ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE UN PROTOTIPO DE ALERTA POR DETECCIÓN DE LLAMAS BASADO EN LORA Y *TELEGRAM*

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN REDES Y TELECOMUNICACIONES

RONALD ADRIÁN CHASIPANTA GUALPA

DIRECTOR: CARLOS ANDRÉS YUNGA SÁNCHEZ

DMQ, julio 2024

CERTIFICACIONES

Yo, Ronald Adrián Chasipanta Gualpa declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

ADRIAN CHASIPANTA

ronald.chasipanta@epn.edu.ec

chasipantaadrian@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por Ronald Adrián Chasipanta Gualpa, bajo mi supervisión.

Carlos Andrés Yunga Sánchez

DIRECTOR

carlos.yunga@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Ronald Adrián Chasipanta Gualpa

CI: 172526274-3

DEDICATORIA

El siguiente proyecto está dedicado a Dios, quien ha sido mi guía constante y mi fortaleza en este camino académico. A mis amados padres, Esperanza y Juan, les debo un reconocimiento especial. Su inquebrantable respaldo, tanto emocional como financiero, ha sido el cimiento sobre el cual he construido mi formación académica y personal.

Agradezco de manera especial a mi abuelita materna, Clemencia, por su constante apoyo y aliento durante mis estudios universitarios. Su carisma y alegría han sido una inspiración para seguir adelante, demostrándome que los obstáculos pueden superarse con determinación.

Finalmente, quiero expresar mi gratitud a mi hermano, quien ha sido mi ejemplo a seguir y mi compañero incondicional en este viaje. Su apoyo diario me ha motivado a esforzarme aún más, y espero poder ser un modelo para él en la consecución de sus propios sueños futuros.

Ronald Adrián Chasipanta Gualpa

AGRADECIMIENTO

Quiero agradecer a mis queridos padres Esperanza y Juan, les debo una gratitud eterna por su amor incondicional, su incansable apoyo y los sacrificios que han hecho para permitirme cursar mis estudios universitarios.

A los ingenieros y profesores de cada materia, les estoy profundamente agradecido por su dedicación y conocimientos impartidos. Sus enseñanzas han sido la base de mi formación académica y profesional, y les estoy agradecido por su orientación y apoyo constante.

A todos los que han sido parte de mi camino universitario, ya sea de cerca o de lejos, les doy las gracias de todo corazón. Su contribución y aliento han sido fundamentales para alcanzar este logro.

Ronald Adrián Chasipanta Gualpa

ÍNDICE DE CONTENIDO

CERTIFICACIONES I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
AGRADECIMIENTO IV
ÍNDICE DE CONTENIDO V
RESUMEN VIII
ABSTRACT IX
1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO 1
1.1 Objetivo general 2
1.2 Objetivos específicos 2
1.3 Alcance 2
1.4 Marco Teórico 2
Detección de llama 2
Tecnología Lora 3
Sensores de Llama 3
Arduino IDE 3
Telegram y *Bot*s 3
2 METODOLOGÍA 4
3 RESULTADOS 5
3.1 Identificación de los requerimientos para el diseño del prototipo 6
Requisitos de *hardware* 6
Requisitos de *Software* 6
Integración de mensajería *Telegram* 6
Simulación y diseño de la placa 7
Proceso de diseño y montaje 7
3.2 Selección del *hardware* y *software* acorde a los requerimientos establecidos 7
Selección del *Hardware* 7

Microcontrolador 7
Sensor de Llama 9
Selección del *Software* 10
3.3 Diseñar el prototipo de detección de llamas 10
Esquema general 10
Creación del *Bot* en *Telegram* 12
Abrir *Telegram* y buscar *BotFather* 12
Obtener el *chat ID* con ID *Bot* 13
Integración del *Bot* en el Código 13
Comunicación Lora entre las placas de heltec Wifi LoRa v2 15
Trasmisor 16
Receptor 17
Justificación de Intervalos Operativos Establecidos para el Prototipo 18
Cálculo del Consumo de Corriente Total del Prototipo 19
Cálculo de Parámetros de Comunicación LoRa 20
Diagrama de flujo 20
3.4 Implementación del prototipo de detección de llamas 23
Montaje de los componentes de la placa 24
Programación del Módulo Heltec Wifi LoRa v2 (Trasmisor) 25
Programación del Módulo Heltec Wifi LoRa v2 (Receptor) 29
3.5 Pruebas de funcionamiento del prototipo 30
Visualización de Alertas en *Telegram* 33
Costos de fabricación e implementación 34
4 CONCLUSIONES 36
5 RECOMENDACIONES 37
6 BIBLIOGRAFIA 38
7 ANEXOS 42
ANEXO I: Certificado de Originalidad i
ANEXO II: Código fuente trasmisor ii

ANEXO III: Código fuente emisor.. vi

RESUMEN

El siguiente trabajo propone el desarrollo de un prototipo de sistema de alerta para detección de llamas utilizando tecnología LoRa y la red de mensajería *Telegram*. La primera sección analiza los componentes principales del sistema, incluida la placa de desarrollo Heltec Wifi LoRa V2, el sensor de llama LM393, el *software Arduino IDE* y la aplicación *Telegram*, esto incluye objetivos tanto generales como específicos del proyecto.

La segunda sección describe el enfoque para lograr los objetivos específicos, como la configuración de componentes y la programación de dispositivos. Se proporcionan diagramas de conexión y código fuente para ayudar con la replicación del proyecto.

La tercera sección presenta los resultados de las pruebas operativas y demostraciones del prototipo, destacando la función para detectar llamas y emitir alarmas vía *Telegram*. Se discute la importancia de esta tecnología para la detección temprana de incendios y sus usos potenciales en muchos contextos.

La cuarta sección proporciona conclusiones y recomendaciones basadas en las experiencias de elaboración del proyecto, así como sugerencias para futuras mejoras y aplicaciones del sistema de alerta de detección de llamas basado en LoRa y *Telegram*.

Finalmente, los anexos comprenden el código de programación, videos de funcionamiento y cualquier información adicional necesaria para comprender y replicar el prototipo.

PALABRAS CLAVE: Detección de llamas, LoRa, *Telegram*, Heltec Wifi LoRa V2, Sensor de flama LM393, *Arduino IDE*.

ABSTRACT

The following paper proposes the development of a prototype flame detection alert system using LoRa technology and the Telegram messaging network. The first section discusses the main components of the system, including the Heltec Wifi LoRa V2 development board, the LM393 flame sensor, the Arduino IDE software, and the Telegram app, this includes both general and specific project objectives.

The second section describes the approach to achieve the specific objectives, such as component configuration and device programming. Connection diagrams and source code are provided to assist with project replication.

The third section presents the results of operational tests and demonstrations of the prototype, highlighting the function to detect flames and issue alarms via Telegram. The importance of this technology for early fire detection and its potential uses in many contexts are discussed.

The fourth section provides conclusions and recommendations based on the experiences of developing the project, as well as suggestions for future improvements and applications of the LoRa and Telegram-based flame detection alert system.

Finally, the appendices include the programming code, operating videos and any additional information necessary to understand and replicate the prototype.

KEYWORDS: *Flame detection, LoRa, Heltec Wifi LoRa V2, LM393 flame sensor, Arduino IDE*

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El desarrollo de este componente consiste en crear un sistema que detecte llamas y emita alertas de manera eficiente utilizando la tecnología LoRa y la plataforma de mensajería *Telegram*. Se utiliza un sensor de llama LM393, que detecta la luz infrarroja emitida por las llamas [1]. Al detectar una llama, el sistema envía una alerta vía *Telegram* y también la muestra en la pantalla OLED de los módulos Heltec.

El objetivo principal es proporcionar información en tiempo real sobre la detección de llamas, permitiendo una rápida respuesta ante posibles incendios. Para ello se utiliza la placa de desarrollo Heltec Wifi LoRa V2, que integra un microcontrolador ESP32 y un módulo LoRa para transmitir datos a distancias largas con bajo consumo de energía [2].

Las alertas se proporcionan a través de *Telegram*, una red de mensajería instantánea acreditada y ampliamente utilizada, elegida por su velocidad y eficiencia. Las alertas incluyen información esencial sobre la detección de llamas, lo que permite a los usuarios tomar medidas tempranas para reducir los peligros de un posible incendio [3].

El sistema consta de numerosos componentes críticos. Primero, el sensor de llama LM393 detecta una llama y transmite una señal digital a la placa Heltec Wifi LoRa V2. Esta placa, programada con el entorno de programación *Arduino IDE*, interpreta las señales de los sensores y transmite mensajes de alerta a través del módulo LoRa [4]. Otro dispositivo Heltec Wifi LoRa V2 recibe el mensaje transmitido por LoRa y notifica al usuario a través de la API de *Telegram* [5].

Además de las notificaciones de *Telegram*, los mensajes se pueden mostrar en las pantallas OLED de los dispositivos Heltec LoRa V2. Este sistema examina los datos recopilados y envía notificaciones cuando se detectan llamas. La integración del sistema se realiza en etapas para garantizar la eficiencia y la confiabilidad [6].

El prototipo presenta dos puntos de control principales. Por un lado, los usuarios se comunican con el sistema mediante la aplicación de chat *Telegram* [3]. Por otro lado, pueden ver la información en las pantallas OLED de las placas de desarrollo. Este doble enfoque garantiza que los usuarios puedan administrar y monitorizar el sistema de forma cómoda y eficaz, adaptándose a sus gustos y necesidades.

1.1 Objetivo general

Implementar un prototipo de alerta por detección de llamas basado en LORA y *Telegram*.

1.2 Objetivos específicos

- Identificar los requerimientos para el diseño del prototipo.
- Seleccionar el *hardware* y *software* acorde a los requerimientos establecidos.
- Diseñar el prototipo de detección de llamas.
- Implementar el prototipo de detección de llamas.
- Realizar pruebas de funcionamiento del prototipo.

1.3 Alcance

El alcance del proyecto que abarca el prototipo de alerta por detección de llamas basado en LORA y *Telegram*, contiene las siguientes funcionalidades:

- Detección de llamas.
- Establecer la comunicación entre las tarjetas de desarrollo por medio de LORA.
- Envió de notificaciones de los datos obtenidos al usuario a través de *Telegram*.

1.4 Marco Teórico

Detección de llama

La detección de la presencia de llamas es un paso crítico en la alerta y prevención de incendios, y se emplea ampliamente tanto en entornos industriales como domésticos. Una llama se crea por una interacción química entre un gas combustible y oxígeno, lo que produce calor y luz. Se sabe que esta reacción genera luz en los espectros ultravioleta (UV) e infrarrojo (IR), lo que la hace detectable mediante sensores especializados [7].

Los sensores de llama están diseñados para captar esta radiación única y generar una señal digital cuando se detecta una llama. La identificación temprana de las llamas es fundamental para minimizar los daños a la propiedad y salvar vidas. Varios elementos, incluidas las condiciones climáticas y la presencia de materiales combustibles, podrían afectar la propagación del incendio, lo que requiere el uso de sistemas de detección eficaces y fiables [1].

Tecnología Lora

LoRa (gran alcance) es un sistema de comunicación inalámbrico que transmite datos a través de grandes distancias con un uso mínimo de energía. Esta tecnología es adecuada para aplicaciones de IoT (Internet de las cosas) que requieren monitoreo remoto y transmisión de datos a través de grandes distancias. LoRa utiliza modulación de espectro ensanchado, lo que lo hace resistente a las interferencias y adecuado para situaciones ruidosas [8].

Sensores de Llama

El sensor de llama LM393 es un dispositivo que detecta la presencia de llamas captando la luz infrarroja emitida por un incendio. Este sensor opera en el espectro infrarrojo, lo que le permite discriminar entre la luz de una llama y otras fuentes de luz, eliminando falsas alarmas. El LM393 envía una señal digital que puede ser procesada por un microcontrolador y activar sistemas de alerta [9].

Arduino IDE

Ar*duino IDE* es una plataforma para programar microcontroladores compatibles con Arduino. Este entorno es fácil de usar e incluye una variedad de bibliotecas y ejemplos para facilitar a los programadores a crear aplicaciones de IoT. En este proyecto, se utiliza el IDE de Arduino para programar la placa Heltec Wifi LoRa V2, que permite el procesamiento de datos del sensor de llama y la comunicación a través de LoRa [4].

Telegram y *Bot*s

Un *bot* en Telegram es una aplicación que interactúa con usuarios, enviando y recibiendo mensajes en tiempo real dentro de la plataforma de mensajería *Telegram* [3]. *Telegram* es una aplicación de mensajería en tiempo real que te permite crear *bot*s, que son programas automatizados que interactúan con otros usuarios. Los *bot*s de *Telegram* pueden enviar mensajes, advertencias y notificaciones, esto los hace apropiados para aplicaciones que requieren monitoreo y control remotos. En este proyecto, se utiliza un *bot* de *Telegram* para enviar notificaciones en tiempo real cuando se detecta una llama, lo que brinda un enfoque rápido y eficiente para advertir a los usuarios sobre posibles incendios [10].

2 METODOLOGÍA

En primer lugar, se examinó y evaluó los elementos tanto de *hardware* y *software* que fueron necesarios para el prototipo del sistema de alarma de detección de llamas, basado en la tecnología LoRa y la plataforma de mensajería *Telegram*. En lo que respecta al *hardware*, se seleccionaron cuidadosamente los componentes físicos, poniendo especial atención en el sensor de llama y el microcontrolador más adecuados para la aplicación.

Se eligió el sensor de llama LM393 debido a su capacidad de detectar la luz infrarroja emitida por las llamas y emitir una señal digital cuando se detecta una llama. Este sensor se seleccionó por su precisión y confiabilidad en la detección de incendios. La placa de desarrollo Heltec Wifi LoRa V2, que incorpora un módulo ESP32 y LoRa, proporcionó las capacidades de procesamiento de datos y comunicación inalámbrica necesarias [9] [6].

En cuanto al *software*, se programó la placa utilizando el IDE de Arduino. Este entorno de desarrollo se eligió por su facilidad de uso y su amplia selección de bibliotecas, que facilitan la programación de aplicaciones de IoT. La programación se centró en recopilar datos del sensor de llama y transmitirlos a través de la red LoRa [11].

Se optó por la plataforma de mensajería *Telegram* para la notificación de alarmas porque permite la creación de *bot*s automatizados como *botFather*. Se creó un *bot* de *Telegram* que obtiene datos del módulo Heltec y brinda alertas en tiempo real a los usuarios cuando se detecta una llama. Esta decisión se basó en la velocidad y eficiencia de *Telegram* en la entrega de mensajes [12].

Además, la pantalla OLED incorporada en el microprocesador Heltec Wifi LoRa V2 se utilizará para mostrar los datos pertinentes directamente en el dispositivo. Esta pantalla proporcionará estadísticas sobre la detección e intensidad de las llamas, categorizadas como llama ligera, llama moderada y llama alta (peligro) dependiendo de qué tan cerca se encuentre, brindando a los usuarios una referencia visual inmediata [2].

El diseño del prototipo comprendió un diagrama de conexión de los siguientes componentes: el sensor de llama LM393, la placa Heltec Wifi LoRa V2 transmisor, la conexión de red LoRa del receptor y la integración con la aplicación de mensajería *Telegram*. Se realizaron modificaciones específicas para garantizar que el *bot* de *Telegram* funcionara correctamente y que las alertas se proporcionaran a tiempo ya que esto es un evento en tiempo real [9].

Durante la implementación del prototipo, los módulos y sensores se combinaron con la aplicación de mensajería y la pantalla OLED. Se realizaron pruebas exhaustivas para garantizar que el sistema funcionara bien, sin fallas ni pérdidas en la conectividad LoRa. Estas pruebas estuvieron enfocadas a los objetivos del proyecto, con el objetivo de asegurar que el sistema funcionara correctamente y cumpliera con los requisitos definidos [13].

Finalmente, las pruebas operativas incluyeron escenarios simulados de detección de llamas para garantizar la funcionalidad del sistema de alerta. Estas pruebas fueron fundamentales para identificar y corregir fallas potenciales, garantizando que el sistema cumpliera sus objetivos de proporcionar alertas de detección de llamas tempranas confiables, mejorando la seguridad frente a eventos de incendios [7].

3 RESULTADOS

El prototipo del sistema de alerta de detección de llamas, que utiliza tecnología LoRa y la red de mensajería *Telegram*, proporciona una solución integral para la detección de incendios en sus primeras etapas. El sistema detecta llamas mediante el sensor de llama LM393 y envía la información a la placa Heltec Wifi LoRa V2 receptor.

La programación tanto en el Heltec trasmisor como en el Heltec receptor se la realizo en *Arduino IDE* ya que este entorno de desarrollo es compatible con dichas placas, así también incorpora una serie de bibliotecas que ayudará a realizar la comunicación Lora mucho más fácil así también poder visualizar los datos en la pantalla OLED de ambos Heltec.

Cuando se detecta una llama, se emite una alerta inmediata como lo es llama leve, llama moderada, llama alta dependiendo a que distancia se encuentre a través de *Telegram* y se muestra en la pantalla OLED de la placa Heltec receptor. Esta combinación de detección precisa, comunicación inalámbrica efectiva y alertas rápidas crea un sistema confiable para la prevención, control de incendios en tiempo real del siniestro.

Además, la adopción de la tecnología LoRa garantiza una amplia cobertura y una comunicación confiable, incluso en ubicaciones rurales o de difícil acceso. Es de vital importancia señalar que el prototipo está pensado para operar en la banda latinoamericana, especialmente en la frecuencia 915E6, asegurando compatibilidad y óptima funcionalidad en esta región.

3.1 Identificación de los requerimientos para el diseño del prototipo

Requisitos de *hardware*

En primer lugar, se identificaron los requisitos necesarios para implementar el prototipo de aviso de detección de llamas. El sistema incluye componentes de *hardware* como un sensor de llama eficiente y un microcontrolador que puede transmitir datos a largas distancias utilizando tecnología LoRa. Se eligió el Heltec Wifi LoRa V2 como microcontrolador principal debido a su capacidad para integrar LoRa y Wi-Fi, lo que proporciona una opción versátil para la comunicación inalámbrica. Este dispositivo funciona en la banda de 915 (MHz), ideal para América Latina, ya que garantiza una transmisión de datos estable y de largo alcance [14].

Se eligió el sensor de llama LM393 debido a su capacidad para detectar llamas a través de luz infrarroja. Este sensor distingue entre llamas y otras fuentes de luz, eliminando falsas alarmas, lo cual es crucial para la confiabilidad del sistema. Además, la placa Heltec Wifi LoRa V2 cuenta con un panel OLED que muestra información de detección en tiempo real, permitiendo un seguimiento directo en el sitio [15].

Requisitos de *Software*

El entorno de programación Arduino IDE fue seleccionado para el software debido a su facilidad de uso y compatibilidad con el dispositivo Heltec Wifi LoRa V2.

Este entorno permite que el microcontrolador procese datos del sensor de llama y administre la comunicación a través de LoRa [16]. La programación consistirá en configurar alertas automáticas enviadas a través de la plataforma de chat *Telegram*, utilizando un *bot* creado especialmente para este fin [10].

Integración de mensajería *Telegram*

La integración del servicio de mensajería instantánea *Telegram* es fundamental para las notificaciones de alertas en tiempo real. Al detectar una anomalía, el sensor de llama activará un *bot* de *Telegram* para enviar mensajes de alerta. Este acuerdo permite a los usuarios recibir notificaciones inmediatas en sus dispositivos móviles, proporcionando una reacción rápida ante cualquier incendio [12].

Simulación y diseño de la placa

El diseño y la simulación de la placa se realizaron utilizando el *software* EasyEDA. Este programa permitió la creación de un diagrama de conexión entre el sensor de llama LM393 y la placa Heltec Wifi LoRa V2, posibilitando la simulación del funcionamiento del sistema antes de su implementación real. EasyEDA facilitó la visualización y prueba del diseño del circuito, asegurando que las conexiones fueran correctas y optimizando el prototipo para su posterior desarrollo. La utilización de este *software* garantizó una evaluación exhaustiva y una preparación precisa del diseño antes de pasar a la fase de construcción física del sistema. [17].

Proceso de diseño y montaje

El diseño del prototipo comprende la conexión de todos los componentes antes mencionados. El sensor de llama LM393 se comunicará con la placa Heltec Wifi LoRa V2, que procesará los datos del sensor. En caso de detección de llama, enviará una señal de alerta a través del módulo receptor Heltec. El panel OLED del Heltec Wifi LoRa V2, tanto en el transmisor como en el receptor, mostrará la información de detección en tiempo real. La configuración del *bot* de *Telegram* permitirá a los usuarios recibir notificaciones en sus teléfonos móviles, ofreciendo una segunda capa de aviso y protección.

3.2 Selección del *hardware* y *software* acorde a los requerimientos establecidos

Selección del *Hardware*

Para la creación del sistema de detección de llamas y notificación de alertas requiere una cuidadosa selección de componentes de *hardware* que permitan la captación, procesamiento y transmisión de datos de manera eficiente y confiable. Los elementos principales incluyen el microcontrolador y el sensor de llama.

Microcontrolador

Para la implementación de este prototipo, se optó por utilizar el Heltec Wifi LoRa V2 debido a sus características avanzadas y su capacidad para soportar comunicaciones LoRa y Wi-Fi simultáneamente. Este dispositivo es ideal para proyectos de IoT que requieren una cobertura de largo alcance y un bajo consumo de energía. La placa Heltec Wifi LoRa V2 también incluye una pantalla OLED que facilita la visualización en tiempo real de la información recolectada por el sensor, sin necesidad de interfaces web adicionales [6].

En comparación con la Heltec CubeCell, aunque esta última ofrece beneficios similares en términos de consumo de energía, la Heltec Wifi LoRa V2 proporciona una integración más conveniente con Wi-Fi y un entorno de desarrollo más amplio gracias a su compatibilidad con el ESP32, como se visualiza en la Tabla 3.1 [18].

Tabla 3.1 Comparación entre Heltec Wifi LoRa V2 y Heltec CubeCell [18] [2]

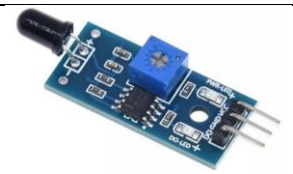
Característica	**Heltec Wifi LoRa V2**	**Heltec CubeCell**
Microcontrolador	ESP32	ASR605x
Memoria ROM	4 (MB)	128 (KB)
RAM	520 (KB)	16 (KB)
Conectividad	Wi-Fi 802.11 b/g/n, Bluetooth v4.2	LoRa
Frecuencia LoRa	915 (MHz) (Latinoamérica)	915 (MHz) (Latinoamérica)
Pantalla OLED	Integrada (0.96 pulgadas, 128x64)	No integrada
Consumo de Energía	Moderado	Muy bajo
Entradas y Salidas	9 (GPIOs), 1 (ADC), 1 (DAC)	8 (GPIOs), 1 (ADC)
Programación	*Arduino IDE*, PlatformIO	*Arduino IDE*, PlatformIO
Antena	Externa	Externa
Costo	Moderado	Bajo
Imagen		

El Heltec Wifi LoRa V2 se seleccionó por su capacidad de proporcionar conectividad Wi-Fi y LoRa, facilitando la integración con redes domésticas y la transmisión de datos a largas distancias, además de la facilidad de programación y visualización directa en su pantalla OLED [13].

Sensor de Llama

Para la detección de llamas, se compararon dos sensores principales: el LM393 y el KY-026. El sensor de llama LM393 se seleccionó debido a su alta sensibilidad a la luz infrarroja emitida por las llamas y su capacidad de filtrar otras fuentes de luz, reduciendo así las falsas alarmas [19]. Un ejemplo de una tabla se visualiza en la Tabla 3.2 .

Tabla 3.2 de comparación entre el sensor de llama LM393 y el KY-026 [19] [15]

Característica	**Sensor de Llama LM393**	**Sensor KY-026**
Tipo de Sensor	Fotodiodo infrarrojo	Fotodiodo infrarrojo + Termistor
Voltaje de Operación	3.3 (V) – 5 (V)	3.3 (V) – 5 (V)
Ángulo de Detección	60 grados	60 grados
Distancia de Detección	Hasta 1 metro	Hasta 1 metro
Salida	Digital (TTL) y analógica	Digital (TTL) y analógica
Sensibilidad Ajustable	Sí	No
Interferencia de Luz	Menor sensibilidad a luz ambiental	Mayor sensibilidad a luz ambiental
Indicador LED	Sí	Sí
Facilidad de Integración	Alta (comúnmente soportado en librerías Arduino)	Alta
Costo	Bajo	Bajo
Imagen		

El sensor de llama LM393 es el más adecuado para este proyecto debido a su alta sensibilidad que además es ajustable y su capacidad para filtrar otras fuentes de luz, lo que lo hace más confiable para la detección precisa de llamas en entornos variados [15].

Selección del *Software*

Para la programación del sistema, se utilizó el *Arduino IDE* por su facilidad de uso y su compatibilidad con la placa Heltec Wifi LoRa V2. Este entorno permite desarrollar el código necesario para el procesamiento de datos del sensor de llama y la transmisión de alertas a través de LoRa [16].

Figura 3.1 Logo *Arduino IDE* [20]

Además, se utilizó el *software* EasyEDA para el diseño y simulación del circuito electrónico. EasyEDA permitió crear un esquema detallado de la conexión entre el sensor de llama LM393 y la placa Heltec Wifi LoRa V2, asegurando la correcta implementación del prototipo antes de su ensamblaje físico [17].

Figura 3.2 Logo EasyEDA [21]

3.3 Diseñar el prototipo de detección de llamas

Esquema general

La Figura 3.1 muestra el diseño de conexión para el prototipo de detección de llama. Este diseño asegura que el dispositivo emisor Heltec Wifi LoRa V2 se comunique de manera eficiente con el receptor Heltec Wifi LoRa V2, permitiendo que los datos se transmitan y se muestren correctamente en la aplicación de mensajería *Telegram*, así como en la pantalla OLED incorporada en ambos dispositivos. Este método de prototipo centralizado garantiza el cumplimiento de los objetivos específicos del proyecto.

Figura 3.1 Esquema del prototipo

El sensor de llama LM393 junto con un emisor heltec, detecta las llamas mediante el reconocimiento de la luz infrarroja generada por el fuego. Una vez detectada una llama, la señal se envía a la placa Heltec Wifi LoRa V2. Esta placa, que incluye un módulo de comunicación LoRa y una pantalla OLED, interpreta la señal y transmite los datos al receptor a través de la red LoRa WAN.

El receptor, también basado en la placa Heltec Wifi LoRa V2, recibe la señal transmitida y la muestra en su pantalla OLED. Adicionalmente, este dispositivo está diseñado para enviar notificaciones al usuario a través de la aplicación *Telegram*. La integración con *Telegram* se logra mediante la programación en el entorno *Arduino IDE* y el uso de la biblioteca correspondiente para establecer la conexión entre la placa y el servidor de *Telegram*.

Un ejemplo detallado de esta implementación se visualiza en la Figura 3.2

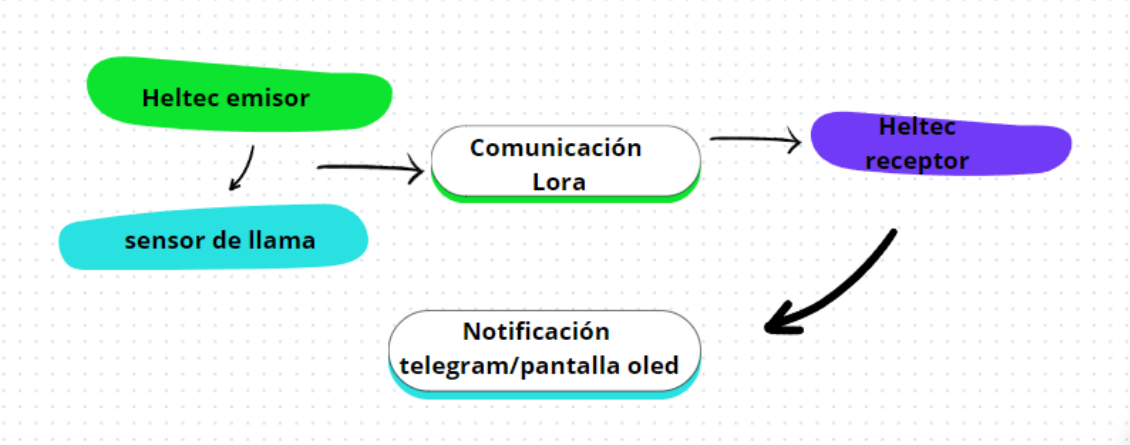


Figura 3.2 Descripción del funcionamiento del prototipo.

El dispositivo funciona de forma autónoma y continua, detectando siempre la presencia de llamas en el ambiente. Cuando se detecta una llama, el sistema no solo actualiza la pantalla OLED del receptor, sino que también proporciona un aviso instantáneo al usuario a través de *Telegram*. Esta advertencia contiene información crucial sobre la detección, lo que permite al usuario tomar medidas rápidas y efectivas.

Creación del *Bot* en *Telegram*

Para crear un *bot* en *Telegram*, se utilizan dos herramientas principales: *BotFather* e ID *Bot*. Un *bot* en *Telegram* es una cuenta automatizada que puede interactuar con los usuarios mediante mensajes y comandos sin necesidad de un número de teléfono. *BotFather* proporciona el token necesario, mientras que ID *Bot* ofrece la ID de chat para configurar y operar el *bot*. A continuación, se presentan instrucciones detalladas para desarrollar y personalizar un *bot* utilizando estas herramientas. [12].

Abrir *Telegram* y buscar *BotFather*

Para comenzar, abra la aplicación *Telegram* y escriba "*BotFather*" en el campo de búsqueda. Al seleccionar el *bot* oficial de *Telegram*, *BotFather*, el usuario puede comenzar el proceso de creación del *bot*, como se aprecia en la Figura 3.3 .

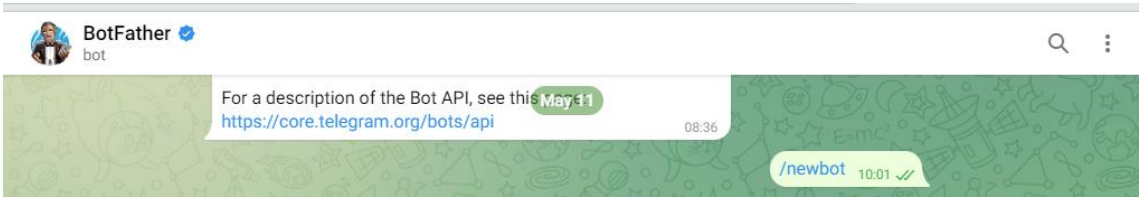


Figura 3.3 Activación del *Bot* en *Telegram*

Al interactuar con *BotFather*, el usuario debe emitir el comando /new*bot*, como se visualiza en la Figura 3.3 , para iniciar la creación del *bot*. *BotFather* nos permite incluir la asignación de un nombre de usuario único para el *bot*, en este caso, "Detector_incendio_*bot* ". Al finalizar este proceso, *BotFather* proporcionará un token de autenticación, visible en la Figura 3.4 . Este token es crucial, ya que permite la operación del *bot* desde el código. Su formato es similar a 123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11.

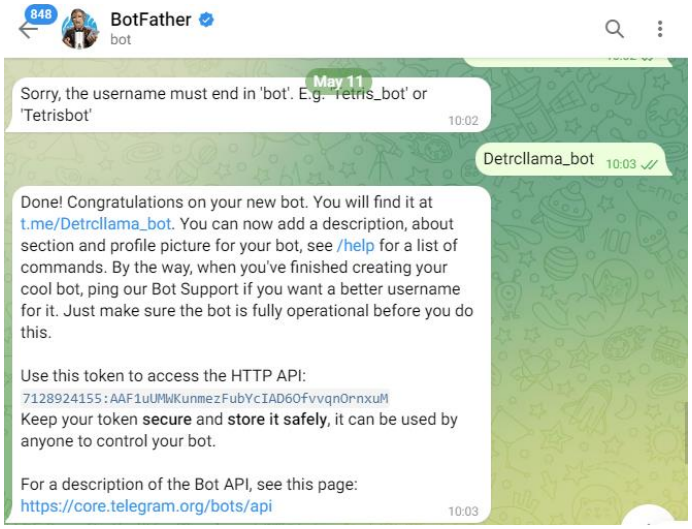


Figura 3.4 Nombre del *Bot* y obtención del token.

Obtener el *chat ID* con ID *Bot*

A continuación el siguiente paso es adquirir el ID del chat para que el *bot* pueda enviar mensajes. Para lograr esto, el usuario debe buscar "ID *Bot*" en *Telegram* e iniciar una conversación con el *bot* [22]. Durante el chat con ID *Bot*, el usuario debe ingresar el comando /start. ID *Bot* devolverá el ID de chat del usuario, que es un identificador único que el *bot* utilizará para enviar mensajes a ese usuario. Como se visualiza en la Figura 3.5.

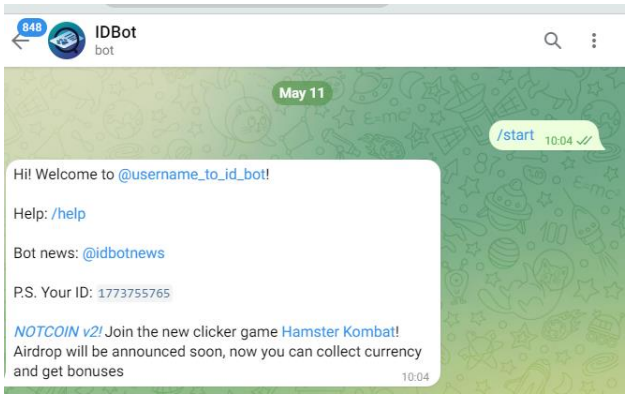


Figura 3.5 Obtención *Chat ID*.

Integración del *Bot* en el Código

Con el token y el ID del chat en mano, el usuario puede incorporarlos al código del receptor para configurar y administrar el *bot*.

El código comienza agregando las bibliotecas relevantes y definiendo las constantes *BOT* token y CHAT_ID con valores recibidos previamente de *BotFather* e ID *Bot*. Luego, se construye un objeto WiFiClientSecure para establecer una conexión segura con los

servidores de *Telegram* y el *bot* se configura con el token y el cliente seguro como se visualiza en la Figura 3.6. Se incluye la librería Universal*TelegramBot*.h que es necesaria para manejar el *bot* de *Telegram*.

```
#include <WiFi.h> // Librería para manejar la conexión WiFi
#include <WiFiClientSecure.h> // Librería para manejar la conexión segura
#include <UniversalTelegramBot.h> // Librería para manejar el bot de Telegram
#include <ArduinoJson.h> // Librería para manejar JSON
```

Figura 3.6 Librerías necesarias para la comunicación con *Telegram*.

Se definen las constantes *BOT*token y CHAT_ID con los valores proporcionados por *BotFather* e ID*Bot* respectivamente, como se visualiza en la Figura 3..

```
// Inicializa el bot de Telegram
#define BOTtoken "7128924155:AAF1uUMWKunmezFubYcIAD6OfvvqnOrnxuM" // Token del Bot (Obtenido del Botfather)

// Usa @myidbot para encontrar el chat ID de un individuo o grupo
// También necesitas hacer clic en "start" en un bot antes de que pueda
// enviarte mensajes
#define CHAT_ID "1773755765"
```

Figura 3.7 Constantes *BOT*token y CHAT_ID

Se crea un objeto WiFiClientSecure que se utiliza para establecer una conexión segura con los servidores de *Telegram*. Se inicializa el *bot* con el token y el cliente seguro como se visualiza en la Figura 3.8 .

```
WiFiClientSecure client; // Cliente seguro para manejar la conexión
UniversalTelegramBot bot(BOTtoken, client); // Inicializa el bot con el token y el cliente seguro
```

Figura 3.8 Arranque wificlient, token, client

El código establece la conexión Wifi usando el SSID y la contraseña de la red. Tener acceso a internet es esencial para que el bot interactúe con los servidores de Telegram y envíe mensajes, tal como se muestra en la Figura 3.9
La conexión Wifi se configura utilizando el SSID y la contraseña de la red, y el bot no puede operar sin acceso a internet.

```
const char* ssid = "Xiaomi"; // Nombre de la red WiFi
const char* password = "ojitos12"; // Contraseña de la red WiFi
```

Figura 3.9 Configuración de la red WIFI

Una vez que la conexión Wifi está establecida, el *bot* puede enviar mensajes de inicio y alertas al *chat ID* definido. Dependiendo de las condiciones especificadas en el loop

principal del código (como la detección de fuego en este caso), se envían diferentes mensajes a través de *Telegram* como se visualiza en la Figura 3.10 .

Figura 3.10 Comunicación con el chat de *telegram*

Comunicación Lora entre las placas de heltec Wifi LoRa v2

El enlace "https://dl.espressif.com/dl/package_esp32_index.json" [6]. Facilita la instalación del paquete Arduino-ESP32 en diversas plataformas y sistemas operativos. Este recurso es útil para configurar el entorno de desarrollo y programar el ESP32 con el software de Arduino. El procedimiento de instalación desde el IDE de Arduino incluye copiar el enlace en las preferencias, instalar la plataforma ESP32 y seleccionar la placa, como es el caso de la Heltec Wifi LoRa v2, tal como se visualiza en la Figura 3.11 .

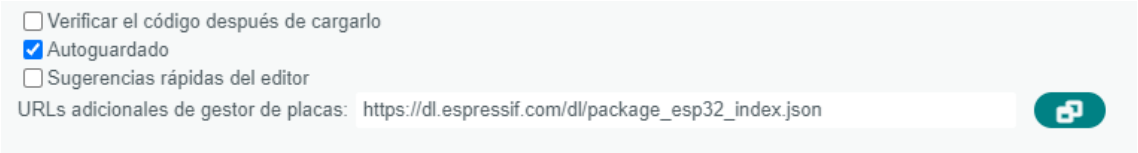


Figura 3.11 Instalación paquete Arduino-ESP32

Ademas, se debe instalar el gestor de placa esp32 de Espressif Systems 2.0.11 para el correcto funcionamiento como se visualiza en la Figura 3.12 .

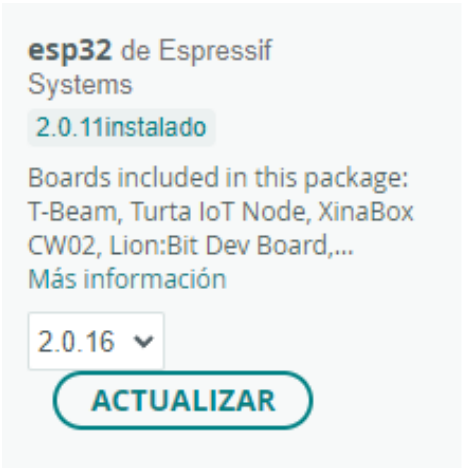


Figura 3.12 Espressif Systems 2.0.11

También se deben instalar las diferentes librerias para que la placa de desarrollo heltec Wifi LoRa v2 no tenga ningun inconveniente tanto en la comunicación arduinoJson ,la librería heltec ESP32 DevBoards y la comunicación con la aplicación *telegram* como es la biblioteca universal*TelegramBot* al momento de verificar el codigo y subirlo a la placa este todo correcto y funcional como se visualiza en la Figura 3.13 .

Figura 3.13 ArduinoJson, heltec ESP32 DevBoards, universal*TelegramBot*

Trasmisor

Primero, se debe establecer las librerías correctas en el trasmisor como son include "heltec.h", include "images.h".

En el bucle principal, se lee el valor del sensor de llama. Luego, se determina el nivel de detección de fuego y se actualiza la pantalla OLED. Finalmente, se envía el valor del sensor a través de LoRa. Además, ambos dispositivos usan la misma banda de frecuencia definida por #define BAND 915E6 como se visualiza en la Figura 3.14 .

La inicialización de LoRa se realiza con Heltec.begin(...), habilitando la pantalla, LoRa, y el puerto serial. Se envía el código del trasmisor para la respectiva comunicación como se visualiza en la Figura 3.15 .

```
#define flama_pin A0 // Define el pin analógico A0 como flama_pin
int ValorSensor = 0; // Inicializa la variable ValorSensor en 0
#define BAND 915E6  // Define la banda de frecuencia LoRa a 915 MHz
```

Figura 3.14 Variables y frecuencia a utilizar.

```
LoRa.beginPacket(); // Inicia un nuevo paquete LoRa
LoRa.setTxPower(14, RF_PACONFIG_PASELECT_PABOOST); // Establece la potencia de transmisión LoRa
LoRa.print(ValorSensor); // Envía el valor del sensor
LoRa.endPacket(); // Finaliza el paquete LoRa
```

Figura 3.15 Comunicación lora a trasmitir.

Receptor

Para el receptor, es necesario configurar las bibliotecas como "heltec.h", "images.h" y <ArduinoJson.h>, como se indica en la Figura 3.16 . Ambos dispositivos utilizan la misma banda de frecuencia definida por #define BAND 915E6, como se visualiza en la Figura 3.17 .

```
#include "heltec.h" // Librería específica de Heltec para manejar el hardware
#include "images.h" // Librería para manejar las imágenes en la pantalla OLED
#include <WiFi.h> // Librería para manejar la conexión WiFi
#include <WiFiClientSecure.h> // Librería para manejar la conexión segura
#include <UniversalTelegramBot.h> // Librería para manejar el bot de Telegram
#include <ArduinoJson.h> // Librería para manejar JSON
```

Figura 3.16 Librerías receptor.

```
#define BAND 915E6 // Configura la banda para la comunicación LoRa, por ejemplo, 868E6, 915E6
```

Figura 3.17 Banda de frecuencia Receptor.

Para finalizar se envía el código de recepción para la correcta comunicación de las dos placas de desarrollo heltec Wifi LoRa v2 como se visualiza en la Figura 3.18 .

```
//LoRa.onReceive(cbk);
LoRa.receive(); // Pone el módulo LoRa en modo de recepción
WiFi.mode(WIFI_STA); // Configura el modo WiFi como estación
WiFi.begin(ssid, password); // Conecta al WiFi con el SSID y contraseña dados
client.setCACert(TELEGRAM_CERTIFICATE_ROOT); // Añade el certificado raíz para api.telegram.org
```

Figura 3.18 Comunicación Lora recibida al receptor.

Elaboración de la placa

En la Figura 3.19 se utilizó EasyEDA para diseñar un circuito que integrara la conexión del Heltec Wifi LoRa V2 y el sensor de llama. EasyEDA que es una herramienta electrónica que permite la creación de esquemáticos y la elaboración de PCB (placas de circuito impreso). Al utilizar EasyEDA, se garantiza una mayor precisión en la simulación y la corrección de errores previos a la fabricación de la placa. [21]

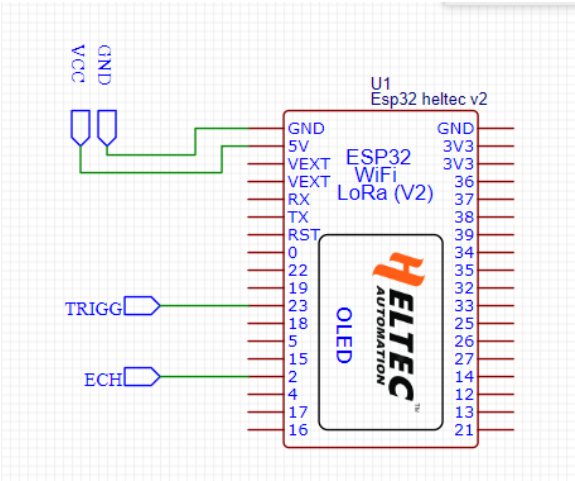


Figura 3.19 Simulación conexión sensor con la placa.

El diseño del circuito fue meticulosamente planificado para incluir los componentes necesarios, asegurando suficiente espacio para el mantenimiento de cada módulo. Además, se tomó en cuenta el diseño de las pistas para evitar que se toquen, lo que podría causar cortocircuitos y otros problemas de funcionamiento.

Justificación de Intervalos Operativos Establecidos para el Prototipo

Para este proyecto, se utiliza Heltec Wifi LoRa V2 para la detección de llamas, lo que requiere rangos de trabajo precisos para obtener mejores resultados. El sensor de llama detecta la radiación infrarroja emitida por el fuego y envía una señal al microcontrolador, que utiliza esta información para tomar decisiones en tiempo real [15].

El Heltec Wifi LoRa V2 y el sensor de llama están eleborado para funcionar dentro de un rango de voltaje y corriente definido, lo que garantiza un rendimiento constante y confiable. Establecer estos rangos operativos es fundamental para garantizar que el sistema funcione correctamente en una variedad de circunstancias ambientales [23].

Rango de Funcionamiento del Sensor de Llama

El sensor de llama utilizado en este proyecto detecta radiación infrarroja en un rango de 760 nm a 1100 nm. La sensibilidad del sensor se ajusta para detectar llamas a una distancia de hasta 100 cm, lo que permite una rápida respuesta ante la presencia de fuego. Este rango de funcionamiento es adecuado para aplicaciones de seguridad y monitoreo de incendios [15].

Justificación de Consumo de Energía

Para calcular el consumo de energía del sistema, se consideraron los rangos de voltaje y corriente de cada uno de los componentes principales con carga significativa. Los valores en la Tabla 3.3 .

Tabla 3.3 Consumo de energía del sistema.

Componentes Electrónicos	**Voltaje (V)**	**Corriente (mA)**
Heltec Wifi LoRa V2	3.3 - 5	70
Sensor de llama	5	15

Cálculo del Consumo de Corriente Total del Prototipo

Para obtener la corriente de operación del prototipo, se utiliza la siguiente ecuación:

$$ISistema = IHeltec + ISensor$$

Ecuación 3.1 Corriente Total del Prototipo [24]

Donde:

$$IHeltec: 70\ (mA)\ (corriente\ del\ módulo\ Heltec\ WiFi\ LoRa\ V2)$$

$$ISensor: 15\ (mA)\ (corriente\ del\ sensor\ de\ llama)$$

Por lo tanto:

$$ISistema = 70mA + 15mA$$

$$ISistema = 85\,mA$$

Dado que el Heltec Wifi LoRa V2 cuenta con una entrada micro USB, y considerando que el proyecto se centra en la detección de llama, se decidió alimentar tanto el receptor como el transmisor utilizando un cargador de celular con un cable micro USB conectado a un tomacorriente. La fuente de alimentación que se utilizará será un cargador de celular que debe proporcionar 5 (V) y ser capaz de abastecer al menos 85 (mA) para asegurar un funcionamiento continuo y estable del sistema [13].

Cálculo de Parámetros de Comunicación LoRa

RSSI es una medida de la potencia de la señal recibida, expresada en decibelios-milivatios (dBm). Valores más bajos (más negativos) indican una señal más débil. Por ejemplo, un RSSI de -123 (dBm) significa que la señal recibida es muy débil. Los valores típicos de RSSI en comunicaciones LoRa suelen estar en el rango de -120 (dBm) a -30 (dBm). [25].

SNR (Relación Señal-Ruido) indica la diferencia entre la potencia de la señal y el nivel de ruido de fondo, expresado en decibelios (dB). Un SNR positivo indica que la señal es más fuerte que el ruido, lo cual es favorable en términos de calidad de la señal. [25]

$$SNR\ (dB) = 10log\ (Potencia\ de\ la\ señal / Potencia\ del\ ruido\ )$$

Ecuación 3.2 Factor de ruido

Para mejorar el RSSI, se pueden tomar medidas como aumentar la potencia de transmisión del transmisor, reducir la distancia entre el transmisor y el receptor, y optimizar la antena utilizando antenas de mayor ganancia o mejor posicionamiento. Estas acciones ayudan a mejorar la potencia de la señal recibida aplicando la Ecuación 3.2 .

Para optimizar el SNR, es importante reducir las interferencias minimizando las fuentes de ruido cercanas y mejorar el entorno implementando el sistema en un entorno con menos obstrucciones [25].

Diagrama de flujo

El siguiente diagrama de flujo del transmisor se basa en la Figura 3.20 . El sistema comprueba el funcionamiento a Internet y, si es válida, comienza a obtener y calcular valores. Los valores se envían a la aplicación *telegram* y al panel OLED de la placa de desarrollo tanto para el remitente como para el receptor.

Si se detecta un incendio, se comunica vía LoRa. La operación se realiza constantemente con un retraso de un segundo entre mediciones para garantizar que el sistema de detección de llamas esté actualizado y proporcione información precisa para mayor seguridad. En la Figura 3.21 se visualiza el diagrama de flujo del receptor en donde lo más relevante es la mensajería que se va a enviar por medio de *telegram*.

Figura 3.20 Diagrama de flujo trasmisor

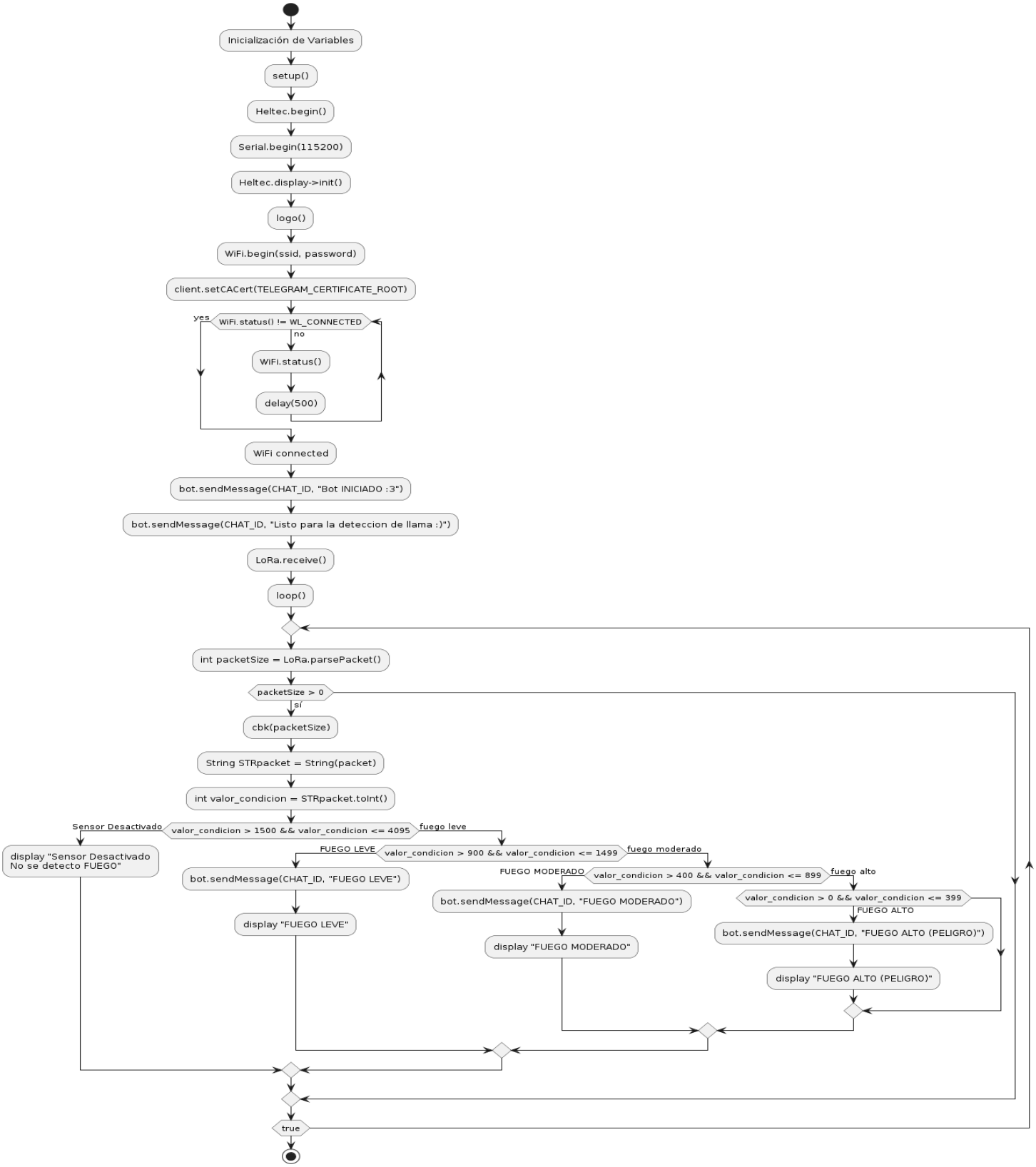


Figura 3.21 Diagrama de flujo receptor

3.4 Implementación del prototipo de detección de llamas

Fabricación de la placa

El *software* EasyEDA se utiliza para fabricar la placa PCB del sistema de detección de llamas. Este *software* es adecuado para crear y exportar el diagrama en formato PDF, como se ilustra en la Figura 3.22 El diseño incluye las especificaciones exactas del módulo Heltec Wifi LoRa v2, así como los puntos de conexión para componentes como el sensor de llama.

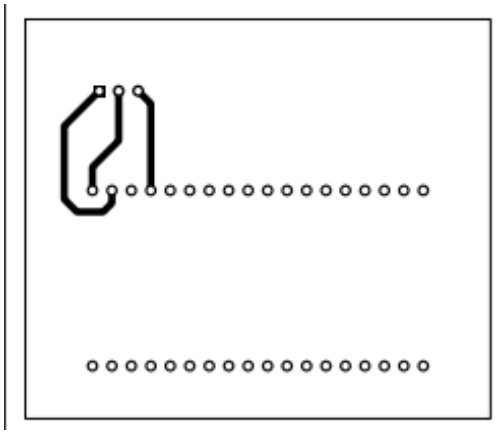


Figura 3.22 Diagrama de la PCB

La placa PCB del sistema de detección de llamas requiere ácido férrico, una placa de baquelita y estaño [26]. Estos componentes permiten la fabricación del tablero, que anteriormente se creó mediante el método de transferencia térmica. Este procedimiento garantiza que el diseño de la PCB se transfiera correctamente a la superficie de cobre de la baquelita. La Figura 3.23 muestra la placa de forma frontal.

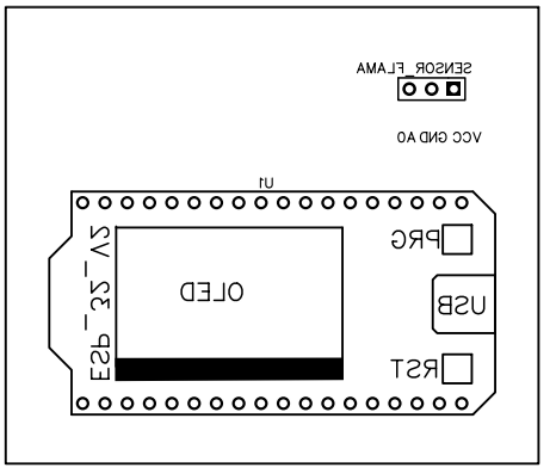


Figura 3.23 Elaboración de la PCB final

Montaje de los componentes de la placa

En la Figura 3.24 se presenta el prototipo montado con todos los componentes electrónicos soldados a la placa PCB.

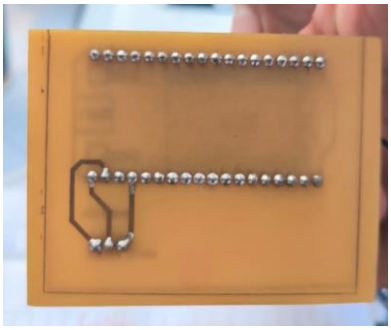


Figura 3.24 Soldadura de la placa PCB

Para proteger el módulo trasmisor se realizó una caja de vidrio con su respectiva tapa para la protección del mismo a si también unas guías de plástico para la antena y sensor de llama, como se visualiza en la Figura 3.25 .

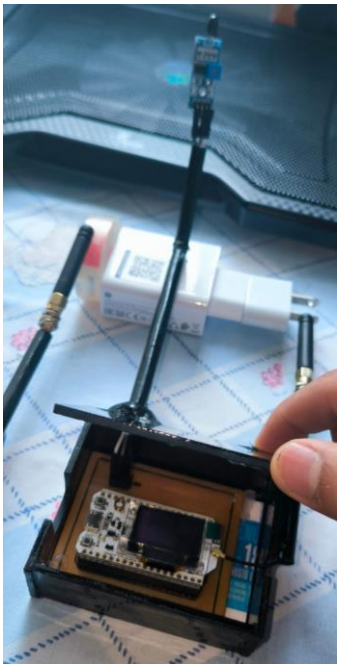


Figura 3.25 Caja de vidrio para el trasmisor

En una segunda caja de vidrio, se encuentra el módulo receptor, también protegido por guías de plástico y sellado con silicona caliente para evitar interferencias y proteger los componentes de polvo y otros contaminantes. La caja se cierra con su respectiva tapa con silicona donde se visualiza la pantalla OLED de la placa de desarrollo.

La Figura 3.26 visualiza la distribución y montaje de los componentes electrónicos en la caja de vidrio para el receptor.

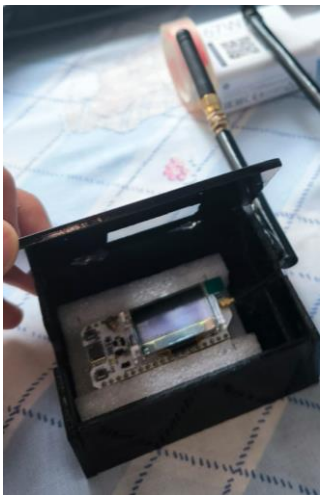


Figura 3.26 Caja de vidrio para el receptor

En la Figura 3.27 se aprecia el prototipo final montado y listo para su operación tanto el trasmisor como el receptor.

Figura 3.27 Prototipo listo trasmisor y receptor

Programación del Módulo Heltec Wifi LoRa v2 (Trasmisor)

El *software* comprende la biblioteca Heltec, necesaria para controlar la pantalla del microprocesador y el módulo de comunicación LoRa, así como la biblioteca de imágenes, que contiene el logotipo que se mostrará en la pantalla.

Defina el pin analógico A0 como entrada del sensor de llama, luego se declara una variable 'SensorValue' para registrar las lecturas del sensor, comenzando desde cero. Además, configura la banda de frecuencia LoRa a 915 (MHz), que prevalece en América del Norte, para permitir una comunicación inalámbrica exitosa, como se visualiza en la Figura 3.28 .

```
#include "heltec.h" // Incluye la librería Heltec
#include "images.h" // Incluye la librería de imágenes
#define flama_pin A0 // Define el pin analógico A0 como flama_pin
int ValorSensor = 0; // Inicializa la variable ValorSensor en 0
#define BAND 915E6  // Define la banda de frecuencia LoRa a 915 MHz
```

Figura 3.28 Librerías y definiciones en el módulo trasmisor.

El programa define varias variables globales para facilitar el funcionamiento del sistema. Inicializa un contador en cero, que podría usarse para llevar la cuenta de eventos o iteraciones. También incluye una cadena de texto rssi para almacenar la intensidad de la señal recibida, comenzando con un valor por defecto de "RSSI ". Otra cadena packSize se utiliza para registrar el tamaño de los paquetes de datos Finalmente, una cadena packet se reserva para guardar los datos de los paquetes de datos que se envían o reciben, aunque no tiene un valor inicial asignado, como se visualizan en la Figura 3.29 .

```
unsigned int counter = 0; // Inicializa el contador en 0
String rssi = "RSSI --"; // Inicializa la variable rssi
String packSize = "--"; // Inicializa la variable packSize
String packet; // Declara la variable packet
```

Figura 3.29 Variables globales.

La función logo se encarga de limpiar la pantalla OLED y dibujar un logo en ella. Primero, la pantalla se limpia completamente como se muestra en la Figura 3.30 Luego, el logo se dibuja en las coordenadas específicas (0, 5). Finalmente actualiza la pantalla para mostrar el logo recién dibujado.

```
void logo() {
  Heltec.display->clear(); // Limpia la pantalla
  Heltec.display->drawXbm(0, 5, logo_width, logo_height, logo_bits); // Dibuja el logo en la pantalla
  Heltec.display->display(); // Muestra el contenido en la pantalla
}
```

Figura 3.30 Función logo.

En la función setup, comienza inicializando la comunicación serial a 115200 baudios. Luego, inicializa el módulo Heltec, habilitando la pantalla OLED, el LoRa y otras características, con la frecuencia especificada. Se configura el pin del sensor de llama como entrada. La pantalla OLED se inicializa.

A continuación, se llama a la función logo para mostrar el logo en la pantalla y se espera 1.5 segundos. Después, la pantalla se limpia y se muestran varios mensajes: "SISTEMA INICIALIZADO" en la parte superior, seguido de "Sistema Indicador" y "detector de Fuego" en diferentes posiciones. Estos mensajes se muestran en la pantalla y finalmente, el sistema espera 10 segundos antes de proceder al siguiente paso, como se visualiza en la Figura 3.31 .

```
21
22  void setup() {
23    Serial.begin(115200); // Inicializa la comunicación serial a 115200 bps
24    Heltec.begin(true, true, true, true, BAND); // Inicializa la pantalla y el módulo LoRa con la banda especificada
25    pinMode(flama_pin, INPUT); // Configura el pin de la flama como entrada
26    Heltec.display->init(); // Inicializa la pantalla OLED
27    Heltec.display->flipScreenVertically(); // Voltea la pantalla verticalmente
28    Heltec.display->setFont(ArialMT_Plain_16); // Establece la fuente de texto
29    logo(); // Llama a la función logo para mostrarlo en la pantalla
30    delay(1500); // Espera 1.5 segundos
31    Heltec.display->clear(); // Limpia la pantalla
32    Heltec.display->drawString(0, 0, "SISTEMA INICIALIZADO"); // Muestra el mensaje en la pantalla
33    Heltec.display->drawString(0, 25, "Sistema Indicador "); // Muestra el mensaje en la pantalla
34    Heltec.display->drawString(0, 40, "detector de Fuego"); // Muestra el mensaje en la pantalla
35    Heltec.display->display(); // Muestra el contenido en la pantalla
36    delay(10000); // Espera 10 segundos
37  }
```

Figura 3.31 Función setup

La función loop se ejecuta repetidamente y contiene las instrucciones principales de la lectura y los datos procesados del sensor.

Primero, el valor analógico del sensor de llama se lee y se guarda en la variable. Luego, este valor se imprime en la consola serial permitiendo monitorear en tiempo real las lecturas del sensor, como se visualiza en la Figura 3.32 .

```
41
42  void loop() {
43    ValorSensor = analogRead(flama_pin); // Lee el valor analógico del pin de la flama
44    Serial.println(ValorSensor); // Imprime el valor del sensor en la consola
```

Figura 3.32 Función loop

Los valores entre 1500 y 4095, así como entre 901 y 1499, se determinan en función de la señal analógica generada por el sensor de llama LM393, que mide la intensidad de la luz infrarroja emitida por una llama. Estos valores se convierten en señales digitales a través del conversor analógico-digital (ADC) del microcontrolador. La variación en estos valores indica diferentes niveles de intensidad de luz infrarroja, lo que permite al sistema diferenciar entre la presencia de fuego y la ausencia del mismo. [19]

En el rango de 1500 a 4095, la baja intensidad de luz infrarroja detectada sugiere la ausencia de fuego, lo que hace que el sistema apague el LED indicador y muestre un mensaje de "sin fuego" en la pantalla. Por otro lado, cuando los valores están entre 901 y 1499, se detecta una mayor intensidad de luz, interpretándose como la presencia de

un fuego leve. En este caso, el sistema enciende el LED y muestra un mensaje de advertencia en la pantalla. Estos umbrales se establecen mediante pruebas experimentales y calibración para asegurar la precisión del sistema en su entorno específico. [15]

Para valores entre 401 y 899, se considera fuego moderado, y para valores entre 1 y 399, se considera fuego alto, lo que indica un peligro más severo. En estos dos últimos casos, se activa el LED y se muestra un mensaje en la pantalla para alertar sobre la situación, como se muestra en la.Figura 3.33 .

```
48  if (ValorSensor > 1500 && ValorSensor <= 4095) {
49    Serial.println("---------------> Sensor Desactivado <---------------"); // Imprime el mensaje en la consola
50    Serial.println("....... No se detecto fuego ....... "); // Imprime el mensaje en la consola
51    digitalWrite(LED, LOW); // Apaga el LED
52    Heltec.display->clear(); // Limpia la pantalla
53    Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda
54    Heltec.display->setFont(ArialMT_Plain_16); // Establece la fuente de texto
55    Heltec.display->drawString(0, 0, "Enviando Datos: "); // Muestra el mensaje en la pantalla
56    Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda
57    Heltec.display->setFont(ArialMT_Plain_10); // Establece la fuente de texto
58    Heltec.display->drawString(0, 25, "Sensor desactivado"); // Muestra el mensaje en la pantalla
59    Heltec.display->drawString(0, 40, "No se detecto FUEGO"); // Muestra el mensaje en la pantalla
60    Heltec.display->display(); // Muestra el contenido en la pantalla
61  } else if (ValorSensor > 900 && ValorSensor < 1499) {
62    Serial.println("---------------> Sensor Activado <---------------"); // Imprime el mensaje en la consola
63    Serial.println("....... Se detecta fuego leve ....... "); // Imprime el mensaje en la consola
64    digitalWrite(LED, HIGH); // Enciende el LED
65    Heltec.display->clear(); // Limpia la pantalla
66    Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda
67    Heltec.display->setFont(ArialMT_Plain_16); // Establece la fuente de texto
68    Heltec.display->drawString(0, 0, "Enviando Datos: "); // Muestra el mensaje en la pantalla
69    Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda
70    Heltec.display->setFont(ArialMT_Plain_10); // Establece la fuente de texto
71    Heltec.display->drawString(0, 25, "Sensor Activado"); // Muestra el mensaje en la pantalla
72    Heltec.display->drawString(0, 40, " FUEGO LEVE"); // Muestra el mensaje en la pantalla
73    Heltec.display->display(); // Muestra el contenido en la pantalla
```

Figura 3.33 Condicionales para evaluar el valor del sensor.

El código utiliza el módulo LoRa para enviar datos a través de la red. Primero, inicia un nuevo paquete de transmisión LoRa.

Luego, configura la potencia de transmisión a 14dBm usando el PA_BOOST. A continuación, envía el valor del sensor a través de LoRa. Finalmente, finaliza el paquete de transmisión. Esto permite enviar información sobre la detección de fuego a otros dispositivos conectados a la red LoRa, como se visualiza en la Figura 3.34 .

```
103   LoRa.beginPacket(); // Inicia un nuevo paquete LoRa
104   LoRa.setTxPower(14, RF_PACONFIG_PASELECT_PABOOST); // Establece la potencia de transmisión LoRa
105   LoRa.print(ValorSensor); // Envía el valor del sensor
106   LoRa.endPacket(); // Finaliza el paquete LoRa
107 }
```

Figura 3.34 Envíos de datos con Lora

Programación del Módulo Heltec Wifi LoRa v2 (Receptor)

Aquí se incluyen las librerías necesarias para el proyecto. "heltec.h" es una librería específica para manejar el *hardware* de la placa utilizada, mientras que "images.h" contiene definiciones de imágenes para la pantalla OLED. Las otras librerías son para manejar la conexión Wifi, la seguridad de la conexión, el *bot* de *Telegram* y para trabajar con JSON. Como se visualiza en la Figura 3.35 .

```
#include "heltec.h" // Librería específica de Heltec para manejar el hardware
#include "images.h" // Librería para manejar las imágenes en la pantalla OLED
#include <WiFi.h> // Librería para manejar la conexión WiFi
#include <WiFiClientSecure.h> // Librería para manejar la conexión segura
#include <UniversalTelegramBot.h> // Librería para manejar el bot de Telegram
#include <ArduinoJson.h> // Librería para manejar JSON
```

Figura 3.35 Librerías del módulo receptor.

Se definen varias variables y constantes utilizadas en el programa. Para manejar el tiempo entre ejecuciones de cierto código. *ssid* y *password* son las credenciales de la red Wifi. *BOT*token y CHAT_ID son utilizados para la configuración del *bot* de *Telegram*. client y *bot* son objetos para manejar la conexión Wifi y el *bot* de *Telegram*. Como se muestra en la Figura 3.36 , BAND es la banda de comunicación LoRa utilizada. rssi, packSize y packet son utilizadas para almacenar información recibida.

```
unsigned long previousMillis = 0; // Variable para almacenar el tiempo anterior
const long interval = 1500; // Intervalo de tiempo en milisegundos
const char* ssid = "Xiaomi"; // Nombre de la red WiFi
const char* password = "ojitos12"; // Contraseña de la red WiFi
// Inicializa el bot de Telegram
#define BOTtoken "7128924155:AAF1uUMWKunmezFubYcIAD6OfvvqnOrnxuM" // Token del Bot (Obtenido del Botfather)
// También necesitas hacer clic en "start" en un bot antes de que pueda
// enviarte mensajes
#define CHAT_ID "1773755765"
WiFiClientSecure client; // Cliente seguro para manejar la conexión
UniversalTelegramBot bot(BOTtoken, client); // Inicializa el bot con el token y el cliente seguro
#define BAND 915E6 // Configura la banda para la comunicación LoRa, por ejemplo, 868E6, 915E6
String rssi = "RSSI --"; // Variable para almacenar el valor de RSSI
String packSize = "--"; // Variable para almacenar el tamaño del paquete
String packet; // Variable para almacenar el contenido del paquete
```

Figura 3.36 Declaración de variables y constantes.

En la función principal, se monitorea constantemente la recepción de datos LoRa y se procesan los datos recibidos para determinar el estado del sensor de detección de fuego como se muestra en la Figura 3.37 . Dependiendo del valor del sensor, se envía un mensaje a través del *bot* de *Telegram* y se actualiza la pantalla OLED para reflejar el estado actual del sensor.

```
} else if (valor_condicion > 0 && valor_condicion < 399) {
  // Si el valor del sensor está entre 1 y 399, se detecta fuego alto (peligro)
  Serial.println("---------------> Sensor Activado <---------------");
  Serial.println("....... Se detecta fuego Alto (Peligro) ....... ");
  bot.sendMessage(CHAT_ID, "FUEGO ALTO (PELIGRO)", ""); // Envía un mensaje a Telegram informando de fuego alto (peligro)
  digitalWrite(LED, HIGH); // Enciende el LED
  Heltec.display->clear(); // Limpia la pantalla OLED
  Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda
  Heltec.display->setFont(ArialMT_Plain_16); // Establece la fuente de texto
  Heltec.display->drawString(0, 0, "Enviando Datos: "); // Muestra el mensaje en la pantalla OLED
  Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda
  Heltec.display->setFont(ArialMT_Plain_10); // Establece la fuente de texto
  Heltec.display->drawString(0, 25, "Sensor Activado"); // Muestra el mensaje en la pantalla OLED
  Heltec.display->drawString(0, 40, "FUEGO ALTO (PELIGRO)"); // Muestra el mensaje en la pantalla OLED
  Heltec.display->display(); // Muestra el contenido en la pantalla OLED
}
}
```

Figura 3.37 Función principal loop.

En la función principal, se monitorea constantemente la recepción de datos LoRa y se procesan los datos recibidos para determinar el estado del sensor de detección de fuego. Dependiendo del valor del sensor, se envía un mensaje a través del *bot* de *Telegram* y se actualiza la pantalla OLED para reflejar el estado actual del sensor. Como se visualiza en la Figura 3.38 .

```
void loop() {
  int packetSize = LoRa.parsePacket(); // Verifica si hay un paquete LoRa recibido
  if (packetSize) {
    cbk(packetSize); // Procesa el paquete recibido
  }
  delay(10); // Espera 10 ms
  String STRpacket = String(packet); // Convierte el paquete recibido a string
  int valor_condicion = STRpacket.toInt(); // Convierte el string a un entero
```

Figura 3.38 Función para recibir datos en Lora.

3.5 Pruebas de funcionamiento del prototipo

Se llevaron a cabo pruebas para validar el funcionamiento del prototipo de detección de llamas con comunicación LoRa y alertas por *Telegram*.

Activación de Lecturas del Sensor desde *Arduino IDE*

Para iniciar las lecturas del sensor, se verifico el código tanto para el receptor como el emisor y subirlo correctamente En el serial de *Arduino IDE* se pueden visualizar los datos en tiempo real de la conexión wifi tanto para el receptor así también los mensajes de alerta y de funcionamiento en las dos placas en el serial de *Arduino IDE*. En las Figura 3.39 Figura 3.40 se muestra el serial del trasmisor y el receptor la información de funcionamiento, respectivamente.

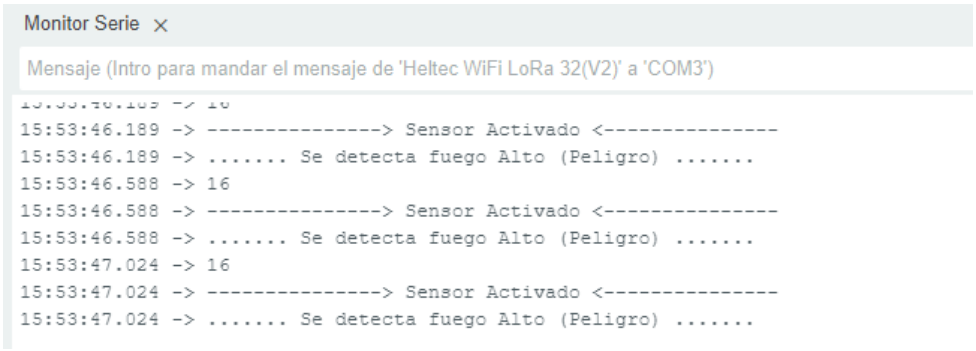


Figura 3.39 Estado del serial trasmisor

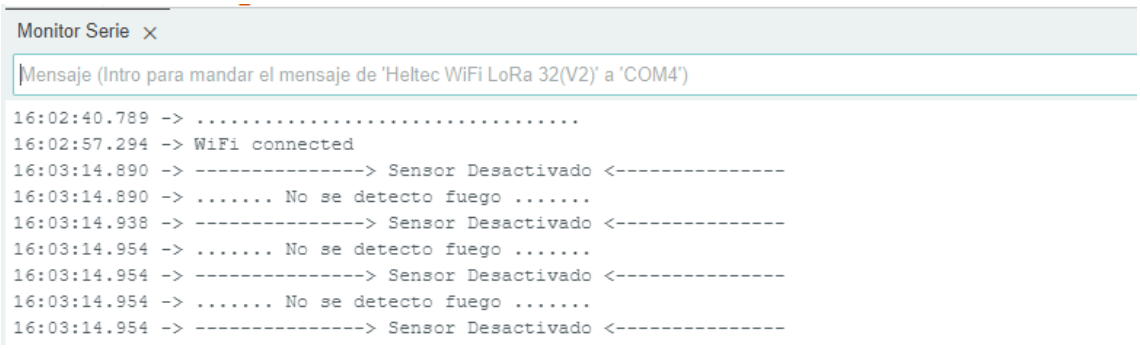


Figura 3.40 Estado del serial receptor

Activación de Lecturas del Sensor desde *Telegram*

En la Figura 3.41 se visualiza el proceso de inicio del *bot* de *Telegram* enviando el mensaje de inicialización de la comunicación para recibir la detección del sensor de llama. El *bot* envía con instrucciones de conexión de el sensor.

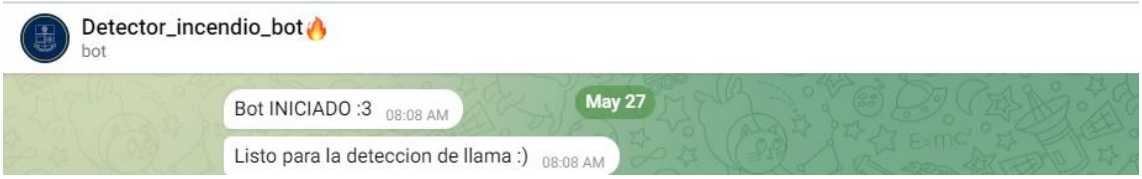


Figura 3.41 Inicio del *bot* en *Telegram*

Interpretación de Valores en las Pantallas OLED

Simultáneamente, los valores y alertas se actualizan en las pantallas OLED de los módulos Heltec LoRa, proporcionando una visualización inmediata y local de los datos.

Las pantallas OLED de los módulos Heltec LoRa están diseñadas para que cualquier usuario pueda interpretar fácilmente las alertas y tomar medidas preventivas. La Figura 3.41 muestra los datos recogidos por el sensor sobre la detección de llamas. En la parte superior de la pantalla, se incluyen notas sobre los rangos de alerta.

Figura 3.41 Visualización de los datos en la pantalla del trasmisor.

Como se visualiza en la Figura 3.42 la pantalla OLED del Heltec receptor se encuentra en un estado de conexión a internet, para así poder enviar los mensajes correspondientes a *telegram* y poder empzar la detección de llama.

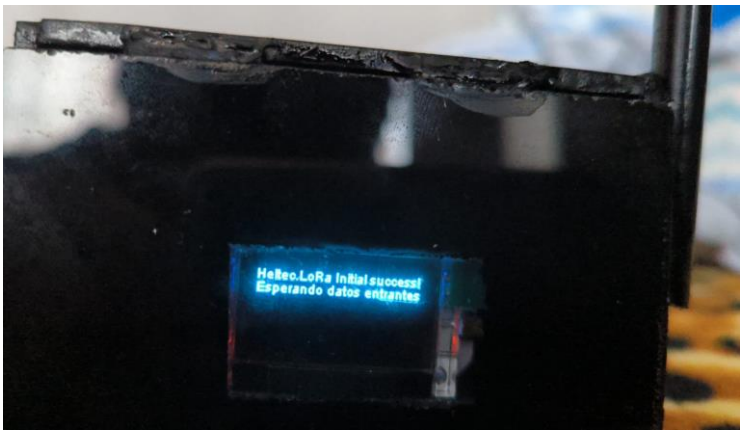


Figura 3.42 Visualización en la pantalla OLED receptor

La Tabla 3.4 a continuación clasifica los niveles de detección de fuego basados en los valores leídos por el sensor de flama en el código. Estos valores determinan el estado del fuego, desde no detectado hasta niveles peligrosos, y permiten al sistema tomar las acciones correspondientes, como mostrar mensajes tanto en *telegram* como en su pantalla OLED.

Tabla 3.4 Intervalo de valores del nivel de llama

Rango de ValorSensor	Nivel de Fuego	Descripción
1500 - 4095	Sensor Desactivado	No se detecta fuego
900 - 1499	Fuego Leve	Se detecta fuego leve
400 - 899	Fuego Moderado	Se detecta fuego moderado
0 - 399	Fuego Alto (Peligro)	Se detecta fuego alto (Peligro)

Para realizar las pruebas, se utilizará una fuente de llama, como un encendedor. Durante las pruebas, mientras la llama esté más alejada del sensor, se registrarán niveles bajos de fuego. Sin embargo, a medida que la llama se acerque más al sensor, se activará una alerta de llama media o alta. Esta alerta se indicará tanto mediante mensajes en tiempo real a través de *Telegram* como en la pantalla OLED del receptor, como se visualiza en la Figura 3.43

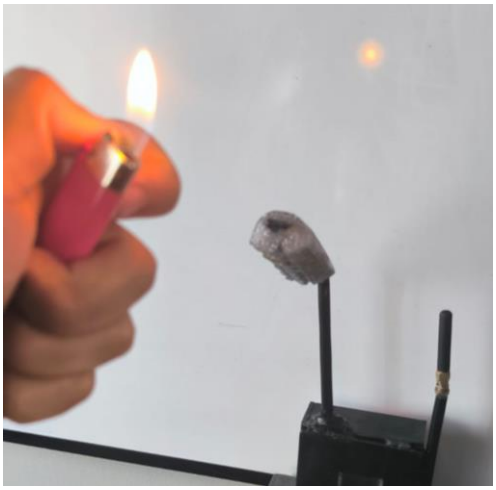


Figura 3.43 prueba de rango en el trasmisor

Visualización de Alertas en *Telegram*

Cuando el sensor detecta una llama, los valores en las pantallas OLED se elevan y se activan las alertas correspondientes. Figura 3.44 muestra estas alertas, indicando un nivel de peligro alto.

Figura 3.44 Visualización de alertas en la pantalla OLED

Simultáneamente, las alertas también se envían a la aplicación de *Telegram*, como se observa en la Figura 3.45 . Las alertas emitidas coinciden en tiempo entre las pantallas OLED y *Telegram*.

Figura 3.45 Visualización de alertas en *Telegram*

Costos de fabricación e implementación

En la Tabla 3.5 los siguientes valores corresponden a los costos de los componentes electrónicos utilizados para la fabricación e implementación de prototipos.

Tabla 3.5 Costo del prototipo

Material	**Cantidad**	**Precio Unitario**	**Precio Total**
Heltec Wifi LoRa V2 (Transmisor)	1 (u)	$ 40.00	$ 40.00
Heltec Wifi LoRa V2 (Receptor)	1 (u)	$ 40.00	$ 40.00
Sensor de llama LM393	1 (u)	$ 3.00	$ 3.00
Caja de vidrio (Transmisor/Receptor)	2 (u)	$ 10.00	$ 20.00
Silicona caliente	3 (u)	$ 1.00	$ 3.00
Pistola de silicona	1 (u)	$ 5.00	$ 5.00
Guías de plástico	2 (u)	$ 1.00	$ 2.00
Cable micro USB (Transmisor/Receptor)	2 (u)	$ 3.00	$ 6.00
Adaptador de cargador (Transmisor/Receptor)	2 (u)	$ 5.00	$ 10.00
Baquelita 10 x 5 (cm)	1 (u)	$ 1.00	$ 1.00
Mano de obra	48 (h)	$ 40.00	$ 40.00
Breadboard y cables hembra	3 (u)	$ 1.00	$ 3.00
Total			$ 173.00

Vídeo del prototipo en funcionamiento.

La Figura 3.46 muestra un código QR que conduce al vídeo de funcionamiento del prototipo.

Figura 3.46 Código QR del vídeo operativo del prototipo.

4 CONCLUSIONES

- El desarrollo de un sistema de detección de llamas con tecnología LoRa y notificaciones vía *Telegram* ha demostrado ser una solución confiable para la monitorización, alerta temprana de incendios. Este dispositivo detecta llamas en sus primeras etapas. La integración de los módulos Heltec Wifi LoRa V2 con sensores de llama LM393 ha sido fundamental para lograr una detección precisa y estable, especialmente en entornos domésticos.
- El sensor de llama LM393 ha demostrado ser esencial para este proyecto en la detección precisa de llamas, crucial para la confiabilidad del sistema. Su capacidad para detectar la luz infrarroja emitida por el fuego permite una respuesta rápida y precisa, activando alarmas y alertas de forma casi instantánea.
- La selección de los módulos Heltec Wifi LoRa V2 resultó ser una decisión acertada debido a su integración de comunicación LoRa, permitiendo una transmisión de datos eficiente y de largo alcance. Esto es especialmente beneficioso en áreas extensas o congestionadas, donde otras tecnologías de comunicación pueden fallar. Además, la capacidad de mostrar datos en las pantallas OLED integradas de los módulos y en el monitor serial de Arduino facilita ver el estado del sistema y las condiciones ambientales en tiempo real.
- La incorporación de notificaciones vía *Telegram* ha mejorado significativamente la usabilidad del sistema, permitiendo a los usuarios recibir alertas inmediatas en sus dispositivos móviles. Esta capacidad facilita una respuesta rápida y coordinada a la detección de llamas, ya que las notificaciones se pueden recibir en cualquier momento y lugar. La configuración de alertas en *Telegram* también permite a los usuarios personalizar mensajes y recibir informes detallados sobre el estado del sistema y las condiciones detectadas por los sensores.
- Para la implementación del prototipo se utilizó una carcasa de vidrio con sellado de silicona caliente, proporcionando una protección adecuada a los componentes electrónicos contra el polvo y otros contaminantes. Esta protección es fundamental para garantizar la longevidad y confiabilidad del prototipo, especialmente en entornos industriales o al aire libre donde el equipo está expuesto a condiciones adversas. Aunque el proyecto está destinado a áreas interiores como una casa, el sellado adecuado de los componentes previene daños por humedad a la vez con la tapa que incorpora poder cambiar si el módulo falla, asegurando un funcionamiento correcto del sistema en tiempo real.

- Establecer una conexión adecuada entre el sensor de llama y el microcontrolador fue crucial durante el proceso de desarrollo de la programación. La asignación correcta de los pines Tx para la comunicación en serie es esencial para asegurar que los datos se transmitan de manera precisa y sin interferencias. Fue necesaria una documentación detallada de las hojas de datos del sensor y del microcontrolador para diseñar adecuadamente el sistema y evitar errores de comunicación que pudieran comprometer la funcionalidad del prototipo.
- Durante el desarrollo del proyecto, se comprobó que la plataforma Arduino IDE, en combinación con Telegram, ofrece una ventaja significativa a los usuarios en términos de facilidad de uso y programación. Aunque el Arduino IDE es gratuito y accesible para todos, su funcionalidad resultó ser más que suficiente para la implementación y operación eficaz del prototipo. La simplicidad de su interfaz y la amplia disponibilidad de bibliotecas y documentación permitieron un desarrollo fluido del sistema, facilitando la integración con Telegram para la gestión de notificaciones en tiempo real.

5 RECOMENDACIONES

- Se recomienda calibrar el sensor de llama LM393 después de que el prototipo haya sido colocado en su ubicación definitiva. Los problemas ambientales, como puede ser la temperatura y el flujo de aire, pueden tener un impacto sustancial en los resultados del sensor, por lo que calibrar la sensibilidad del sensor a estas condiciones específicas es fundamental para una detección confiable. Además, la configuración del sensor debe revisarse y ajustarse periódicamente para garantizar la precisión a lo largo del tiempo.
- Se recomienda que futuras innovaciones y mejoras del sistema investiguen la implementación de diferentes tipos de sensores en el prototipó, como detectores de humo y sensores de temperatura, para permitir una detección de incendios más completa y confiable. Utilizar múltiples sistemas de detección puede disminuir las probabilidades de falsas alarmas y mejorar la confiabilidad del sistema. Del mismo modo, la integración de diversos tipos de sensores puede ofrecer una visión más completa de las condiciones ambientales y de seguridad en el área bajo monitoreo.
- Para futuros trabajos, se sugiere considerar fuentes de energía autónomas como paneles solares o baterías. Esto aseguraría el funcionamiento continuo del

sistema incluso durante cortes de energía, garantizando una detección y notificación constante de incendios. La implementación de un suministro de energía autónomo también facilitaría la instalación del sistema en áreas rurales donde el acceso a la electricidad puede ser limitado.

- Los repetidores deben instalarse estratégicamente para mejorar la cobertura y confiabilidad de la red LoRa. Esto garantiza una transferencia de datos confiable en áreas más amplias y mayores obstáculos físicos, lo que hace que el sistema funcione de manera eficiente incluso en circunstancias difíciles. La ampliación de la red LoRa mediante repetidores permite la monitorización simultánea de numerosas ubicaciones, aumentando la adaptabilidad y aplicación del sistema.
- Desarrollar una interfaz de usuario más intuitiva y accesible para la configuración y control del sistema puede significativamente mejorar la experiencia del usuario. Esta interfaz debería permitir a los usuarios ingresar datos como nombres y contraseñas de redes Wi-Fi, así como detalles de la aplicación de mensajería Telegram, sin necesidad de modificar directamente el código fuente. Simplificar la configuración del sistema puede reducir errores y hacer que el sistema sea más accesible para usuarios con diferentes niveles de experiencia técnica.
- Es fundamental preservar los componentes electrónicos del sistema de condiciones ambientales adversas. El uso de cajas con sellado adecuado y materiales resistentes puede mejorar la durabilidad y confiabilidad del sistema en una variedad de condiciones. Proteger los componentes electrónicos del polvo, la humedad y las altas temperaturas ayuda a extender la vida útil del sistema y a mantener un rendimiento óptimo a largo plazo, eliminando la necesidad de reparaciones y reemplazos frecuentes.

6 BIBLIOGRAFIA

[1] «Electrónica Triacs,» [En línea]. Available: https://triacs.cl/sensores/281-sensor-de-llama-3-pines-lm393-.html. [Último acceso: 20 Mayo 2024].

[2] «Heltec Wifi LoRa V2,» Heltec Automation, [En línea]. Available: https://heltec.org/project/wifi-lora-32-v3/. [Último acceso: 20 Mayo 2024].

[3] «*Telegram*,» Cómo usar *Telegram*: ¿Qué es *Telegram*?, [En línea]. Available: https://edu.gcfglobal.org/es/curso-de-*telegram*/que-es-*telegram*/1/. [Último acceso: 21 Mayo 2024].

[4] «Arduino Cloud Editor,» *Arduino IDE* 2.3.2, [En línea]. Available: https://www.arduino.cc/en/*software*. [Último acceso: 21 Mayo 2024].

[5] S. Web, «*Telegram Bot*s: A practical Guide,» *Telegram*, [En línea]. Available: https://core.*telegram*.org/*bot*s. [Último acceso: 22 Mayo 2024].

[6] «Wifi LoRa 32 v2,» Heltec.org, Mayo 2020. [En línea]. Available: https://resource.heltec.cn/download/Manual%20Old/WiFi%20Lora32Manual.pdf. [Último acceso: 22 Mayo 2024].

[7] «DEFINICIÓN DE LLAMA,» Definición.de, 2008. [En línea]. Available: https://definicion.de/llama/. [Último acceso: 24 Mayo 2024].

[8] «¿Qué es la tecnología LoRa y por qué es importante para IoT?,» The Things Network User, 9 Diciembre 2019. [En línea]. Available: https://www.thethingsnetwork.org/community/santa-rosa/post/que-es-la-tecnologia-lora-y-por-que-es-importante-para-iot. [Último acceso: 25 Mayo 2024].

[9] «Modulo Sensor Detector de Flama Infrarrojo LM393,» ElectronicaPTY, [En línea]. Available: http://electronicapty.com/tienda/modulos-y-sensores/modulo-sensor-detector-de-flama-infrarrojo-lm393-detail. [Último acceso: 22 Mayo 2024].

[10] «Cómo usar *Telegram*: ¿Qué son los *bot*s de *Telegram*?,» GCFGlobal, [En línea]. Available: https://edu.gcfglobal.org/es/curso-de-*telegram*/que-son-los-*bot*s-de-*telegram*/1/. [Último acceso: 26 Mayo 2024].

[11] «Github,» Arduino library for Heltec ESP32 (or ESP32+LoRa) based boards, [En línea]. Available: https://github.com/HelTecAutomation/Heltec_ESP32. [Último acceso: 27 Mayo 2024].

[12] «Crea tu propio *bot* de *Telegram* sin saber programar,» Blogthinkbig.com, 2020. [En línea]. Available: https://blogthinkbig.com/crear-*bot*-de-*telegram-botfather*/. [Último acceso: 28 Mayo 2024].

[13] «HELTEC WIFI LORA 32 V2 - 868MHZ,» Molukas, [En línea]. Available: https://shop.molukas.com/es/nodos/36-heltec-wifi-lora-32-v2-868mhz.html. [Último acceso: 20 Mayo 2024].

[14] «¿Cuál es la diferencia entre 433MHz y 915MHz para redes LoRaWAN??,» LoraAntena, [En línea]. Available: https://www.loraantenna.com/es/whats-the-difference-between-433mhz-and-915mhz-for-lorawan-networks/. [Último acceso: 30 Mayo 2024].

[15] «LM393 Módulo de sensor de detección de llama IR de 3 pines Detector de incendios Módulo receptor de infrarrojos,» ELECbee, [En línea]. Available: https://www.elecbee.com/es-26346-LM393-3-Pin-IR-Flame-Detection-Sensor-Module-Fire-Detector-Infrared-Receiver-Module. [Último acceso: 29 Mayo 2024].

[16] «*Arduino IDE*,» Wikipedia, 7 Febrero 2024. [En línea]. Available: https://es.wikipedia.org/wiki/Arduino_IDE. [Último acceso: 2 Junio 2024].

[17] «Conoce EasyEDA, un completo *software* de simulación de circuitos y diseño de PCB online,» EasyEDA, [En línea]. Available: https://www.redeszone.net/2016/02/19/conoce-easyeda-un-completo-*software*-de-simulacion-de-circuitos-y-diseno-de-pcb-online/. [Último acceso: 2 Junio 2024].

[18] «CubeCell – AB01 Dev-Board (V2),» Heltec Automation, [En línea]. Available: https://heltec.org/project/htcc-ab01-v2/. [Último acceso: 3 Junio 2024].

[19] «Sensor infrarrojo detector de Flama KY-026,» Novatronic, [En línea]. Available: https://novatronicec.com/index.php/product/sensor-de-sonido-ky037/. [Último acceso: 4 Mayo 2024].

[20] «*Arduino IDE*,» Microsoft, [En línea]. Available: https://apps.microsoft.com/detail/9nblggh4rsd8?hl=es-mx&gl=MX. [Último acceso: 3 Juino 2024].

[21] «Easy-to-use & Free PCB Design *Software*,» EasyEDA, [En línea]. Available: https://easyeda.com/es. [Último acceso: 4 Junio 2024].

[22] «Obtener nuestro ID de *telegram*,» jejo.es (Em50L), [En línea]. Available: https://jejo.es/posts/*telegram*/1-

obtener_nuestro_id_de_*telegram*/#:~:text=El%20id%20de%20*telegram*%20es,a%20traves%20de%20un%20*bot*.. [Último acceso: 5 JUNIO 2024].

[23] «WiFi and LoRa Energy Consumption Comparison in IoT ESP 32/ SX1278 Devices,» 26 Abril 2023. [En línea]. Available: https://hal.science/hal-04083109/document. [Último acceso: 6 Junio 2024].

[24] «Cómo calcular la corriente total,» WikiHow, [En línea]. Available: https://es.wikihow.com/calcular-la-corriente-total. [Último acceso: 6 Junio 2024].

[25] « SEMTECH,» de *AN1200.22 LoRa™ Modulation Basics* , Semtech Corporation, 2015, p. 26.

[26] «Montaje del PCB: los componentes más críticos y vulnerables,» Proto-Electronics, [En línea]. Available: https://www.proto-electronics.com/es/blog/montaje-pcb-componentes-criticos-vulnerables. [Último acceso: 7 Junio 2024].

7 ANEXOS

La lista de los **Anexos** se muestra a continuación:

ANEXO I. Certificado de originalidad

ANEXO II. Código trasmisor

ANEXO III. Código emisor

ANEXO I: Certificado de Originalidad

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 24 de julio de 2024

De mi consideración:

Yo, **CARLOS ANDRÉS YUNGA SÁNCHEZ**, en calidad de Directora del Trabajo de Integración Curricular titulado **IMPLEMENTACIÓN DE UN PROTOTIPO DE ALERTA POR DETECCIÓN DE LLAMAS BASADO EN LORA Y TELEGRAM** asociado al **IMPLEMENTACIÓN DE UN PROTOTIPO DE ALERTA POR DETECCIÓN DE LLAMAS BASADO EN LORA Y TELEGRAM** elaborado por la estudiante **RONALD ADRIÁN CHASIPANTA GUALPA** de la carrera en Tecnología Superior en Redes y Telecomunicaciones, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito completo, producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 12%.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el link del informe generado por la herramienta Turnitin.

Atentamente,

CARLOS ANDRÉS YUNGA SÁNCHEZ

Técnico Docente

Escuela de Formación de Tecnólogos

ANEXO II: Código fuente trasmisor

```
#include "heltec.h" // Incluye la librería Heltec

#include "images.h" // Incluye la librería de imágenes

#define flama_pin A0 // Define el pin analógico A0 como flama_pin

int ValorSensor = 0; // Inicializa la variable ValorSensor en 0

#define BAND 915E6  // Define la banda de frecuencia LoRa a 915 MHz


unsigned int counter = 0; // Inicializa el contador en 0

String rssi = "RSSI --"; // Inicializa la variable rssi

String packSize = "--"; // Inicializa la variable packSize

String packet; // Declara la variable

void logo() {

  Heltec.display->clear(); // Limpia la pantalla

  Heltec.display->drawXbm(0, 5, logo_width, logo_height, logo_bits); // Dibuja el logo en la pantalla

  Heltec.display->display(); // Muestra el contenido en la pantalla

}

void setup() {

  Serial.begin(115200);

  Heltec.begin(true, true, true, true, BAND); // Inicializa la pantalla y el módulo LoRa con la banda especificada

  pinMode(flama_pin, INPUT); // Configura el pin de la flama como entrada

  Heltec.display->init(); // Inicializa la pantalla OLED

  Heltec.display->flipScreenVertically(); // Voltea la pantalla verticalmente

  Heltec.display->setFont(ArialMT_Plain_16); // Establece la fuente de texto

  logo(); // Llama a la función logo para mostrarlo en la pantalla

  delay(1500); // Espera 1.5 segundos

  Heltec.display->clear(); // Limpia la pantalla
```

```
  Heltec.display->drawString(0, 0, "SISTEMA INICIALIZADO"); // Muestra el mensaje en la pantalla

  Heltec.display->drawString(0, 25, "Sistema Indicador "); // Muestra el mensaje en la pantalla

  Heltec.display->drawString(0, 40, "detector de Fuego"); // Muestra el mensaje en la pantalla

  Heltec.display->display(); // Muestra el contenido en la pantalla

  delay(10000); // Espera 10 segundos

}

void loop() {

  ValorSensor = analogRead(flama_pin); // Lee el valor analógico del pin de la flama

  Serial.println(ValorSensor); // Imprime el valor del sensor en la consola

  if (ValorSensor > 1500 && ValorSensor <= 4095) {

    Serial.println("----------> Sensor Desactivado <----------"); // Imprime el mensaje en la consola

    Serial.println("..... No se detecto fuego ..... "); // Imprime el mensaje en la consola

    digitalWrite(LED, LOW); // Apaga el LED

    Heltec.display->clear(); // Limpia la pantalla

    Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda

    Heltec.display->setFont(ArialMT_Plain_16); // Establece la fuente de texto

    Heltec.display->drawString(0, 0, "Enviando Datos: "); // Muestra el mensaje en la pantalla

    Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda

    Heltec.display->setFont(ArialMT_Plain_10); // Establece la fuente de texto

    Heltec.display->drawString(0, 25, "Sensor desactivado"); // Muestra el mensaje en la pantalla

    Heltec.display->drawString(0, 40, "No se detecto FUEGO"); // Muestra el mensaje en la pantalla

    Heltec.display->display(); // Muestra el contenido en la pantalla

  } else if (ValorSensor > 900 && ValorSensor < 1499) {
```

```
    Serial.println("----------> Sensor Activado <----------"); // Imprime el mensaje en la consola

    Serial.println("..... Se detecta fuego leve ..... "); // Imprime el mensaje en la consola

    digitalWrite(LED, HIGH); // Enciende el LED

    Heltec.display->clear(); // Limpia la pantalla

    Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda

    Heltec.display->setFont(ArialMT_Plain_16); // Establece la fuente de texto

    Heltec.display->drawString(0, 0, "Enviando Datos: "); // Muestra el mensaje en la pantalla

    Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda

    Heltec.display->setFont(ArialMT_Plain_10); // Establece la fuente de texto

    Heltec.display->drawString(0, 25, "Sensor Activado"); // Muestra el mensaje en la pantalla

    Heltec.display->drawString(0, 40, " FUEGO LEVE"); // Muestra el mensaje en la pantalla

    Heltec.display->display(); // Muestra el contenido en la pantalla

  } else if (ValorSensor > 400 && ValorSensor < 899) {

    Serial.println("----------> Sensor Activado <----------"); // Imprime el mensaje en la consola

    Serial.println("..... Se detecta fuego moderado ..... "); // Imprime el mensaje en la consola

    digitalWrite(LED, HIGH); // Enciende el LED

    Heltec.display->clear(); // Limpia la pantalla

    Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda

    Heltec.display->setFont(ArialMT_Plain_16); // Establece la fuente de texto

    Heltec.display->drawString(0, 0, "Enviando Datos: "); // Muestra el mensaje en la pantalla

    Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda

    Heltec.display->setFont(ArialMT_Plain_10); // Establece la fuente de texto
```

```
    Heltec.display->drawString(0, 25, "Sensor Activado"); // Muestra el mensaje en la pantalla

    Heltec.display->drawString(0, 40, " FUEGO MODERADO"); // Muestra el mensaje en la pantalla

    Heltec.display->display(); // Muestra el contenido en la pantalla

  } else if (ValorSensor > 0 && ValorSensor < 399) {

    Serial.println("----------------> Sensor Activado <----------------"); // Imprime el mensaje en la consola

    Serial.println("..... Se detecta fuego Alto (Peligro) ..... "); // Imprime el mensaje en la consola

    digitalWrite(LED, HIGH); // Enciende el LED

    Heltec.display->clear(); // Limpia la pantalla

    Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda

    Heltec.display->setFont(ArialMT_Plain_16); // Establece la fuente de texto

    Heltec.display->drawString(0, 0, "Enviando Datos: "); // Muestra el mensaje en la pantalla

    Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda

    Heltec.display->setFont(ArialMT_Plain_10); // Establece la fuente de texto

    Heltec.display->drawString(0, 25, "Sensor Activado"); // Muestra el mensaje en la pantalla

    Heltec.display->drawString(0, 40, " FUEGO ALTO (PELIGRO)"); // Muestra el mensaje en la pantalla

    Heltec.display->display(); // Muestra el contenido en la pantalla

  }

  LoRa.beginPacket(); // Inicia un paquete nuevo

  LoRa.setTxPower(14, RF_PACONFIG_PASELECT_PABOOST); // Establece la potencia de transmisión LoRa

  LoRa.print(ValorSensor); // Envía el valor del sensor

  LoRa.endPacket(); // Finaliza el paquete LoRa


}
```

ANEXO III: Código fuente emisor

```
#include "heltec.h" // Librería específica de Heltec para manejar el hardware

#include "images.h" // Librería para manejar las imágenes en la pantalla OLED

#include <WiFi.h> // Librería para manejar la conexión Wifi

#include <WiFiClientSecure.h> // Librería para manejar la conexión segura

#include <UniversalTelegramBot.h> // Librería para manejar el bot de Telegram

#include <ArduinoJson.h> // Librería para manejar JSON

unsigned long previousMillis = 0;

const long interval = 1500;

const char* ssid = "Xiaomi"; // red Wifi

const char* password = "ojitos12"; // Contraseña

#define BOTtoken "7128924155:AAF1uUMWKunmezFubYcIAD6OfvvqnOrnxuM" // Token del Bot (Obtenido del Botfather)

// También necesitas hacer clic en "start" en un bot antes de que pueda

#define CHAT_ID "1773755765"

WiFiClientSecure client; // Cliente seguro para manejar la conexión

UniversalTelegramBot bot(BOTtoken, client); // Inicializa el bot con el token y el cliente seguro

#define BAND 915E6 // Configura la banda para la comunicación LoRa, por ejemplo, 868E6, 915E6

String rssi = "RSSI --"; // Variable para almacenar el valor de RSSI

String packSize = "--"; // Variable para almacenar el tamaño del paquete

String packet; // Variable para almacenar el contenido del paquete
```

```
void logo() {
  Heltec.display->clear(); // Limpia la pantalla
  Heltec.display->drawXbm(0, 5, logo_width, logo_height, logo_bits); // Dibuja el logo en la pantalla OLED
  Heltec.display->display(); // Muestra el contenido en la pantalla OLED
}
void LoRaData() {
  Heltec.display->clear(); // Limpia la pantalla
  Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda
  Heltec.display->setFont(ArialMT_Plain_10); // Establece la fuente de texto
  Heltec.display->drawString(0, 15, "Received " + packSize + " bytes"); // Muestra el tamaño del paquete recibido
  Heltec.display->drawStringMaxWidth(0, 26, 128, packet); // Muestra el contenido del paquete
  Heltec.display->drawString(0, 0, rssi); // Muestra el valor de RSSI
  Heltec.display->display(); // Muestra el contenido en la pantalla OLED
}
void cbk(int packetSize) {
  packet = ""; // Limpia el contenido del paquete
  packSize = String(packetSize, DEC); // Convierte el tamaño del paquete a string
  for (int i = 0; i < packetSize; i++) {
    packet += (char) LoRa.read(); // Lee cada byte del paquete y lo agrega a la variable packet
  }
  rssi = "RSSI " + String(LoRa.packetRssi(), DEC); // Obtiene el valor de RSSI y lo convierte a string
```

```
  //LoRaData();
}
void setup() {
  // La serie WIFI Kit V1 no soporta el control Vext
  Heltec.begin(true /*DisplayEnable Enable*/, true /*LoRa Enable*/, true /*Serial Enable*/, true /*PABOOST Enable*/, BAND /*LoRa Band*/);
  Serial.begin(115200); // Inicializa la comunicación serial a 115200 bps
  Heltec.display->init(); // Inicializa la pantalla OLED
  Heltec.display->flipScreenVertically(); // Voltea la pantalla OLED verticalmente
  Heltec.display->setFont(ArialMT_Plain_10); // Establece la fuente de texto
  logo(); // Muestra el logo en la pantalla OLED
  delay(1500); // Espera 1.5 segundos
  Heltec.display->clear(); // Limpia la pantalla OLED
  Heltec.display->drawString(0, 0, "Heltec.LoRa inicializado con exito!"); // Muestra el mensaje de éxito de inicialización
  Heltec.display->drawString(0, 10, "Esperando datos entrantes..."); // Muestra el mensaje de espera de datos entrantes
  Heltec.display->display(); // Muestra el contenido en la pantalla OLED
  delay(1000); // Espera 1 segundo
  //LoRa.onReceive(cbk);
  LoRa.receive(); // Pone el módulo LoRa en modo de recepción
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password); // Conecta al Wifi con el SSID y contraseña dados
  client.setCACert(TELEGRAM_CERTIFICATE_ROOT);
  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("Wifi connected");
  bot.sendMessage(CHAT_ID, "Bot INICIADO :3", "");
  bot.sendMessage(CHAT_ID, "Listo para la deteccion de llama :)", ""); // Envía un mensaje al chat de Telegram indicando que el bot se ha iniciado
}
void loop() {
  int packetSize = LoRa.parsePacket(); // Verifica si hay un paquete LoRa recibido
  if (packetSize) {
    cbk(packetSize); // Procesa el paquete recibido
  }
  delay(10); // Espera 10 ms
  String STRpacket = String(packet); // Convierte el paquete recibido a string
  int valor_condicion = STRpacket.toInt(); // Convierte el string a un entero
  if (valor_condicion > 1500 && valor_condicion <= 4095) {
    // Si el valor del sensor está entre 1501 y 4095, el sensor está desactivado (no se detecta fuego)
    Serial.println("----------> Sensor Desactivado <----------");
    Serial.println("..... No se detecto fuego ..... ");
    digitalWrite(LED, LOW); // Apaga el LED
    Heltec.display->clear(); // Limpia la pantalla OLED
```

Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda

Heltec.display->setFont(ArialMT_Plain_16); // Establece la fuente de texto

Heltec.display->drawString(0, 0, "Recibiendo Datos: "); // Muestra el mensaje en la pantalla OLED

Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda

Heltec.display->setFont(ArialMT_Plain_10); // Establece la fuente de texto

Heltec.display->drawString(0, 25, "Sensor desactivado"); // Muestra el mensaje en la pantalla OLED

Heltec.display->drawString(0, 40, "No se detecto FUEGO"); // Muestra el mensaje en la pantalla OLED

Heltec.display->display(); // Muestra el contenido en la pantalla OLED

// *bot*.sendMessage(CHAT_ID, "No se detecto FUEGO", ""); // Envía un mensaje a *Telegram* informando que no se detectó fuego

} else if (valor_condicion > 900 && valor_condicion < 1499) {

// Si el valor del sensor está entre 901 y 1499, se detecta fuego leve

Serial.println("----------> Sensor Activado <----------");

Serial.println("..... Se detecta fuego leve ..... ");

bot.sendMessage(CHAT_ID, "FUEGO LEVE", ""); // Envía un mensaje a *Telegram* informando de fuego leve

digitalWrite(LED, HIGH); // Enciende el LED

Heltec.display->clear(); // Limpia la pantalla OLED

Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda

Heltec.display->setFont(ArialMT_Plain_16); // Establece la fuente de texto

Heltec.display->drawString(0, 0, "Recibiendo Datos: "); // Muestra el mensaje en la pantalla OLED

Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda

Heltec.display->setFont(ArialMT_Plain_10); // Establece la fuente de texto

Heltec.display->drawString(0, 25, "Sensor Activado"); // Muestra el mensaje en la pantalla OLED

Heltec.display->drawString(0, 40, "FUEGO LEVE"); // Muestra el mensaje en la pantalla OLED

Heltec.display->display(); // Muestra el contenido en la pantalla OLED

} else if (valor_condicion > 400 && valor_condicion < 899) {

// Si el valor del sensor está entre 401 y 899, se detecta fuego moderado

Serial.println("----------> Sensor Activado <----------");

Serial.println("..... Se detecta fuego moderado ..... ");

bot.sendMessage(CHAT_ID, "FUEGO MODERADO", ""); // Envía un mensaje a *Telegram* informando de fuego moderado

digitalWrite(LED, HIGH); // Enciende el LED

Heltec.display->clear(); // Limpia la pantalla OLED

Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda

Heltec.display->setFont(ArialMT_Plain_16); // Establece la fuente de texto

Heltec.display->drawString(0, 0, "Enviando Datos: "); // Muestra el mensaje en la pantalla OLED

Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda

Heltec.display->setFont(ArialMT_Plain_10); // Establece la fuente de texto

Heltec.display->drawString(0, 25, "Sensor Activado"); // Muestra el mensaje en la pantalla OLED

Heltec.display->drawString(0, 40, "FUEGO MODERADO"); // Muestra el mensaje en la pantalla OLED

Heltec.display->display(); // Muestra el contenido en la pantalla OLED

} else if (valor_condicion > 0 && valor_condicion < 399) {

// Si el valor del sensor está entre 1 y 399, se detecta fuego alto (peligro)

Serial.println("----------> Sensor Activado <----------");

Serial.println("..... Se detecta fuego Alto (Peligro) ..... ");

bot.sendMessage(CHAT_ID, "FUEGO ALTO (PELIGRO)", ""); // Envía un mensaje a *Telegram* informando de fuego alto (peligro)

digitalWrite(LED, HIGH); // Enciende el LED

Heltec.display->clear(); // Limpia la pantalla OLED

Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda

Heltec.display->setFont(ArialMT_Plain_16); // Establece la fuente de texto

Heltec.display->drawString(0, 0, "Enviando Datos: "); // Muestra el mensaje en la pantalla OLED

Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT); // Alinea el texto a la izquierda

Heltec.display->setFont(ArialMT_Plain_10); // Establece la fuente de texto

Heltec.display->drawString(0, 25, "Sensor Activado"); // Muestra el mensaje en la pantalla OLED

Heltec.display->drawString(0, 40, "FUEGO ALTO (PELIGRO)"); // Muestra el mensaje en la pantalla OLED

Heltec.display->display(); // Muestra el contenido en la pantalla OLED

}}