final

Referencias

[1] Microside Technology. X-NODE XN13-1 Relevador 10A - Documentación Oficial. Microside 2024. https://docs.microside.com/plataforma-xide-iot/x-nodes/x-node-xn13-1-relevador-10a

[2] Blynk IoT Platform. Blynk Documentation. Blynk 2024. https://docs.blynk.io/en/

[3] Espressif Systems. ESP32 Series Datasheet. Espressif 2024. https://www.espressif.com/en/products/socs/esp32/resources

[4] PlatformIO. PlatformIO IDE for VSCode - Documentation. PlatformIO Labs 2025. https://docs.platformio.org/en/latest/integration/ide/vscode.html

[5] Arduino Project Hub. UART Communication: Basics and Implementation. Arduino.cc 2024. https://docs.arduino.cc/learn/communication/uart

[6] Adafruit Industries. Understanding Relays: Basics and Applications. Adafruit Learning System 2024. https://learn.adafruit.com/adafruit-arduino-lesson-13-dc-motors/overview

[7] XIDE IoT Platform. XIDE IoT Cloud Overview. Microside 2024. https://docs.microside.com/plataforma-xide-iot/xide-iot/introduccion

[8] Texas Instruments. Relay Basics: Working Principle, Types, and Applications. Texas Instruments 2024. https://www.ti.com/lit/an/slva930/slva930.pdf

[9] Pimoroni. Comparing Ultrasonic and Infrared Sensors for Obstacle Detection. Pimoroni Learning Center 2024. https://learn.pimoroni.com/tutorial/sandyj/comparing-ultrasonic-and-infrared-distance-sensors

[10] DFRobot. Introduction to Capacitive and Resistive Soil Moisture Sensors. DFRobot Wiki 2024. https://wiki.dfrobot.com/Capacitive_Soil_Moisture_Sensor_SKU_SEN0193

[11] MQTT Organization. MQTT: The Standard for IoT Messaging. MQTT 2024. http://mqtt.org/

[12] HTTP Protocol Overview. MDN Web Docs 2024. https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

[13] Espressif Systems. Wi-Fi Networking with ESP32. Espressif 2024. https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_wifi.html

[14] Cambridge University Press. Internet of Things: Principles and Paradigms. Rajkumar Buyya, Amir Vahid Dastjerdi (2016). ISBN: 9780128053959.