UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE



“ SMART ENERGY DATA FOR REMOTE ELDERLY CARE”

BRUNO ALEJANDRO NAVARRETE ROMÁN

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRÓNICO MENCIÓN COMPUTADORES

PROFESOR GUÍA: **MOHAMED ABDELHAMID**
PROFESOR CORREFERENTE: **JOSÉ MANUEL MARTÍNEZ**

Septiembre 2026

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☑ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Smart Energy Data For Remote Elderly Care

Nombre del candidato(a): Bruno Alejandro Navarrete Román

Carrera / Grado: Ingeniería Civil Electrónica / Ingeniero Civil Electrónico

Campus: Casa Central **Departamento:** Departamento de Electrónica

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Mohamed Abdelhamid, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☑ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por **(marcar una opción)**:

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

_____

_____

_____

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 26 de octubre de 2026 **Firma:** 

Estudiante o Candidato(a):

Fecha: 26 de octubre de 2026 **Firma:** 

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM

La vejez no es más que el resultado de una larga vida de amor, de dolores, de luchas y de alegrías. En el rostro de un anciano se lee la historia de un alma.

Para mi abuela Orielle.

Agradecimientos

Sin duda, lo primero siempre será la familia; nada de esto habría sido posible sin su apoyo constante. Agradezco profundamente a mis padres, responsables de que hoy sea quien soy; a mis hermanas, que me han acompañado desde mi primer día; a mi abuela, raíz y matriarca de lo que más amo en mi vida; a mi tía y tíos; a mis primos y primas; y, por supuesto, a mis sobrinos.

Los agradecimientos no pueden partir por una persona que no se ella, mi madre, Alejandra. Mi mamá es la base sobre la que se construyó todo lo que soy hoy, sin ella yo no sería esta persona. Gracias a ella he logrado terminar este arduo proceso, desde quedar en la universidad de mis sueños, superar cada asignatura, levantarme de las caídas, hasta escribir esta memoria y terminar mi carrera. Ella fue la que me motivó, a su manera, para subir mis notas en enseñanza media. Ella estuvo ahí, para mi, cada vez que lo necesité. Ella se aseguró de que trabajara en mi escrito, después de mucho tiempo inactivo. Su apoyo incondicional fue primordial para poder seguir adelante cuando las cosas no resultaban como uno espera o cuando el mundo parecía acabar, ella secó mis lagrimas y me dio las palabras de aliento que permitieron alzar la mirada nuevamente y fijar rumbo en los objetivos. Gracias por estar ahí, por escucharme, por tus concejos, por retarme cuando correspondía, gracias por tu amor, por todo lo que haces, gracias mamá.

Mi papá también merece un lugar especial. Gracias por todo lo que me has brindado y por el esfuerzo que hiciste para permitirme estudiar lo que quise, en la universidad que elegí, aun cuando tomó más tiempo del esperado. Gracias por intentar inculcarme disciplina y responsabilidad; creo que bastante de eso quedó en mí. Gracias por todo, papá.

Ahora mis amigos, por suerte soy una persona con muchas buenas amistades y personas en las que puedo confiar. Agradezco a todas mis amistades por sus palabras de ánimo, por las risas y los buenos momentos, por escucharme mientras hablo ñoñerias que solo me interesan a mi. Agradezco a mis amistades de Parral por darme una buena bienvenida cada vez que volví a mi pueblo y cada verano que se aprovechó a concho. Agradezco a las personas que conocí en Valparaíso y que se volvieron parte de mi círculo, por todo lo que vivimos, por las aventura y locuras. Sin duda, de estos años de universidad son lo mejor que llevo.

Menciones honrosas, al Pato, al Rebo y a la Ano por ser mi grupo de carrera, por todas esas noches de estudios, por todas las conversaciones, por su cariño, por la compañía, por recibirme en sus casas; a la Eli por estar todos los días, por su honestidad, por escucharme y por sus podcast; a la Anaís por ser una distinta, por aportar esa calidez humana y por ser la astrología de mi astronomía.

También agradezco a todos los profesores de esta universidad que con su talento y vocación traspasaron sus conocimientos hacia mi. A los funcionarios también, que hacen mas fácil la estadía en esta compleja casa de estudios. A los estudiantes que alguna vez me ayudaron resolviendo consultas, dando consejos o simplemente haciendo más amena la vida universitaria. Dentro de este grupo, un agradecimiento muy especial a mi profesor guía, Mohamed Abdelhamid, por su tiempo, su inagotable paciencia, sus palabras de aliento y por recibirme siempre con las puertas abiertas.

Funding

This work was supported by ANID Subdirección de Investigación Aplicada FONDEF NAM25I0038, ANID CTI 250019 Innovation Center for Sustainable Energy Transition (SET-Chile), and Proyecto Interno USM 2025 under Grant PI_LIR_25_15.

Resumen

El envejecimiento de la población exige soluciones no intrusivas para el monitoreo de adultos mayores. Este trabajo evalúa la viabilidad de utilizar el consumo eléctrico como indicador de actividad, comparando empíricamente dos enfoques de adquisición de datos: visión por computadora y detección de pulsos luminosos.

Se desarrollaron y evaluaron dos prototipos en un ambiente de laboratorio y asegurado mediante un diferencial y un interruptor termomagnético. El primero emplea una ESP32-CAM que envía fotografías del medidor a un servidor local, donde un script de Python utiliza Tesseract OCR para extraer el consumo. El algoritmo segmenta la imagen y aplica un barrido de umbrales para mitigar las variaciones de luz, realizando validaciones estadísticas antes de enviar los datos a ThingSpeak mediante HTTP POST. El segundo prototipo utiliza un fotodiodo BPW34 conectado a un ESP32 que cuenta los pulsos de un medidor electrónico (1600 imp/kWh). Este enfoque realiza el cálculo de consumo completamente en el dispositivo (Full Edge) y actualiza ThingSpeak de forma directa y autónoma.

Las pruebas contemplaron variaciones de potencia, tasas de muestreo, condiciones lumínicas y ensayos continuos de 48 horas para medir el error acumulado, así como tasa de éxito y de ancho de banda. Los resultados demostraron la clara superioridad del fotodiodo: es menos intrusivo, reduce drásticamente la carga en la red WiFi, evita la pérdida de datos por desincronización de captura y posee una precisión 160 veces mayor. El principal logro de esta investigación es validar que un dispositivo económico y de fácil instalación puede extraer información altamente valiosa a partir de simples pulsos luminosos, permitiendo inferir rutinas de manera confiable para el monitoreo remoto de personas mayores.

palabras clave: **NILM (Non-Intrusive Load Monitoring)**, **IoT (Internet of Things)**, **Remote Elderly Care**, **Computer Vision**, **Edge Computing**

Abstract

Population aging demands non-intrusive solutions for monitoring the elderly. This work evaluates the feasibility of using electrical consumption as an activity indicator by empirically comparing two data acquisition approaches: computer vision and light pulse detection.

Two prototypes were developed and tested in a laboratory enviroment. The testbed is protected with a differential and an electromagnetic circuit breaker. The first prototype uses an ESP32-CAM that sends meter photographs to a local server, where a Python script uses Tesseract OCR to extract consumption data. The algorithm segments the image and applies a threshold sweep to mitigate lighting variations, performing statistical validations before sending the data to ThingSpeak via HTTP POST. The second prototype uses a BPW34 photodiode connected to an ESP32 to count the pulses of an electronic meter (1600 imp/kWh). This approach calculates consumption entirely on the device (Full Edge) and updates ThingSpeak directly and autonomously.

Testing included load variations, sampling rates, lighting conditions, bandwidth and computational cost analysis, and 48-hour continuous trials to measure cumulative error. The results demonstrated the clear superiority of the photodiode: it is less intrusive, drastically reduces WiFi network load, prevents data loss from capture timing issues, and offers 160 times higher precision. The main achievement of this research is validating that an affordable, easy-to-install device can extract highly valuable information from simple light pulses, reliably inferring daily routines for the remote monitoring of older adults.

keywords: **NILM (Non-Intrusive Load Monitoring)**, **IoT (Internet of Things)**, **Remote Elderly Care**, **Computer Vision**, **Edge Computing**

Glosario

ADL *Activities of Daily Living* (Actividades de la vida diaria), rutinas básicas que reflejan la autonomía funcional.

AFE *Analog Front-End*, dispositivo o módulo utilizado en aplicaciones de medición de alta precisión.

ANN *Artificial Neural Networks* (Redes Neuronales Artificiales), empleadas en algoritmos de desagregación.

API *Application Programming Interface* (Interfaz de programación de aplicaciones), conjunto de rutinas y protocolos para la comunicación entre servicios web.

BPW34 Fotodiodo PIN de silicio de alta velocidad utilizado para la captura de pulsos ópticos.

CERMI Medidor Inteligente Comercial desplegado por compañías distribuidoras de energía en Chile.

CT *Current Transformer*, sensor o transformador de corriente.

Deep Sleep Estado de suspensión profunda del microcontrolador para maximizar la eficiencia energética.

Edge Computing Procesamiento en el borde, técnica que traslada el cálculo directamente al microcontrolador local.

ESP32 Familia de microcontroladores SoC con potente capacidad de procesamiento y conectividad WiFi nativa.

ESP32-CAM Módulo de cámara SoC de bajo costo y tamaño reducido, utilizado para tareas de visión por computadora.

Full Edge Paradigma de procesamiento donde el cálculo se realiza de manera 100% autónoma en el dispositivo local.

GPIO *General Purpose Input/Output*, pines de entrada y salida de propósito general en placas de desarrollo.

Heatmap Mapa de calor analítico utilizado para identificar patrones y anomalías en la rutina de consumo.

HTTP *Hypertext Transfer Protocol*, protocolo utilizado para el envío de paquetes de datos a la nube.

I2C *Inter-Integrated Circuit*, bus y protocolo de comunicación en serie.

IEC TS 63297 Estándar técnico internacional que establece el marco de referencia para clasificar dispositivos NILM.

ILM *Intrusive Load Monitoring*, sistemas intrusivos de monitoreo de carga con conexión galvánica.

IoT *Internet of Things* (Internet de las Cosas).

JSON *JavaScript Object Notation*, formato de archivo utilizado para almacenar estados históricos y variables.

KNN *K-Nearest Neighbors*, algoritmo clasificador (k-vecinos más cercanos) utilizado en aprendizaje automático.

LoRaWAN *Long Range Wide Area Network*, tecnología de comunicación de área amplia y bajo consumo.

LPWAN *Low Power Wide Area Network*, red inalámbrica diseñada para baja tasa de datos y largo alcance.

LSTM *Long Short-Term Memory*, tipo de red neuronal recurrente.

MQTT *Message Queuing Telemetry Transport*, protocolo de mensajería ligero estandarizado para IoT.

NILM *Non-Intrusive Load Monitoring*, técnica para inferir consumo eléctrico desagregado sin intervenir la red de 220V.

NSD *NILM Sensing Device*, dispositivo de sensado empleado en sistemas de Monitoreo de Carga No Intrusivo.

NTP *Network Time Protocol*, protocolo utilizado para la sincronización exacta del reloj interno de la placa.

OCR *Optical Character Recognition*, Reconocimiento Óptico de Caracteres para extraer texto de imágenes.

PCB *Printed Circuit Board*, placa de circuito impreso.

PLC *Power Line Communication*, comunicación de datos a través de las propias líneas de energía eléctrica.

PMD *Power Metering Devices*, instrumentos de medición de energía eléctrica con certificaciones de grado comercial.

PSO *Particle Swarm Optimization*, algoritmo algorítmico de Optimización por Enjambre de Partículas.

PWM *Pulse Width Modulation*, modulación por ancho de pulsos, utilizada para regular la intensidad del flash LED.

RESTful Arquitectura de *software* que usa métodos estándar para crear servicios web y transmitir datos.

RF Requisitos Funcionales del sistema informático y físico.

RNF Requisitos No Funcionales y restricciones operativas del sistema.

ROI *Region of Interest*, Región de Interés englobada durante el preprocesamiento de la fotografía.

SEC Superintendencia de Electricidad y Combustibles de Chile.

Smart Meter Medidor de energía inteligente con capacidad de transmisión remota bidireccional.

SoC *System on a Chip*, circuito integrado que encapsula todos los componentes de un sistema en un solo microchip.

SPI *Serial Peripheral Interface*, estándar de comunicación síncrona.

Tesseract Motor de código abierto altamente reconocido para realizar procesos OCR de forma local.

ThingSpeak Plataforma en la nube operada por MathWorks, utilizada para la agregación, visualización y análisis de datos IoT.

UXGA *Ultra Extended Graphics Array*, resolución fotográfica de 1600x1200 píxeles.

ZigBee Protocolo y estándar de comunicación inalámbrica orientada a la creación de redes en malla (*mesh*).

Índice general

1 Introducción **1**

1.1 Descripción del Problema 2

1.2 Objetivo General 4

1.3 Objetivos Específicos 5

1.4 Estructura del documento 6

2 Estado del Arte **9**

2.1 Temas relevantes a tratar 9

2.1.1 Medidores inteligentes 9

2.1.2 Captura de datos de energía 10

2.1.3 Monitoreo de Carga No Intrusivo (NILM) 10

2.1.4 Procesamiento en el Borde y Visión por Computadora 11

2.1.5 Protocolos de comunicación 12

2.1.6 Actividades de la vida diaria (ADL) 13

2.2 Estándar NILM 13

2.2.1 Componentes de un sistema NILM 14

2.2.2 Clasificación de dispositivos NILM (NSD) 14

2.2.3 Relación con equipos de medición tradicionales 15

2.3 Trabajos Relacionados 16

3 Propuesta **27**

3.1 Arquitectura 27

3.2 Análisis de las alternativas posibles de solución 28

3.2.1 Capa de captura de datos 28
3.2.2 Capa de Comunicación 34
3.2.3 Capa de Análisis de Datos 37
3.3 Alternativas seleccionadas para el proyecto 44
3.3.1 Sensores y Captura de Datos 45
3.3.2 Microcontrolador 46
3.3.3 Tecnologías de Comunicación 46
3.3.4 Plataforma de Middleware y Visualización 46
3.4 Requerimientos de la plataforma 47
3.4.1 Requisitos Funcionales 47
3.4.2 Requisitos no Funcionales 49

4 Implementación **51**
4.1 Entorno de Pruebas y Montaje General 51
4.1.1 Alimentación y Cableado 51
4.1.2 Protecciones Eléctricas 52
4.1.3 Equipos de Medición 52
4.1.4 Montaje Final del Tablero 54
4.2 Implementación del Prototipo Óptico (BPW34) 55
4.2.1 Acondicionamiento de Hardware y Circuitaje 55
4.2.2 Validaciones de Detección 56
4.2.3 Instalación sobre Medidor Eléctrico 59
4.2.4 Lógica del Firmware 60
4.3 Implementación del Prototipo de Visión Artificial (ESP32-CAM) . . . 61
4.3.1 Instalación Física, Enfoque y Control del Flash LED 61
4.3.2 Algoritmo de Preprocesamiento de Imagen y Tesseract OCR . 64
4.3.3 Lógica de Validación y Manejo de Errores (Modo Ráfaga) . . 65
4.3.4 Transmisión de la Imagen al Servidor Local 66
4.4 Comunicación con la Nube y Visualización 67

4.4.1 Configuración de Canales y Campos 67
4.4.2 Métricas e Indicadores de Actividad 67
4.4.3 Sistema de Alerta Integrado y Control Temporal 71

5 Resultados **73**
5.1 Análisis de Frecuencia de Muestreo 73
5.1.1 Frecuencia de muestreo: 20 segundos 74
5.1.2 Frecuencia de muestreo: 30 segundos 75
5.1.3 Frecuencia de muestreo: 60 segundos 76
5.2 Consumo Energético de los Prototipos 77
5.3 Pruebas de Estrés Lumínico 80
5.4 Sincronización y Desfase 81
5.5 Tasa de Éxito de Captura 81
5.6 Consumo de Ancho de Banda y Tráfico de Red 82
5.7 Detección de Actividades y Monitoreo en Entorno Real 83
5.7.1 Optimización de la Frecuencia de Captura en el Prototipo de Visión 86

6 Conclusiones y trabajo futuro **91**
6.1 Conclusiones 91
6.2 Trabajo Futuro 93

Bibliografía **96**

Anexos **103**

Índice de figuras

1.1 Esquema general del proyecto y flujo de información. 5

2.1 Comparación entre sistemas intrusivos (ILM) y no intrusivos (NILM). 11
2.2 Ejemplos de actividades de la vida diaria. 13
2.3 Ejemplo de implementación de un sistema NILM bajo el estándar IEC. 14
2.4 Esquema implementado por Iacobucci 2021. 17
2.5 Esquema implementado por Zhuang 2023. 18
2.6 Esquema implementado por Noury 2011. 19
2.7 Esquema implementado por Gao 2020. 20
2.8 Dispositivo Fludia FM432e IoT electricity consumption sensor. 25
2.9 Dispositivo Milesight X1 Sensing Camera. 26

3.1 Arquitectura general del sistema IoT. 28
3.2 Sensor PZEM-004T. 29
3.3 Modulo AFE ADE9153A. 30
3.4 Medidor Inteligente CERMI. 31
3.5 Fotodiodo BPW34. 32
3.6 Módulo ESP32-CAM. 33
3.7 Logo HTTP. 35
3.8 Logo LoRaWAN 35
3.9 Logo ZigBee. 36
3.10 Diagrama Edge-computing. 37
3.11 Modulo ESP32. 38

3.12 Diagrama Cloud-computing. 39
3.13 Logo AWS IoT. 40
3.14 Logo Microsoft Azure IoT Hub. 40
3.15 Logo IBM Watson IoT Platform. 41
3.16 Raspberry Pi 4. 43
3.17 Arquitectura con las alternativas seleccionadas para el sistema IoT. . . 44

4.1 Interruptor termomagnético (16A) y diferencial (25A) montados en riel DIN. 52
4.2 Medidor LEXO DDS608 (patrón certificado SEC) junto al medidor electrónico auxiliar. 53
4.3 Multímetro digital TRUPER-830 utilizado para las mediciones de hardware. 53
4.4 Vista general del tablero de pruebas finalizado. 54
4.5 Diagrama eléctrico del tablero de pruebas finalizado. 55
4.6 Diagrama de conexión del sensor BPW34 y ESP32. 56
4.7 Implementación física del prototipo óptico montado. 56
4.8 Prueba de saturación del fotodiodo mediante estimulación con linterna. 57
4.9 Comportamiento del fotodiodo ante distintas frecuencias de estimulación. 58
4.10 Validación del contador de pulsos versus pulso emisor a 8 Hz. 58
4.11 Encapsulado para garantizar un aislamiento óptico absoluto. 59
4.12 Instalación física y acople óptico del fotodiodo sobre el medidor eléctrico. 60
4.13 ESP32-CAM montada sobre su base adaptadora. 62
4.14 Montaje de la ESP32-CAM frente al medidor utilizando brazo articulado a 25 cm de distancia. 63
4.15 Interfaz de selección de la Región de Interés (ROI) sobre los dígitos del medidor. 64

4.16 Procesamiento de imagen: binarización e inversión de colores previa a la lectura OCR. 64
4.17 Visualización en tiempo real del consumo eléctrico de las últimas 24 horas. 68
4.18 Consumo por hora versus consumo basal del hogar. 68
4.19 Indicador binario de actividad en el hogar basado en el consumo basal. 69
4.20 Mapa de calor (Heatmap) que ilustra los patrones de consumo de los últimos 7 días. 70
4.21 Comparativa del perfil de consumo por hora de las últimas 48 horas. . 70
4.22 Comparativa extendida del perfil de consumo por hora de las últimas 72 horas. 71

5.1 Registro comparativo a 20s de muestreo y carga de 2000W. 74
5.2 Registro comparativo a 20s de muestreo y carga de 3000W. 74
5.3 Registro comparativo a 30s de muestreo y carga de 2000W. 75
5.4 Registro comparativo a 30s de muestreo y carga de 3000W. 75
5.5 Registro comparativo a 60s de muestreo y carga de 2000W. 76
5.6 Registro comparativo a 60s de muestreo y carga de 3000W. 76
5.7 Esquema del banco de pruebas: multímetro conectado en serie en la *protoboard* mediante la interrupción del cable USB. 78
5.8 Registro del consumo de corriente (mA) para los tres estados operativos del prototipo óptico. 79
5.9 Registro del consumo de corriente (mA) para los tres estados operativos del prototipo de visión. 80
5.10 Reflejo producido sobre el medidor. 81
5.11 Mapa de calor del consumo eléctrico por hora en vivienda de ocupación única. 84
5.12 Franjas horarias de inactividad asociadas al ciclo de descanso. 85

5.13 Detección de ausencia prolongada en la vivienda durante los fines de semana. 86
5.14 Comparativa de detección de actividad entre ambos prototipos. 88
5.15 Comparativa de lecturas válidas versus esperadas para frecuencias de 60, 30 y 15 minutos. 89

Índice de cuadros

2.1 Comparación de protocolos de comunicación IoT. 12
2.2 Clasificación de NSD según la frecuencia de muestreo. 15
2.3 Clasificación de NSD según la tasa de salida de datos. 15
2.4 Clasificación de NSD según la tasa de bits. 15
2.5 Matriz comparativa de trabajos relacionados. 23
2.6 Comparación de Alternativas Comerciales para Monitoreo. 24

3.1 Comparación de alternativas para captura de datos 33
3.2 Comparación de alternativas para comunicación 36
3.3 Comparativa de Plataformas Cloud para Procesamiento Total 42
3.4 Comparación de alternativas para análisis de datos 43

Índice de códigos

6.1 Código fuente para la prueba de frecuencia y saturación del fotodiodo. 103
6.2 Configuración Inicial y Setup del ESP32 para Prototipo Óptico 104
6.3 Gestión de conexión WiFi, sincronización NTP y lectura del BPW34 . 105
6.4 Lógica de almacenamiento en búfer circular offline 105
6.5 Transmisión dosificada de datos a ThingSpeak mediante HTTP 106
6.6 Servidor web, control PWM de flash y captura en ESP32-CAM. . . . 107
6.7 Configuración general, parámetros de red y variables de entorno para OCR. 109
6.8 Funciones de captura, validación estadística y persistencia de estado. . 109
6.9 Rutina de calibración manual, selección de ROI y recuperación de estado previo. 112
6.10 Bucle principal de captura periódica y validación de saltos en la lectura del medidor. 114
6.11 Lógica de control de errores, activación del modo ráfaga y temporización del ciclo. 115
6.12 Script de MATLAB para la detección de inactividad y envío de alertas temporizadas. 116

Capítulo 1

Introducción

El envejecimiento de la población es una tendencia demográfica global irreversible, y Chile se proyecta como una de las naciones más envejecidas de América Latina hacia el año 2050, momento en el cual más del 30 % de su población tendrá 60 años o más [1]. Este fenómeno conlleva múltiples desafíos, particularmente en el ámbito de la salud y el cuidado a largo plazo, exigiendo soluciones innovadoras que permitan a los adultos mayores mantener su independencia de forma segura.

En este contexto, la tendencia en servicios médicos es migrar desde un modelo centrado en el hospital hacia uno centrado en el hogar [2]. Sin embargo, las alternativas tecnológicas tradicionales, como las cámaras de vigilancia indoor o los sensores portátiles (wearables), suelen enfrentar un alto rechazo por parte de los usuarios debido a su nivel de intrusión, complejidad de uso y el estigma de sentirse vigilados. Como respuesta, los datos de consumo energético capturados desde el medidor del hogar abren una oportunidad para la monitorización remota de forma pasiva, discreta y de bajo costo.

El Monitoreo de Carga No Intrusivo (NILM) permite inferir las Actividades de la Vida Diaria (AVD) de los residentes al analizar los patrones de uso eléctrico, detectando desviaciones de la rutina que pueden indicar emergencias o deterioro en la salud [3]. Aprovechando el despliegue de medidores electrónicos e inteligentes en el país

[4, 5, 6], este trabajo se enfoca en la etapa más crítica de este ecosistema: la captura confiable del dato de consumo sin intervenir la red eléctrica.

Para ello, esta investigación diseña, implementa y compara empíricamente dos metodologías de adquisición de datos de bajo costo: un sistema de visión por computadora mediante una ESP32-CAM y procesamiento OCR, frente a un sistema de detección de pulsos luminosos utilizando un fotodiodo BPW34 con procesamiento *Full Edge*. El propósito es determinar la solución tecnológica óptima en términos de precisión, uso de ancho de banda e intrusividad para habilitar sistemas de teleasistencia escalables.

1.1 Descripción del Problema

El deseo de la mayoría de los adultos mayores es envejecer en sus propios hogares. Sin embargo, vivir solos incrementa significativamente el riesgo asociado a caídas, accidentes domésticos o el agravamiento silencioso de enfermedades crónicas, situaciones que pueden pasar desapercibidas durante días si no existe una supervisión constante. Ante la inviabilidad económica de contar con cuidadores presenciales 24/7 y el rechazo generalizado hacia los sistemas de monitoreo invasivos, la inferencia de rutinas a través del consumo eléctrico se presenta como una alternativa altamente prometedora.

A pesar de que el marco teórico del Monitoreo de Carga No Intrusivo (NILM) está bien establecido desde los trabajos pioneros de Hart [7] y ha demostrado su utilidad teórica en la detección de eventos [8], la implementación física y real de la captura de datos en viviendas existentes presenta importantes barreras de ingeniería. Los enfoques tradicionales de NILM requieren frecuencias de muestreo elevadas [9] y un procesamiento intensivo en la nube para desagregar electrodomésticos [10, 11]. Más problemático aún, la obtención de datos eléctricos ricos en detalle suele requerir la instalación de sensores de corriente (CT) y de voltaje directamente en el tablero eléctrico principal. Esta intervención directa en el cableado de 220V representa un riesgo de seguridad, requiere personal certificado para su instalación y eleva los costos del sistema, limitando severamente su escalabilidad y adopción masiva.

Por lo tanto, la problemática central radica en **cómo capturar el consumo eléctrico agregado de manera precisa, segura, económica y 100 % no intrusiva**, desde el exterior del medidor, para alimentar plataformas de teleasistencia.

Ante este desafío, surgen metodologías de captura externas. Una opción es la captura óptica del display del medidor mediante visión por computadora. Si bien es intuitiva, esta alternativa enfrenta problemas prácticos en entornos reales: la calidad de la lectura es altamente sensible a las variaciones lumínicas ambientales, la captura intermitente de fotografías es propensa a desincronizaciones frente a cambios dinámicos del medidor, y el procesamiento de las imágenes (OCR) demanda altos recursos computacionales y ancho de banda, requiriendo externalizar el cálculo hacia un servidor o un PC local en lugar de procesar en el propio dispositivo (*Edge*).

Una segunda alternativa es la detección de los pulsos luminosos emitidos por el diodo LED presente en la inmensa mayoría de los medidores electrónicos (por ejemplo, a razón de 1600 imp/kWh). Aunque conceptualmente más simple, existe una brecha en la literatura respecto a la validación empírica y la cuantificación del error a largo plazo de este método frente a soluciones basadas en visión, especialmente considerando las métricas de carga de red, latencia y viabilidad de ejecutar algoritmos *Full Edge* en microcontroladores de bajo costo.

En consecuencia, es fundamental realizar un estudio comparativo, en ambiente de laboratorio controlado y aislado, que evalúe el desempeño real de la extracción de datos mediante visión artificial versus la detección de pulsos. A través de una validación empírica del hardware y de las arquitecturas de software asociadas, será posible establecer qué tecnología garantiza la confiabilidad y autonomía necesarias para sostener un sistema de monitoreo que cautele la vida de personas mayores de forma ininterrumpida.

1.2 Objetivo General

El objetivo general de este proyecto es diseñar, implementar y evaluar experimentalmente un sistema IoT de bajo costo y completamente no intrusivo para la captura de datos de consumo eléctrico residencial, orientado a la inferencia de patrones de actividad para el monitoreo remoto y el cuidado de adultos mayores que viven solos.

Para cumplir con esta meta, la investigación se centra en realizar una comparativa empírica rigurosa entre dos metodologías de adquisición de datos de borde (*Edge*): por un lado, un sistema de visión por computadora basado en una ESP32-CAM y procesamiento local con Tesseract OCR; por otro lado, un sistema de detección directa de pulsos luminosos mediante un fotodiodo BPW34 acoplado a un microcontrolador ESP32 operando en modo *Full Edge*.

A través de ensayos controlados en un laboratorio doméstico aislado y protegido, el proyecto busca determinar la tecnología óptima en términos de precisión de lectura, baja intrusividad, minimización de carga en la red WiFi y estabilidad a largo plazo. La información de potencia y consumo acumulado obtenida es transmitida mediante el protocolo HTTP hacia la plataforma en la nube ThingSpeak, permitiendo estructurar un esquema de visualización accesible para familiares o cuidadores y sentando las bases para el análisis retrospectivo de rutinas cotidianas sin vulnerar la privacidad del usuario.

En la Figura 1.1 se presenta un diagrama que ilustra el objetivo general del proyecto. El esquema abarca desde el entorno del hogar y la captura del consumo eléctrico agregado mediante las alternativas de hardware evaluadas, pasando por la extracción y el procesamiento de los datos en el borde, hasta su almacenamiento y visualización en la plataforma en la nube. El sistema propuesto constituye una solución integral de adquisición no intrusiva orientada a revelar patrones de actividad y regularidad en las rutinas de personas mayores, facilitando la supervisión remota de su bienestar.

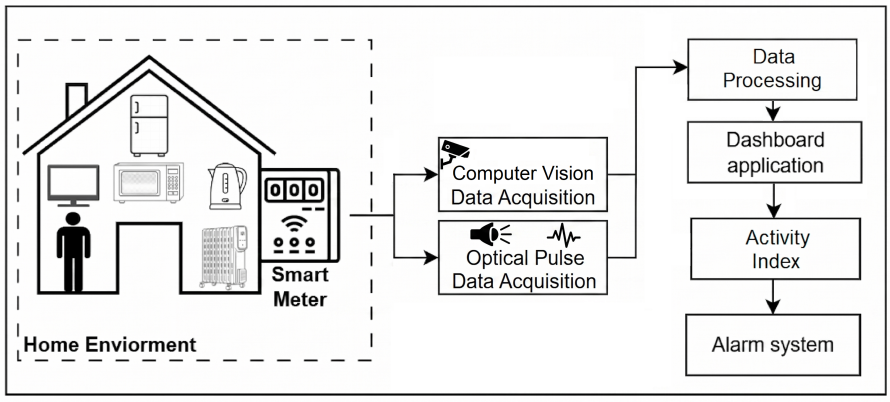


Figura 1.1: Esquema general del proyecto y flujo de información.

1.3 Objetivos Específicos

Para alcanzar el objetivo general planteado, se establecen los siguientes objetivos específicos:

- **Diseñar e implementar dos prototipos de captura no intrusiva:** Construir un módulo basado en ESP32-CAM para la captura fotográfica del medidor y otro módulo basado en un microcontrolador ESP32 acoplado a un fotodiodo BPW34 para el registro de pulsos luminosos.

- **Desarrollar la arquitectura de procesamiento y comunicación de datos:** Programar un script en Python con Tesseract OCR y algoritmos de umbralización para el procesamiento de imágenes en un servidor local, y desarrollar el firmware *Full Edge* para el conteo de pulsos en el ESP32, estableciendo para ambos casos la transmisión de datos hacia la plataforma ThingSpeak mediante protocolos HTTP/WiFi.

- **Diseñar un entorno de pruebas eléctrico aislado y seguro:** Montar un laboratorio de ensayos doméstico protegido con interruptor termomagnético y diferencial, que incluya un medidor digital multifunción y el medidor electrónico objetivo, garantizando un entorno controlado para las mediciones.

- **Evaluar y comparar empíricamente el rendimiento de los prototipos:** Ejecutar pruebas de carga controlada y ensayos de larga duración (48 horas) para medir y contrastar la tasa de acierto, el error acumulado, el ancho de banda consumido y la carga sobre la red WiFi de ambas tecnologías.
- **Validar la aplicabilidad de los datos para la inferencia de rutinas:** Demostrar, a partir del prototipo ganador, que la información de consumo agregado obtenida permite generar gráficas representativas que faciliten la identificación de la actividad diaria del usuario en la plataforma en la nube.

1.4 Estructura del documento

El presente documento se organiza en seis capítulos, que se describen a continuación:

- **Capítulo 1 – Introducción:** Presenta el contexto demográfico del envejecimiento poblacional en Chile y la consiguiente necesidad de desarrollar sistemas de teleasistencia y monitoreo remoto no intrusivo para adultos mayores que viven solos. Se expone la problemática central relacionada con la captura segura y económica de los datos de consumo eléctrico residencial, evitando intervenciones riesgosas. Además, se detallan los objetivos generales y específicos que guían la investigación, enfocados en la comparación de metodologías de adquisición en el borde (*Edge*), finalizando con la estructura del documento.
- **Capítulo 2 – Estado del Arte:** Revisa los conceptos fundamentales vinculados a la medición inteligente y al Monitoreo de Carga No Intrusivo (NILM), estableciendo el marco teórico necesario para el desarrollo del proyecto. Incluye una revisión exhaustiva de las tecnologías de captura de datos de energía, los protocolos de comunicación IoT aplicables y el estándar internacional que rige estos sistemas. Del mismo modo, se discuten trabajos relacionados y soluciones comerciales en el ámbito del cuidado de la salud (*Ambient Assisted Living*), analizando sus arquitecturas, ventajas y limitaciones prácticas en escenarios reales.

- **Capítulo 3 – Propuesta:** Describe detalladamente el diseño de la arquitectura IoT planteada para el sistema, estructurada en capas de captura de datos, comunicación y análisis. Se presenta un análisis de las diversas alternativas tecnológicas disponibles para cada estrato, justificando la selección de la ESP32-CAM y el fotodiodo BPW34 como las opciones más viables para la percepción no intrusiva. Finalmente, se define el trabajo a realizar mediante la especificación de los requisitos funcionales y de interfaces que aseguran la operatividad, seguridad y correcta integración del sistema propuesto.

- **Capítulo 4 – Implementación:** Expone el desarrollo técnico y experimental del proyecto, detallando la construcción física y la programación de los dos prototipos de captura de datos. Se describe el montaje del entorno de pruebas aislado y seguro, así como la codificación del firmware para la detección de pulsos luminosos en modalidad *Full Edge* y la configuración del servidor local con Tesseract OCR para la visión por computadora. También se explica la integración de la transmisión de datos vía WiFi hacia la plataforma en la nube ThingSpeak.

- **Capítulo 5 – Resultados:** Expone el análisis empírico y el rendimiento obtenido al someter ambos prototipos a pruebas estandarizadas de frecuencia de muestreo, consumo energético y estrés lumínico. Se detallan las métricas cuantitativas de sincronización y desfase a corto plazo (48 horas), así como la tasa de éxito de captura frente a las transiciones mecánicas del medidor. Asimismo, se cuantifica y contrasta el impacto en el ancho de banda y tráfico de red generado por cada arquitectura de medición.

- **Capítulo 6 – Conclusiones y trabajo futuro:** Sintetiza los resultados obtenidos tras la evaluación empírica de ambos prototipos, destacando la superioridad técnica, menor intrusividad y eficiencia en red del fotodiodo frente a la cámara. Se discute el cumplimiento de los objetivos planteados y la validación de la hipótesis al lograr inferir rutinas diarias mediante el consumo eléctrico agregado.

Para concluir, se proponen líneas de investigación y mejoras futuras orientadas a escalar el sistema hacia escenarios comunitarios y optimizar el análisis de datos a largo plazo para el cuidado de adultos mayores.

Capítulo 2

Estado del Arte

2.1 Temas relevantes a tratar

2.1.1 Medidores inteligentes

Los medidores inteligentes son dispositivos electrónicos capaces de registrar el consumo eléctrico en intervalos de tiempo cortos (desde segundos hasta minutos) y de transmitir esta información de manera remota y bidireccional. Esto los diferencia de los medidores tradicionales, ya que permiten habilitar servicios avanzados como la lectura automática, la gestión de la demanda, el corte y reposición a distancia y la aplicación de esquemas tarifarios dinámicos.

En Chile, diversas compañías distribuidoras han comenzado a implementar esta tecnología. Por ejemplo, CGE dispone de un sistema de medición inteligente que permite la lectura y control remoto de los equipos instalados en sus clientes residenciales [5]. Por su parte, Enel Distribución ha incorporado el modelo de medidor inteligente CERM1, que registra y transmite información de consumo mediante concentradores y redes celulares, habilitando funcionalidades como monitoreo en línea y facturación avanzada [6].

En la Región de Valparaíso, la empresa Chilquinta Energía desarrolla pilotos de medidores inteligentes basados en tecnología de comunicación RF punto–multipunto.

Estos equipos, fabricados por Aclara, permiten registrar el consumo cada 15 minutos, realizar cortes remotos y entregar información detallada de calidad de energía, constituyendo un referente clave para la implementación de soluciones de monitoreo masivo [12].

2.1.2 Captura de datos de energía

Es el proceso mediante el cual se recolectan los parámetros eléctricos del medidor (voltaje, corriente, potencia activa/reactiva, consumo acumulado, entre otros). Tradicionalmente, la frecuencia de muestreo determina el nivel de detalle con el que es posible caracterizar los electrodomésticos y, por ende, la precisión en la desagregación de carga. Este concepto es crítico, ya que un muestreo bajo puede ser insuficiente para distinguir dispositivos, mientras que un muestreo alto aumenta la carga de transmisión y almacenamiento.

En el contexto de soluciones 100 % no intrusivas orientadas a medidores ya instalados, la captura de datos se traslada al exterior del equipo. Las aproximaciones contemporáneas evitan la intervención de la red eléctrica de 220V mediante métodos ópticos, tales como la lectura de los pulsos luminosos emitidos por el diodo LED del medidor (ej. 1600 imp/kWh) utilizando fotodiodos, o la captura fotográfica del display digital para extraer los valores de consumo acumulado mediante técnicas de visión por computadora.

2.1.3 Monitoreo de Carga No Intrusivo (NILM)

El *Non-Intrusive Load Monitoring* (NILM) es una técnica que permite descomponer la señal total de consumo eléctrico de una vivienda en las contribuciones individuales de los aparatos conectados. Su origen se remonta al trabajo pionero de Hart [7], y desde entonces ha evolucionado hacia modelos que emplean procesamiento de señales y algoritmos de extracción de características para identificar las firmas de cada dispositivo [8]. A diferencia de los métodos intrusivos, el NILM requiere únicamente

un punto de medición en la entrada, como se muestra en la figura 2.1, reduciendo costos y complejidad. Su importancia en este trabajo radica en la posibilidad de inferir actividades de la vida diaria (ADL) de los adultos mayores y detectar cambios en sus rutinas de forma pasiva.

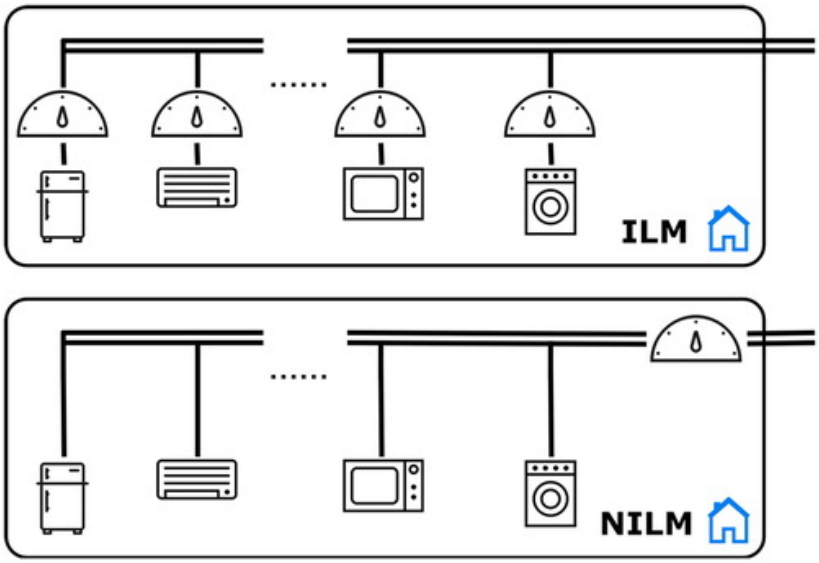


Figura 2.1: Comparación entre sistemas intrusivos (ILM) y no intrusivos (NILM).

2.1.4 Procesamiento en el Borde y Visión por Computadora

Históricamente, el NILM dependía de potentes servidores en la nube para ejecutar redes neuronales complejas. Sin embargo, la tendencia actual es el procesamiento en el borde (*Edge Computing*), que traslada el cálculo directamente al microcontrolador local. Esto minimiza el ancho de banda utilizado y protege la privacidad del usuario, ya que solo se transmiten los valores finales procesados. En paralelo, para la extracción de datos desde medidores legados o digitales sin puertos de comunicación abiertos, se emplean técnicas de visión por computadora como el Reconocimiento Óptico de Caracteres (OCR, por sus siglas en inglés). Herramientas como Tesseract OCR, combinadas con algoritmos de umbralización, permiten digitalizar los displays mecánicos o LCD en entornos locales, actuando como un puente entre la infraestructura eléctrica existente y las plataformas IoT modernas.

2.1.5 Protocolos de comunicación

Conjunto de reglas que definen cómo se transmiten los datos extraídos hacia servidores locales o en la nube. En el contexto de IoT residencial para teleasistencia, los protocolos más utilizados se comparan en el cuadro 2.1

Cuadro 2.1: Comparación de protocolos de comunicación IoT.

Protocolo	**Frecuencia / Banda**	**Alcance aprox.**	**Tasa de datos**	**Notas relevantes**
WiFi (IEEE 802.11)	2,4 GHz / 5 GHz	50–100 m	Hasta ∼600 Mbps	Ideal para ambientes domésticos con alto ancho de banda.
Bluetooth 4.2	2,5 GHz	50–150 m	∼1 Mbps	Bajo consumo, útil para dispositivos personales.
Zigbee	2,4 GHz	10–100 m	∼250 Kbps	Red mesh, bajo consumo para sensores domésticos.
LoRaWAN	Diversas bandas ISM	2,5 km urbano / 15 km rural	0,3 a 50 Kbps	Alcance largo con bajo consumo.

La selección del protocolo (típicamente WiFi combinado con HTTP o MQTT) incide directamente en la latencia, la carga de la red del usuario y la viabilidad de enviar imágenes pesadas versus variables numéricas ligeras.

2.1.6 Actividades de la vida diaria (ADL)

Conjunto de rutinas básicas que reflejan la autonomía funcional de una persona, tales como alimentarse, dormir, cocinar o mantener la higiene personal. La monitorización de estas actividades mediante las variaciones de consumo eléctrico abre la posibilidad de detectar desviaciones significativas que puedan indicar deterioro cognitivo, problemas de salud física o alteraciones emocionales en adultos mayores que viven solos.



Figura 2.2: Ejemplos de actividades de la vida diaria.

2.2 Estándar NILM

El estándar técnico **IEC TS 63297:2021** [13] establece un marco de referencia para la clasificación de los *NILM Sensing Devices* (NSD), es decir, los dispositivos de sensado empleados en sistemas de Monitoreo de Carga No Intrusivo. Su propósito no es definir algoritmos de desagregación, sino proporcionar un lenguaje común que permita caracterizar, comparar e integrar de manera uniforme diferentes soluciones de hardware.

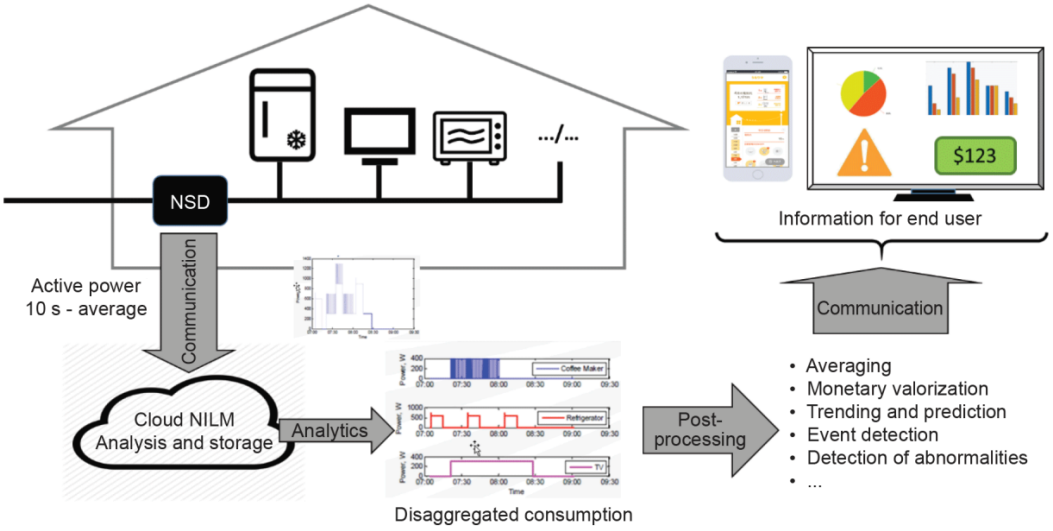


Figura 2.3: Ejemplo de implementación de un sistema NILM bajo el estándar IEC.

2.2.1 Componentes de un sistema NILM

De acuerdo con el estándar, un sistema NILM se compone de dos elementos principales:

- **NILM Sensing Device (NSD):** dispositivo conectado a la instalación eléctrica que genera datos relevantes que pueden ser utilizados por NILM Analytics.
- **NILM Analytics:** conjunto de algoritmos que procesan los datos producidos por el NSD para entregar información sobre el comportamiento energético.

2.2.2 Clasificación de dispositivos NILM (NSD)

La IEC TS 63297 clasifica los NSD en función de tres parámetros esenciales: **frecuencia de muestreo**, **tasa de salida de datos** y **tasa de bits**. Cada uno se organiza en clases, lo que permite describir un dispositivo (como la ESP32-CAM o el módulo BPW34) de manera precisa y comparable.

Cuadro 2.2: Clasificación de NSD según la frecuencia de muestreo.

Clase	**Frecuencia de muestreo (f)**
P	Parámetros estimados (sin forma de onda)
1	$f < 5$ kHz
2	$5\,\text{kHz} \leq f < 8\,\text{kHz}$

Cuadro 2.3: Clasificación de NSD según la tasa de salida de datos.

Clase	**Tasa de salida de datos (d)**
E	$d > 30$ min
D	$30\,\text{min} \geq d > 1\,\text{min}$
C	$1\,\text{min} \geq d > 1\,\text{s}$

Cuadro 2.4: Clasificación de NSD según la tasa de bits.

Clase	**Tasa de bits (b)**
L	$b < 100$ bps
M	$100\,\text{bps} \leq b < 1\,\text{kbps}$
H	$1\,\text{kbps} \leq b < 10\,\text{kbps}$

2.2.3 Relación con equipos de medición tradicionales

El estándar también establece que, si bien los NSD producen información valiosa para la desagregación de cargas, **no deben considerarse instrumentos de medición certificados**. Dispositivos como *Power Metering Devices* (PMD) cumplen normas estrictas, mientras que los prototipos NSD enfocados en teleasistencia pueden entregar estimaciones de actividad útiles sin requerir dichas certificaciones de grado comercial.

2.3 Trabajos Relacionados

A continuación, se revisan diversos estudios y proyectos que abordan el monitoreo de actividades, la detección de eventos y el bienestar de adultos mayores utilizando datos de consumo eléctrico, estableciendo la base empírica y metodológica sobre la cual se construye esta propuesta.

El trabajo [9] analiza el efecto de la resolución temporal de los datos en la detección de eventos dentro de sistemas NILM. El estudio determina qué frecuencias de muestreo son suficientes para identificar correctamente cambios de estado, un aspecto crítico para aplicaciones de asistencia. Los hallazgos revelan una relación no lineal entre la tasa de muestreo y la precisión, donde tasas alrededor de 1 kHz resultan suficientes, mientras que reducciones extremas deterioran la capacidad de detección. Sin embargo, menores tasas de muestreo reducen el tiempo de cómputo y las necesidades de almacenamiento, justificando la decisión de optimizar la frecuencia de captura (ya sea mediante conteo de pulsos o intervalos de captura de imágenes) en la presente tesis para no saturar los microcontroladores.

Por su parte, el estudio [10] aborda directamente el uso de Non-Intrusive Load Monitoring aplicado al cuidado de personas mayores, proponiendo una arquitectura híbrida edge-cloud. En este esquema, el procesamiento de eventos se realiza localmente, mientras que la identificación y análisis se llevan a cabo en la nube. Destaca por su enfoque en la reducción de ancho de banda, transmitiendo únicamente ventanas de datos asociadas a eventos, protegiendo la privacidad. La inferencia de Actividades de la Vida Diaria (ADL) y la visualización remota mediante ThingSpeak prueban la viabilidad de plataformas IoT ligeras. Las limitaciones señaladas por los autores respecto a la dependencia parcial de la nube fundamentan directamente el objetivo de esta investigación de procesar las variables íntegramente de manera local (mediante herramientas como Tesseract OCR o detección Full Edge) antes de su transmisión.

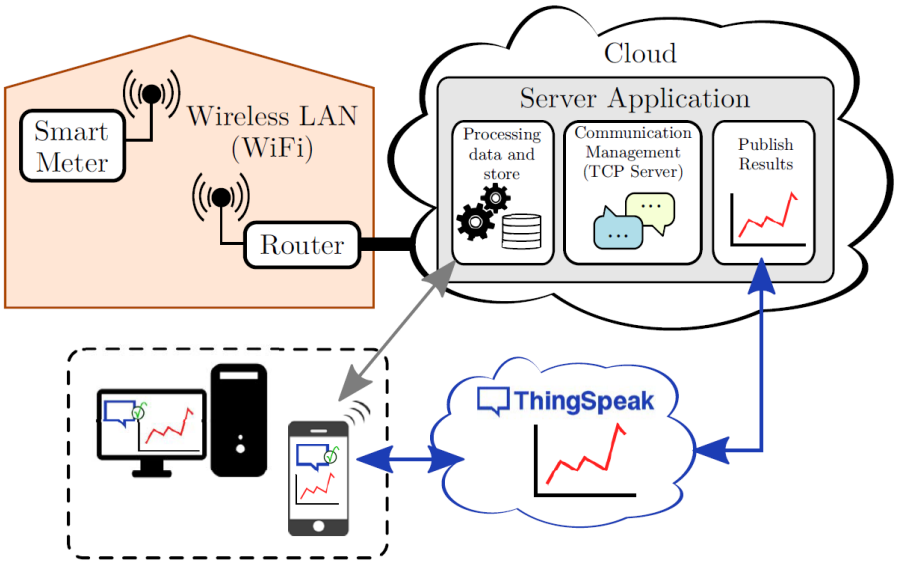


Figura 2.4: Esquema implementado por Iacobucci 2021.

El artículo [11] introduce un modelo de NILM basado en la combinación de Redes Neuronales Artificiales (ANN) y el algoritmo de Optimización por Enjambre de Partículas (PSO). Utilizando datasets públicos (REDD, UK-DALE e IAWE) y métricas de evaluación rigurosas (MAE y NRMSE), el enfoque logra alta precisión para la detección del uso de electrodomésticos, permitiendo inferir patrones de comportamiento. No obstante, la elevada demanda computacional de estos modelos híbridos en la nube refuerza la necesidad de investigar alternativas de captura continua que sean computacionalmente económicas en el hardware terminal.

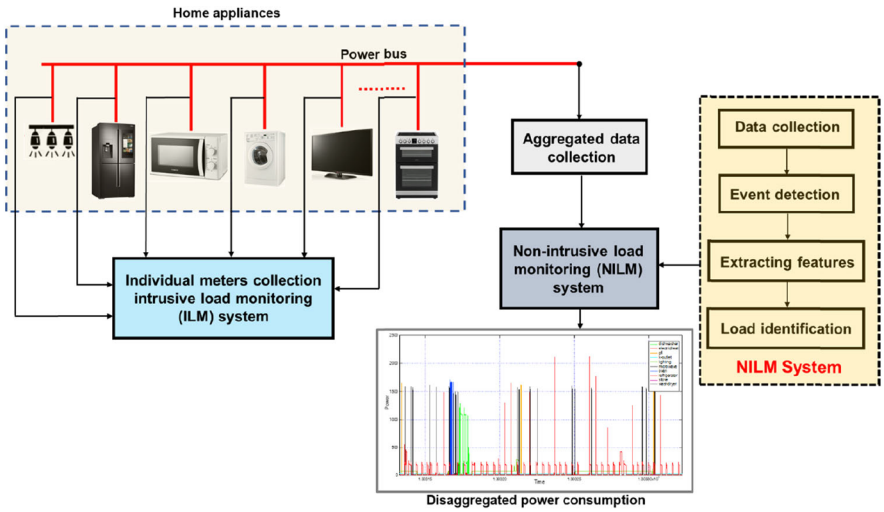


Figura 2.5: Esquema implementado por Zhuang 2023.

En el ámbito aplicado, la investigación [14] constituye una referencia fundamental al describir un sistema pionero de monitorización no intrusiva de la actividad cotidiana en adultos mayores. El sistema incorpora un sensor WPC (*Wireless Powerline Communication*, una tecnología híbrida patentada por WATTECO que transmite señales de radio a través de la red eléctrica existente) en el cuadro eléctrico, una matriz de ponderación, algoritmos de clustering k-means y modelos difusos para obtener un índice global de actividad diaria (I_{day}). Sus resultados, validados en entornos reales durante meses, demuestran la viabilidad técnica de usar datos eléctricos como proxy del comportamiento y la utilidad de un índice sintético para apoyar la teleasistencia. Al mismo tiempo, evidencia la necesidad de buscar métodos de captura externos más simples que eviten la intervención directa de la instalación domiciliaria.

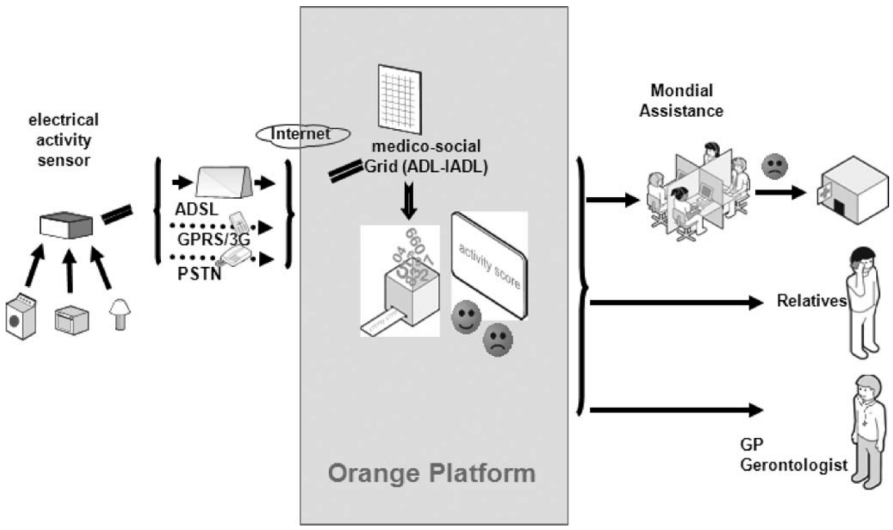


Figura 2.6: Esquema implementado por Noury 2011.

El trabajo [15] propone un método de NILM de alta precisión y baja complejidad, ideal para aplicaciones en tiempo real y con recursos limitados. Incorpora descomposición wavelet y una característica novedosa de *overshoot multiple* acoplada a un clasificador KNN ponderado. Este enfoque logra tiempos de cómputo mínimos (0.06 s para 201 muestras), habilitando aplicaciones en tiempo real con bajo costo de sensores. Esta filosofía de optimización de recursos apoya sólidamente la viabilidad de algoritmos locales en plataformas como el ESP32, donde la eficiencia algorítmica es crítica.

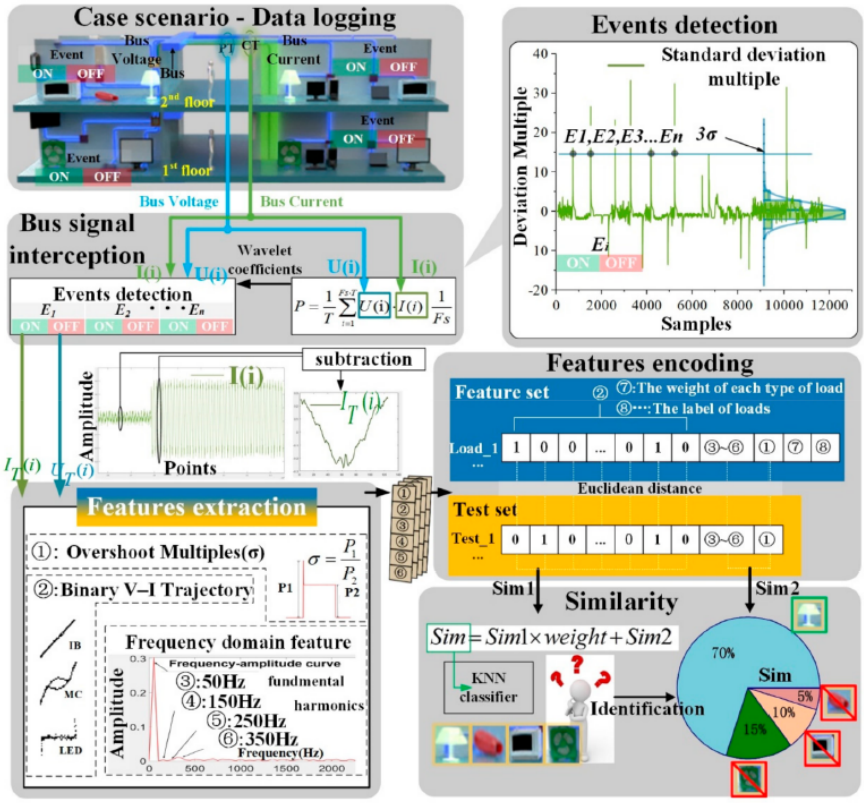


Figura 2.7: Esquema implementado por Gao 2020.

En la misma línea de optimización, el estudio [16] presenta un generador de alarmas de comportamiento a corto plazo basado en datos de un medidor inteligente comercial. Los autores evalúan cuatro enfoques de aprendizaje automático (LSTM, CNN, Random Forest y Decision Tree) para predecir el consumo energético horario. La detección de anomalías se produce cuando la medición difiere estadísticamente de la predicción, siendo la red LSTM la de mejor rendimiento. Este trabajo es fundamental porque valida la generación de alarmas utilizando datos de muy baja resolución temporal, alineándose con las capacidades de captura óptica local propuestas en esta tesis.

Profundizando en dicha metodología, la investigación posterior [17] propone un sistema de dos etapas utilizando medidores comerciales. La primera etapa utiliza una red neuronal para la identificación de siete Actividades de la Vida Diaria, mientras que

la segunda evalúa la “normalidad” creando un perfil estadístico basado en la hora de inicio, duración y actividad previa. Esta arquitectura entrega un método efectivo para cuantificar la regularidad del comportamiento, proporcionando un marco analítico interpretable.

El artículo [18] es altamente relevante para la capa de captura, este emplea un enfoque no intrusivo mediante una baliza que se acopla ópticamente al puerto del medidor eléctrico. Este dispositivo lee datos cada 15 minutos sin intervenir la red. La validación de una lectura óptica externa respalda directamente la hipótesis de utilizar el diodo LED de consumo como mecanismo robusto y seguro para la teleasistencia.

Desde la perspectiva de la protección de datos, el trabajo [19] aborda el conflicto entre precisión y privacidad al emplear deliberadamente datos de baja resolución (15 minutos). Mediante el establecimiento de un umbral base, el sistema genera un patrón binario de actividad para calcular índices de regularidad de forma computacionalmente ligera. Esta simplicidad algorítmica es de alto valor para una implementación de análisis en el propio borde (Edge), requiriendo recursos mínimos.

En cuanto a la estructuración de estas soluciones, [20] analiza la complejidad arquitectónica de los sistemas eHealth. El documento enfatiza la división en capas de recolección, procesamiento en el borde (Fog Processing) y centralizado, destacando la necesidad de estándares de interoperabilidad para integrar estas mediciones en el sistema de salud.

Finalmente, el reporte [21] ofrece una advertencia crítica sobre la viabilidad social al evaluar un proyecto piloto que fracasó por barreras humanas y no técnicas. El documento detalla que la profunda desconfianza en la privacidad, el temor al costo energético y el estigma social impidieron la adopción. Para esta investigación, subraya la obligación ineludible de priorizar soluciones arquitectónicas, como el procesamiento total en el microcontrolador y la transmisión de variables abstractas en lugar de imágenes crudas, que ofrezcan garantías comprobables de no invasividad y respeto por la autonomía del adulto mayor.

Matriz Comparativa de Soluciones y Enfoques NILM

La siguiente matriz consolida el análisis de los trabajos relacionados, clasificándolos según el método de adquisición física, la arquitectura de procesamiento y las estrategias de privacidad. Esta comparativa fundamenta la pertinencia de desarrollar un sistema de captura óptica (como el uso de fotodiodos o el motor Tesseract OCR) que opere de manera *Full Edge*, mitigando los riesgos de adopción documentados en implementaciones previas.

Cuadro 2.5: Matriz comparativa de trabajos relacionados.

Ref.	Tecnología de Captura	Procesamiento	Privacidad y Transmisión
[9]	Dispositivo DAQ de alta velocidad.	Local (Edge).	**Alta.** Usa baja resolución para reducir almacenamiento.
[10]	AFE conectado a red eléctrica (4 kHz).	Híbrido (Edge/Cloud).	**Alta.** Solo transmite ventanas de datos de eventos detectados.
[11]	Medidor inteligente (potencias y armónicos).	Nube (Redes híbridas).	**Baja.** Exige transmisión constante de la forma de onda.
[14]	Sensor WPC intrusivo en el cuadro eléctrico.	Nube (Orange Labs).	**Media.** Transmite índices diarios agregados.
[15]	Smart metering ($\sim$1 kHz).	Local (Edge).	**Alta.** Detección *in situ* con bajo coste computacional.
[16]	Medidor comercial Wibeee (1 hora).	Nube.	**Media.** Uso de datos de muy baja resolución temporal.
[17]	Medidores comerciales.	Nube.	**Media.** Anonimización mediante perfiles estadísticos.
[18]	Baliza óptica externa acoplada al medidor.	Nube.	**Media-Alta.** Lectura óptica aislada sin intervenir la red.
[19]	Medidor inteligente (15 minutos).	Local (Edge).	**Muy Alta.** Transforma consumo a un patrón binario (1/0).
[20]	Revisión teórica de arquitecturas eHealth.	Fog y Cloud.	**N/A.** Enfatiza estándares de seguridad (HL7 FHIR).
[21]	Evaluación de piloto comercial.	N/A.	**Crítica.** Miedo a la vigilancia como barrera de adopción.

Alternativas presentes en el mercado

El monitoreo de consumo eléctrico y su aplicación en el cuidado de adultos mayores (*Ambient Assisted Living*) cuenta con diversas soluciones comerciales. Analizar estas alternativas permite dimensionar los costos reales de implementación y los enfoques tecnológicos (intrusivos, no intrusivos, ópticos o basados en cámaras) disponibles actualmente en la industria.

Cuadro 2.6: Comparación de Alternativas Comerciales para Monitoreo.

Compañía / Disp.	**Tecnología de Monitoreo**	**Enfoque en Cuidado / Aplicación**	**Precio Referencia (CLP)**
Voltaware	NILM: Sensor único en el cuadro eléctrico con desagregación mediante IA.	Solución *Elderly Care* dedicada. Detección de anomalías en rutinas.	$184.000 [22]
Fludia FM432e	Sensor óptico IoT (LoRaWAN) acoplado al LED del medidor.	Lectura de pulsos de consumo sin intervención eléctrica intrusiva.	$180.000 [23]
Milesight X1	Cámara sensora con procesamiento de visión artificial.	Digitalización visual de consumos en medidores análogos o digitales.	$84.990 [24]
Howz	Híbrido (ILM + Sensores): Kit de movimiento, puertas y enchufes.	Foco en aprender rutinas mediante la fusión de datos de múltiples sensores.	$250.000 [25]
Smilics Wibeee	NILM: Hardware *clip-on* en el cuadro eléctrico. Muestreo en alta frecuencia.	Inferencia de rutinas y detección de inactividad para el bienestar.	$250.000 [26]

La Tabla 2.6 consolida las principales alternativas comerciales analizadas, integrando sus tecnologías de monitoreo, su enfoque hacia el bienestar del adulto mayor y sus precios referenciales de mercado. Las alternativas mas relevantes para este trabajo son:

- **Fludia FM432e (IoT LoRaWAN):** Un sensor óptico diseñado para leer los pulsos del diodo emisor de luz (LED) del medidor eléctrico. Representa la versión comercial del enfoque de conteo de pulsos, ofreciendo transmisión inalámbrica de bajo consumo, pero con un costo de adquisición elevado para despliegues comunitarios masivos.



Figura 2.8: Dispositivo Fludia FM432e IoT electricity consumption sensor.

- **Milesight X1 Sensing Camera:** Una cámara sensora con visión artificial incorporada, diseñada específicamente para capturar y digitalizar los valores de los medidores. Es el equivalente comercial directo al uso de módulos de captura fotográfica con reconocimiento óptico de caracteres (OCR), demostrando el interés de la industria por capturar el dato visual directamente en el borde (*Edge*) sin intervenir la circuitería.



Figura 2.9: Dispositivo Milesight X1 Sensing Camera.

Capítulo 3

Propuesta

3.1 Arquitectura

El diseño de un sistema de monitoreo estructurado sigue una cadena lógica que va desde el hardware de medición hasta la interfaz del usuario final. El proceso se inicia en la capa de captura de datos, donde un dispositivo sensor obtiene la señal de la vivienda. Este sensor necesita estar conectado a un microcontrolador (MCU) o un dispositivo de cómputo en el borde (Edge).

Aquí es donde se produce la primera integración crucial: el dispositivo MCU actúa como el anfitrión de la capa de comunicación. El MCU tiene la doble responsabilidad de leer los datos del sensor de captura y empaquetar esos datos para su transmisión usando un protocolo de red. Finalmente, los datos son recibidos por la capa de análisis de datos, que actúa como el cerebro del sistema, ejecutando los algoritmos y alimentando las interfaces para permitir a los cuidadores monitorear el bienestar del adulto mayor.

En la Figura 3.1 se presenta la arquitectura general del sistema IoT, estructurada en cuatro capas: captura de datos, comunicación, middleware y aplicación, integrando las principales alternativas tecnológicas consideradas para cada una de ellas.

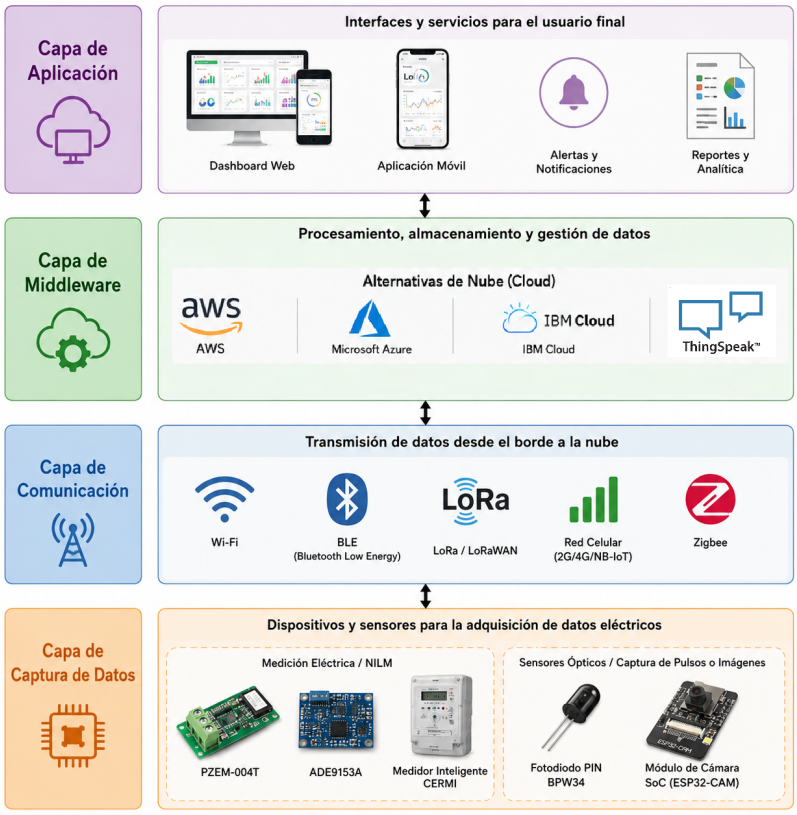


Figura 3.1: Arquitectura general del sistema IoT.

3.2 Análisis de las alternativas posibles de solución

3.2.1 Capa de captura de datos

La capa de captura de datos es responsable de la medición y adquisición de los parámetros eléctricos en el punto de entrada de la vivienda. Esta capa constituye la base del sistema NILM, ya que la calidad y frecuencia de los datos capturados determinan directamente la precisión en la identificación de actividades. Según el estándar IEC 63297, los dispositivos de esta capa se clasifican según su frecuencia de muestreo, tasa de salida de datos y tasa de bits.

Alternativa 1: PZEM-004T [27]

Esta alternativa utiliza el sensor PZEM-004T, un dispositivo económico ampliamente utilizado en proyectos IoT. Es ideal para implementaciones iniciales y prototipos, ofreciendo un balance aceptable entre costo y funcionalidad. Su principal ventaja radica en la facilidad de integración con plataformas como Arduino y ESP32, permitiendo despliegues rápidos en viviendas individuales.

Especificaciones Técnicas:

- **Dispositivo**: Sensor PZEM-004T
- **Precisión**: Clase 1.0
- **Rango Voltaje**: 80-260V AC
- **Rango Corriente**: 0-100A
- **Rango Frecuencia**: 45-65 Hz
- **Frecuencia refresco de datos**: ∼1-10 Hz
- **Resolución ADC**: 12 bits
- **Comunicación**: TTL Serial, Modbus
- **Parámetros**: Voltaje RMS, Corriente RMS, Potencia Activa, Energía
- **Costo**: $18.900 CLP [28]



Figura 3.2: Sensor PZEM-004T.

Alternativa 2: ADE9153A [29]

El AFE ADE9153A representa una solución profesional para aplicaciones que requieren alta precisión y capacidades avanzadas de análisis. Esta alternativa es adecuada para implementaciones comunitarias donde la calidad de datos es crítica, permitiendo el monitoreo de armónicos y calidad de energía para una desagregación más precisa.

Especificaciones Técnicas:

- **Dispositivo**: AFE ADE9153A + Microcontrolador
- **Precisión**: Clase 0.5
- **Rango Voltaje**: 120-480V AC
- **Rango Corriente**: 0-200A
- **Frecuencia Muestreo**: 4 kHz
- **Resolución ADC**: 24 bits Σ-Δ
- **Comunicación**: SPI, I2C
- **Parámetros**: V/I RMS, P/Q/S, Frecuencia, FP, Armónicos hasta 40º
- **Costo**: $176.960 CLP [30]

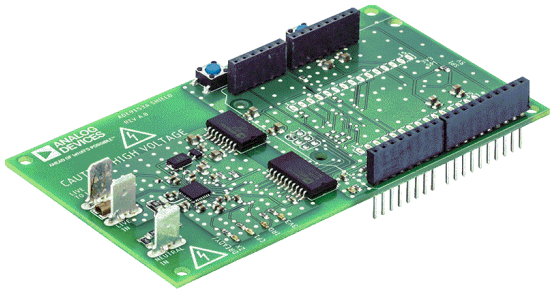


Figura 3.3: Modulo AFE ADE9153A.

Alternativa 3: Medidor Inteligente Comercial CERMI [31]

Esta alternativa aprovecha la infraestructura existente de medidores inteligentes desplegados por compañías distribuidoras como Enel y CGE. Es la opción más viable para implementaciones a escala comunitaria, ya que elimina costos de hardware y aprovecha dispositivos ya certificados y mantenidos.

Especificaciones Técnicas:

- **Dispositivo**: Medidor CERMI (Enel) o similar
- **Precisión**: Clase 0.5S
- **Rango Voltaje**: 120-240V AC
- **Rango Corriente**: 0-200A
- **Frecuencia Muestreo**: 1-15 minutos (configurable)
- **Comunicación**: PLC, Cellular, RF
- **Parámetros**: Consumo energético, calidad básica de energía
- **Costo**: Incluido en despliegue masivo



Figura 3.4: Medidor Inteligente CERMI.

Alternativa 4: Fotodiodo PIN de Silicio (BPW34) [32]

Para mejorar la precisión y velocidad en la lectura de pulsos ópticos, se plantea el uso de un fotodiodo PIN de silicio de alta velocidad en un empaque plástico miniatura. Su construcción plana lo hace ideal como reemplazo de dispositivos TO-5 en múltiples aplicaciones.

Especificaciones Técnicas:

- **Dispositivo**: Fotodiodo BPW34.
- **Área sensible**: Cuenta con un área radiante sensible grande de $7{,}5\ mm^2$.
- **Ángulo de visión**: Amplio ángulo de media sensibilidad de $\pm 65°$.
- **Espectro**: Adecuado para radiación visible e infrarroja cercana (600 a 1050 nm).
- **Tiempos de respuesta**: Rápido tiempo de subida y bajada de 100 ns.
- **Costo**: $2.400 CLP[33].

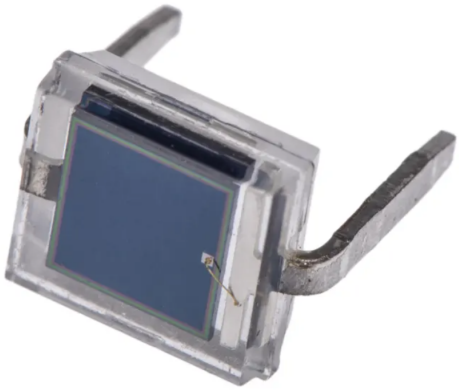


Figura 3.5: Fotodiodo BPW34.

Alternativa 5: Módulo de Cámara SoC (ESP32-CAM)[34]

Una alternativa no intrusiva enfocada en la captura visual directa de los caracteres del medidor para su posterior procesamiento. Funciona como un sistema mínimo de huella ultra reducida.

Especificaciones Técnicas:

- **Dispositivo**: Módulo ESP32-CAM con CPU de 32 bits de bajo consumo.
- **Dimensiones**: Formato compacto de $27 \times 40{,}5 \times 4{,}5$ mm.
- **Cámara y hardware**: Soporta cámaras OV2640 y OV7670, incorpora lámpara flash y soporta tarjeta TF.
- **Memoria**: SRAM de 520 KB incorporada y PSRAM externa de 4M.
- **Costo**: $12.271 CLP[35].

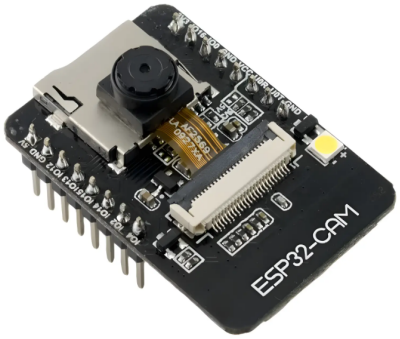


Figura 3.6: Módulo ESP32-CAM.

Cuadro 3.1: Comparación de alternativas para captura de datos

Parámetro	**PZEM-004T**	**ADE9153A**	**Medidor Comercial**	**BPW34**	**ESP32-CAM**
Precisión	Clase 1.0	Clase 0.5	Clase 0.5S	Dependiente del ADC	Visual (OV2640 / OV7670)
Rango Voltaje	80-260V AC	120-480V AC	120-240V AC	No aplica	No aplica (Opera a 5V)
Rango Corriente	0-100A	0-200A	0-200A	No aplica	No aplica
Frec. Muestreo	∼1-10 Hz	4 kHz	1-15 min	Tiempos de respuesta 100 ns	Variable (por procesamien-to)
Comunicación	TTL Serial, Modbus	SPI, I2C	PLC, Cellular, RF	Señal Analógica	Inalámbrica (Wi-Fi/BT)
Costo	$18.900 CLP	$176.960 CLP	Incluido	$2.400 CLP	$12.271 CLP

3.2.2 Capa de Comunicación

La capa de comunicación gestiona la transmisión de datos desde los dispositivos de captura hasta los sistemas de procesamiento. Esta capa es crítica para la escalabilidad del sistema, especialmente en entornos comunitarios donde múltiples dispositivos deben comunicarse simultáneamente. La selección del protocolo impacta directamente en el consumo energético, latencia y confiabilidad del sistema.

Alternativa 1: WiFi + HTTP [36]

Esta alternativa utiliza redes WiFi existentes combinadas con el protocolo HTTP (Hypertext Transfer Protocol) mediante arquitecturas RESTful. Es una opción altamente accesible e ideal para viviendas individuales con infraestructura WiFi estable, ya que permite enviar peticiones directa y sencillamente a servidores o endpoints locales/remotos mediante métodos estándar (GET/POST) sin requerir un intermediario (*broker*).

Especificaciones Técnicas:

- **Protocolo**: HTTP / HTTPS sobre WiFi
- **Componente**: ESP32-WROOM-32
- **Tasa Datos**: Hasta 54 Mbps
- **Alcance**: 50-100 m
- **Consumo**: 100-300mA (medio-alto)
- **Latencia**: 100-500 ms
- **Costo Módulo**: $11.990 CLP
- **Seguridad**: WPA2, TLS/SSL (HTTPS)



Figura 3.7: Logo HTTP.

Alternativa 2: LoRaWAN [37]

LoRaWAN es una tecnología de área amplia de bajo consumo, diseñada específicamente para aplicaciones IoT. Esta alternativa es ideal para entornos rurales o implementaciones comunitarias extensas donde la cobertura WiFi es limitada. Su bajo consumo permite operación por batería por largos períodos.

Especificaciones Técnicas:

- **Protocolo**: LoRaWAN
- **Componente**: RAK3172 [38] + Gateway LPS8N [39]
- **Tasa Datos**: 0.3-50 kbps
- **Alcance**: 2-15 km
- **Consumo**: 10-50mA (muy bajo)
- **Latencia**: 1-5 segundos
- **Costo Módulo**: $230.000 CLP
- **Seguridad**: AES 128-bit



Figura 3.8: Logo LoRaWAN

Alternativa 3: ZigBee Mesh [40]

ZigBee ofrece una solución de red mesh auto-configurable, ideal para condominios y edificios donde los dispositivos pueden actuar como repetidores. Esta tecnología es

robusta para entornos comunitarios densos, permitiendo comunicación confiable entre múltiples unidades.

Especificaciones Técnicas:

- **Protocolo**: ZigBee 3.0
- **Componente**: XBee Series 3 [41]
- **Tasa Datos**: 250 kbps
- **Alcance**: 10-100 m (extensible con mesh)
- **Consumo**: 20-40mA (bajo)
- **Latencia**: 10-100 ms
- **Costo Módulo**: $28.990 CLP
- **Seguridad**: AES 128-bit



Figura 3.9: Logo ZigBee.

Cuadro 3.2: Comparación de alternativas para comunicación

Parámetro	**WiFi + HTTP**	**LoRaWAN**	**ZigBee**
Protocolo	HTTP sobre WiFi	LoRaWAN	ZigBee 3.0
Componentes	ESP32-WROOM	RAK3172 + Gateway LPS8N	XBee Series 3
Tasa Datos	Hasta 54 Mbps	0.3-50 kbps	250 kbps
Alcance	50-100 m	2-15 km	10-100 m
Consumo	Medio-Alto	Muy Bajo	Bajo
Latencia	100-500 ms	1-5 segundos	10-100 ms
Costo	$11.990 CLP	$230.000 CLP	$28.990 CLP

3.2.3 Capa de Análisis de Datos

La capa de análisis procesa los datos recibidos para realizar la inferencia actividades. Esta capa puede implementarse en diferentes arquitecturas según los requerimientos de escalabilidad, privacidad y capacidad de procesamiento. La elección de arquitectura es particularmente importante al escalar de viviendas individuales a entornos comunitarios.

Alternativa 1: Arquitectura Edge-Cloud Híbrida [42]

Esta arquitectura balancea el procesamiento entre el borde (edge) y la nube (cloud), detectando eventos localmente y realizando la desagregación completa en la nube. Es ideal para implementaciones comunitarias donde se busca optimizar el ancho de banda manteniendo capacidades avanzadas de análisis.



Figura 3.10: Diagrama Edge-computing.

Especificaciones Técnicas:

- **Arquitectura**: Híbrida Edge-Cloud
- **Edge**: ESP32 [43] + detección de eventos
- **Cloud**: AWS IoT + Lambda + LSTM
- **Algoritmo**: LSTM 3 capas
- **Latencia**: 100-500 ms
- **Costo Mensual**: $5.000-20.000 CLP
- **Escalabilidad**: Alta
- **Ancho Banda**: 10-100 KB/día

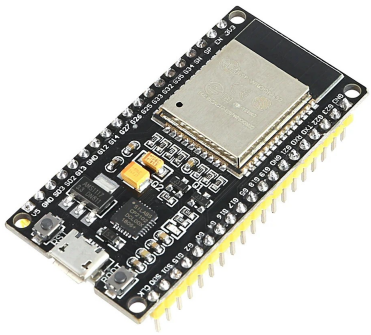


Figura 3.11: Modulo ESP32.

Alternativa 2: Procesamiento Total en Cloud

En esta arquitectura, todos los datos se envían a la cloud para procesamiento completo. Ofrece máxima capacidad de procesamiento y es fácil de escalar, pero requiere mayor ancho de banda y depende completamente de la conectividad. Esta aproximación es ideal para implementaciones comunitarias donde se prioriza la capacidad de procesamiento sobre consideraciones de ancho de banda.



Figura 3.12: Diagrama Cloud-computing.

AWS IoT Suite [44] La plataforma AWS ofrece un ecosistema completo para IoT con servicios gestionados que simplifican la implementación. AWS IoT Core permite la conexión segura de millones de dispositivos, mientras que Lambda y SageMaker facilitan el procesamiento y entrenamiento de modelos de machine learning.

Especificaciones Técnicas:

- **Servicio IoT**: AWS IoT Core
- **Procesamiento**: AWS Lambda (Python 3.9, 2GB RAM)
- **ML**: Amazon SageMaker para entrenamiento de modelos
- **Almacenamiento**: Amazon Timestream (series temporales)
- **Escalabilidad**: Auto-scaling automático
- **Costo Estimado**: $15.000-40.000 CLP/mes para 100 dispositivos
- **Ventajas**: Ecosistema completo, alta confiabilidad
- **Desventajas**: Costo puede escalar rápidamente

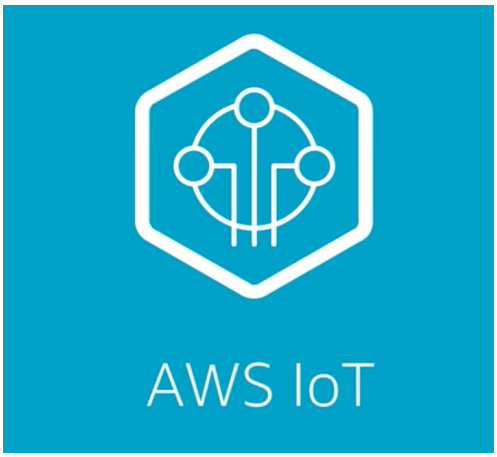


Figura 3.13: Logo AWS IoT.

Microsoft Azure IoT Hub [45] Azure proporciona una integración robusta con servicios de analytics y machine learning, destacando por su integración con Power BI para visualización y Azure Machine Learning para el desarrollo de modelos NILM.

Especificaciones Técnicas:

- **Servicio IoT**: Azure IoT Hub
- **Procesamiento**: Azure Functions
- **ML**: Azure Machine Learning
- **Almacenamiento**: Azure Data Explorer
- **Visualización**: Power BI integrado
- **Costo Estimado**: $12.000-35.000 CLP/mes para 100 dispositivos
- **Ventajas**: Integración con herramientas Microsoft, buena documentación
- **Desventajas**: Curva de aprendizaje más pronunciada



Figura 3.14: Logo Microsoft Azure IoT Hub.

IBM Watson IoT Platform [46] IBM ofrece capacidades avanzadas de analytics y AI con Watson, ideal para aplicaciones que requieren procesamiento complejo de datos y análisis predictivo del comportamiento de adultos mayores.

Especificaciones Técnicas:

- **Servicio IoT**: IBM Watson IoT Platform
- **Procesamiento**: IBM Cloud Functions
- **ML**: Watson Studio para modelos NILM
- **Almacenamiento**: Cloudant NoSQL
- **Analytics**: IBM Streaming Analytics
- **Costo Estimado**: $18.000-45.000 CLP/mes para 100 dispositivos
- **Ventajas**: Capacidades avanzadas de AI, análisis predictivo
- **Desventajas**: Mayor costo, ecosistema más cerrado



Figura 3.15: Logo IBM Watson IoT Platform.

Cuadro 3.3: Comparativa de Plataformas Cloud para Procesamiento Total

Parámetro	**AWS IoT Suite**	**Microsoft Azure**	**IBM Watson**
Servicio IoT	IoT Core	IoT Hub	Watson IoT Platform
Procesamiento	Lambda	Azure Functions	Cloud Functions
ML/AI	SageMaker	Azure ML	Watson Studio
Almacenamiento	Timestream	Data Explorer	Cloudant
Visualización	QuickSight	Power BI	Cognos Analytics
Costo 100 dispositivos	$15-40k CLP	$12-35k CLP	$18-45k CLP
Escalabilidad	Muy Alta	Muy Alta	Alta
Latencia	500-1500 ms	600-1800 ms	800-2000 ms
Integración ML	Excelente	Muy Buena	Excelente
Facilidad Uso	Media	Media-Alta	Media-Baja
Seguridad	IAM + KMS	Azure AD + Key Vault	App ID + Key Protect
Ventajas	Ecosistema completo	Integración Microsoft	AI avanzada
Desventajas	Costo escalable	Curva aprendizaje	Mayor costo

Alternativa 3: Procesamiento Local con Sincronización Cloud [47]

Esta alternativa prioriza la privacidad y autonomía, realizando todo el procesamiento localmente y sincronizando solo resultados con la cloud. Es adecuada para viviendas individuales o entornos con preocupaciones de privacidad.

Especificaciones Técnicas:

- **Arquitectura**: Local con sync Cloud
- **Dispositivo**: Raspberry Pi 4
- **Costo Dispositivo**: $116.990 CLP

- **Algoritmo**: KNN ponderado
- **Latencia**: 10-50 ms
- **Costo Mensual**: $1.000-5.000 CLP
- **Escalabilidad**: Media
- **Ancho Banda**: 1-10 KB/día



Figura 3.16: Raspberry Pi 4.

Cuadro 3.4: Comparación de alternativas para análisis de datos

Parámetro	**Edge-Cloud**	**Cloud Total**	**Local + Cloud**
Arquitectura	Híbrida	Total Cloud	Local con sync
Procesamiento	Edge: eventos, Cloud: LSTM	Cloud completo	Local completo
Algoritmo	LSTM	ANN-PSO	KNN ponderado
Latencia	100-500 ms	500-2000 ms	10-50 ms
Costo Mensual	$5-20k CLP	$10-50k CLP	$1-5k CLP
Escalabilidad	Alta	Muy Alta	Media
Privacidad	Media	Baja	Alta

3.3 Alternativas seleccionadas para el proyecto

Tras el análisis de las diversas opciones y considerando los objetivos principales del proyecto, se han seleccionado los siguientes componentes. La selección prioriza la no intrusividad y la facilidad de implementación para el prototipo individual, factores críticos para la aceptación del usuario [21]. Dado que el alcance no requiere la desagregación de cargas, sino el monitoreo de patrones de actividad agregados, se prefieren soluciones que operen con una baja tasa de datos [19].

En la Figura 3.17 se presenta la arquitectura del sistema IoT propuesta, estructurada en cuatro capas: captura de datos, comunicación, middleware y aplicación, incorporando las alternativas tecnológicas seleccionadas para cada una de ellas.

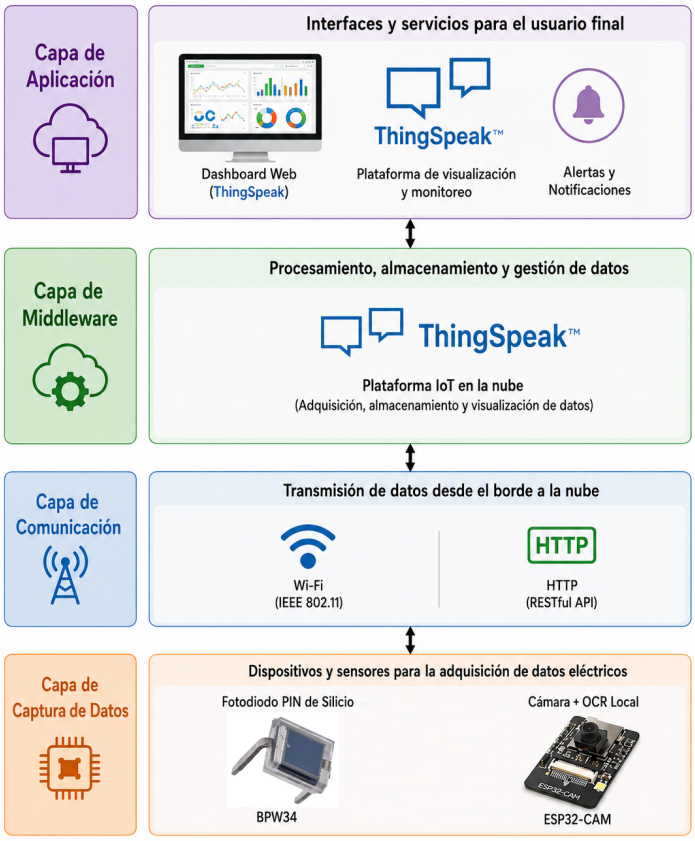


Figura 3.17: Arquitectura con las alternativas seleccionadas para el sistema IoT.

3.3.1 Sensores y Captura de Datos

Para abordar el estudio de caso de la vivienda individual, se comparan enfoques de hardware que miden o infieren el consumo eléctrico de forma distinta, buscando la opción más viable para el prototipo de validación.

Alternativa 1: Fotodiodo PIN de Silicio (BPW34) [32, 33]

Este enfoque utiliza un fotodiodo de alta velocidad para captar los pulsos ópticos del medidor e inferir el consumo energético. Su principal fortaleza radica en la no intrusividad total, ya que no requiere ninguna intervención eléctrica, lo que elimina riesgos y simplifica drásticamente la instalación en el tablero domiciliario. Además, posee alta sensibilidad y facilidad de montaje gracias a su gran área activa ($7{,}5\ mm^2$) y amplio ángulo de visión ($\pm 65°$), facilitando la captura de pulsos sin requerir una alineación milimétrica perfecta. El costo del componente es extremadamente bajo, ideal para mantener el presupuesto reducido en el desarrollo del prototipo. Finalmente, produce una señal de baja resolución que resulta eficiente y suficiente para inferir patrones de actividad diaria [19].

Mientras que la principal limitación de esta alternativa es su tasa de muestreo variable, pues la frecuencia del dato de potencia depende directamente del consumo en tiempo real; a muy bajo consumo, la recolección de datos disminuye considerablemente. Asimismo, la precisión de la medición final está sujeta completamente a la constante de impulsos (por ejemplo, 1000 imp/kWh) del medidor físico existente.

Alternativa 2: Cámara + OCR Local (ESP32-CAM) [34, 35]

Este enfoque utiliza visión por computador para fotografiar el display numérico del medidor, representando una solución no intrusiva enfocada en obtener una lectura de energía absoluta. Este método destaca por obtener una lectura absoluta (kWh), leyendo el valor exacto y acumulado de la pantalla del medidor para proporcionar una fuente de verdad directa y robusta. Otra gran ventaja es la privacidad e independencia

tecnológica, dado que el procesamiento de imágenes se realiza utilizando la librería Tesseract OCR de manera local. Esto evita el pago recurrente o la dependencia de servicios de visión computacional externos.

Por otro lado, presenta una alta complejidad de configuración, requiriendo el ajuste meticuloso de algoritmos de visión y el manejo de scripts locales en Python. Asimismo, la captura fotográfica demanda un mayor consumo de recursos energéticos y de capacidad de procesamiento en comparación con el fotodiodo. Finalmente, el sistema posee sensibilidad ambiental, siendo vulnerable a interferencias por cambios en la iluminación, acumulación de polvo o reflejos directos sobre el acrílico del medidor.

3.3.2 Microcontrolador

Arquitectura ESP32: Se selecciona esta familia de SoCs por su potente integración y bajo costo. La arquitectura combina una excelente capacidad de procesamiento con conectividad WiFi nativa. Esto permite estandarizar el núcleo del proyecto, siendo el hardware capaz tanto de gestionar las interrupciones rápidas del fotodiodo BPW34, como de manejar y transmitir las capturas de imágenes al utilizar el módulo ESP32-CAM.

3.3.3 Tecnologías de Comunicación

WiFi (IEEE 802.11) + HTTP [36]: Esta combinación se selecciona por su facilidad de implementación directa para el caso de estudio. El uso de WiFi permite aprovechar la red del domicilio sin necesidad de añadir gateways extras. La adopción del protocolo HTTP mediante peticiones RESTful simplifica enormemente el flujo de datos.

3.3.4 Plataforma de Middleware y Visualización

ThingSpeak [48]: Para la fase de prototipado, se selecciona la plataforma ThingSpeak de MathWorks de forma exclusiva. Esta herramienta es ideal para la validación rápida,

ya que se integra fluidamente con las peticiones API HTTP despachadas por la capa de comunicación y entrega herramientas de visualización inmediatas. Esto permite generar las gráficas de consumo requeridas sin la complejidad técnica y el tiempo asociado a levantar y configurar múltiples servicios web paralelos.

3.4 Requerimientos de la plataforma

La implementación de este sistema de monitoreo debe estar alineada con los objetivos del proyecto y con las restricciones propias de una medición no intrusiva en entornos domiciliarios. Al basarse en un procesamiento computacional en el borde (*Edge Computing*), resulta esencial establecer con claridad los requerimientos específicos para adecuar la captura y transmisión de datos.

El diseño debe garantizar una captura de datos que no intervenga el cableado eléctrico, operando con total autonomía energética mediante el uso de baterías debido a la habitual ausencia de tomacorrientes en las cercanías de los medidores. Asimismo, la extracción de información debe ejecutarse de forma local para minimizar el ancho de banda y resguardar la privacidad del entorno.

Bajo este contexto, a continuación se definen los requerimientos del sistema, separados en funcionales y no funcionales.

3.4.1 Requisitos Funcionales

Los requisitos funcionales definen el comportamiento explícito del sistema, detallando las acciones precisas que los componentes deben ejecutar para lograr la medición de energía. Han sido redactados de manera atómica para ser directamente verificables durante la etapa de pruebas:

Prototipo basado en Visión (OCR):

RF1 El módulo ESP32-CAM debe capturar fotografías nítidas del display numérico del medidor eléctrico.

RF2 El módulo de cámara debe subir la fotografía capturada al servidor local para su posterior análisis.

RF3 El script de procesamiento debe reconocer la lectura numérica ejecutando un OCR.

RF4 El sistema debe subir el dato de energía numérico ya procesado a la plataforma en la nube.

Prototipo Óptico (Fotodiodo):

RF5 El fotodiodo BPW34 debe detectar las variaciones lumínicas (pulsos) emitidas por el LED del medidor eléctrico.

RF6 El microcontrolador debe contabilizar los pulsos acumulados para calcular el consumo energético correspondiente.

RF7 El sistema debe enviar el valor de energía calculado a la plataforma en la nube una vez transcurrido el intervalo de tiempo preestablecido.

Requisitos Generales de Operación y Visualización:

RF8 Todo el hardware de captura, procesamiento y transmisión debe operar de forma continua alimentado exclusivamente por baterías.

RF9 El microcontrolador debe sincronizar su reloj interno mediante un servidor NTP (*Network Time Protocol*) al establecer conexión, garantizando que cada lectura enviada cuente con una marca de tiempo exacta.

RF10 El sistema debe ejecutar una rutina de reconexión automática a la red WiFi en caso de pérdida de señal, asegurando la reanudación de la transmisión de datos sin requerir intervención manual.

RF11 La plataforma de destino debe graficar de forma automatizada el consumo eléctrico correspondiente a cada intervalo de muestreo preestablecido.

RF12 El sistema debe proporcionar una visualización de datos históricos, aplicando un filtro que permita observar el consumo de energía durante los últimos 7 días.

RF13 La interfaz gráfica debe ser capaz de procesar las lecturas para resaltar los picos de consumo, diferenciando visualmente la actividad real del consumo basal de la vivienda.

3.4.2 Requisitos no Funcionales

Los requisitos no funcionales del sistema definen los atributos de calidad, restricciones de diseño y estándares operativos bajo los cuales el usuario y el entorno juzgan la operación del sistema:

RNF1 El sistema debe ser completamente no intrusivo, operando de manera externa sin conexión galvánica, modificación de circuitos ni intervención física en el cuadro eléctrico de la vivienda.

RNF2 El procesamiento de los datos debe realizarse de forma local (*Edge Computing*) para evitar la transmisión de fotografías crudas a internet, garantizando así la privacidad del entorno residencial.

RNF3 La comunicación inalámbrica debe utilizar el estándar WiFi (IEEE 802.11) por su disponibilidad nativa en los microcontroladores seleccionados y en entornos domiciliarios.

RNF4 La plataforma de visualización en la nube (ThingSpeak) debe presentar los datos históricos y en tiempo real de forma clara a través de gráficos temporales.

Requisitos de Interfaces

Respecto a los requisitos de interfaz del sistema, estos se abordan en tres niveles fundamentales: la interacción física de los sensores (hardware), la transmisión inalámbrica (lógica/red) y la presentación visual de la información (capa de aplicación).

A nivel de hardware, el fotodiodo requerirá una interfaz de pines GPIO configurados con interrupciones para asegurar la captura de pulsos rápidos en el microcontrolador. Por su parte, el módulo de cámara utilizará la interfaz de bus paralelo interna de la placa ESP32-CAM para transferir la información gráfica capturada del entorno exterior hacia la memoria antes de su procesamiento local.

En cuanto a las interfaces lógicas para la transmisión de datos, el protocolo base a utilizar es WiFi (IEEE 802.11), estableciendo el envío de paquetes de datos a través de peticiones HTTP (RESTful) cada vez que se cumpla el período de tiempo de muestreo establecido. En escenarios donde la red WiFi no esté disponible, la interfaz lógica habilitará de forma automática el almacenamiento temporal en el búfer interno del microcontrolador hasta que se recupere el enlace.

Finalmente, en la capa de aplicación, la interfaz de usuario (UI) debe priorizar una visualización de datos que sea clara y altamente intuitiva. Dado el propósito de monitoreo del sistema, la plataforma no debe limitarse a mostrar gráficas de energía bruta, sino que debe implementar indicadores visuales de actividad real. Esta distinción es crítica, ya que la mera existencia de consumo eléctrico no implica necesariamente actividad en la vivienda, debido a la presencia de cargas basales o electrodomésticos que consumen energía de forma continua o cíclica durante todo el día (como es el caso de un refrigerador). Por ende, la interfaz debe ser capaz de resaltar las variaciones y picos de consumo que representen interacciones directas, traduciendo los datos eléctricos en información útil sobre las rutinas y el bienestar del usuario.

Capítulo 4

Implementación

4.1 Entorno de Pruebas y Montaje General

Para llevar a cabo las validaciones prácticas de los prototipos, se construyó un laboratorio de pruebas a escala montado sobre una plancha de madera MDF. Este tablero simula las condiciones reales de un empalme y circuito eléctrico domiciliario bajo la normativa chilena, permitiendo conectar cargas reales y realizar mediciones de forma segura y controlada. El montaje eléctrico se organizó sobre un riel DIN estándar, integrando los siguientes componentes:

4.1.1 Alimentación y Cableado

Para la energización de todo el sistema desde la red domiciliaria, se dotó al inicio del circuito de un enchufe macho de 16 amperes. El cableado estructurado del tablero se realizó utilizando conductores de 4 mm^2 de sección transversal, respetando rigurosamente el código de colores normativo: rojo para la fase, blanco para el neutro y verde para la tierra de protección. Este cableado distribuye la energía desde las protecciones hasta finalizar en una serie de enchufes hembra de superficie, permitiendo conectar distintos electrodomésticos para simular el consumo dinámico de un hogar.

4.1.2 Protecciones Eléctricas

Para garantizar la seguridad del circuito y del operador durante las pruebas, se instaló un interruptor termomagnético de 16 amperes y un interruptor diferencial de 25 amperes.

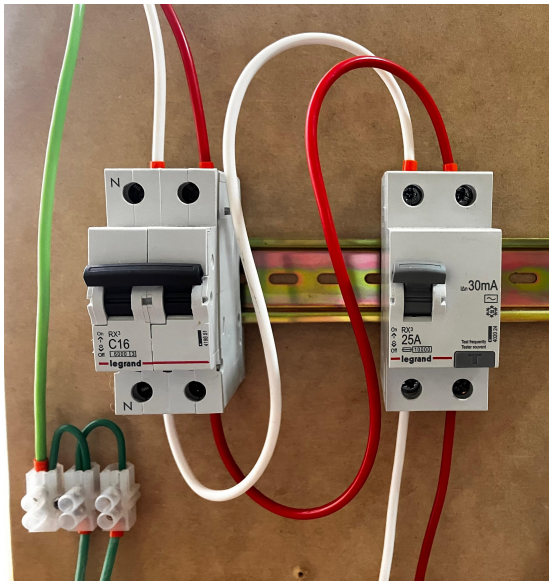


Figura 4.1: Interruptor termomagnético (16A) y diferencial (25A) montados en riel DIN.

4.1.3 Equipos de Medición

Para el monitoreo del consumo y las variables eléctricas, se dispusieron dos equipos. Como patrón de referencia absoluto para el registro de energía acumulada, se utilizó un medidor monofásico LEXO DDS608, el cual cuenta con certificación de la Superintendencia de Electricidad y Combustibles (SEC). Adicionalmente, se instaló el medidor digital multiparámetro D52-2066, de forma auxiliar para monitorear en tiempo real variables instantáneas (voltaje, corriente y potencia activa), delegando la métrica de consumo total exclusivamente al equipo certificado.



Figura 4.2: Medidor LEXO DDS608 (patrón certificado SEC) junto al medidor electrónico auxiliar.

Para la verificación de voltajes, de corriente continua, testeo de continuidad y resolución de problemas a nivel de hardware en las placas de desarrollo, se empleó un multímetro digital TRUPER-830.



Figura 4.3: Multímetro digital TRUPER-830 utilizado para las mediciones de hardware.

4.1.4 Montaje Final del Tablero

Para la energización del sistema, la alimentación de corriente alterna ingresa a través de un enchufe macho de 16 amperes mediante un cableado estructurado compuesto por tres conductores de $4\ mm^2$ (rojo para fase, blanco para neutro y verde para tierra).

Inicialmente, los conductores de fase y neutro se conectan al interruptor termomagnético, mientras que el cable de tierra se encarga de proteger de forma directa las salidas de enchufe. Desde las protecciones termomagnéticas, la fase y el neutro continúan hacia el interruptor diferencial y, posteriormente, alimentan la entrada del medidor patrón LEXO DDS608 a través de sus bornes principales (pines 1 y 2).

Desde la salida de este equipo certificado (pines 3 y 4), el circuito se conecta al medidor electrónico auxiliar D52-2066; en este punto, el conductor de neutro se distribuye directamente hacia los enchufes, en tanto que la fase atraviesa el toroide de corriente (*CT*) del medidor auxiliar para el registro de las variables instantáneas, culminando su trayecto en las salidas de enchufe hembra para la conexión de las cargas de prueba.

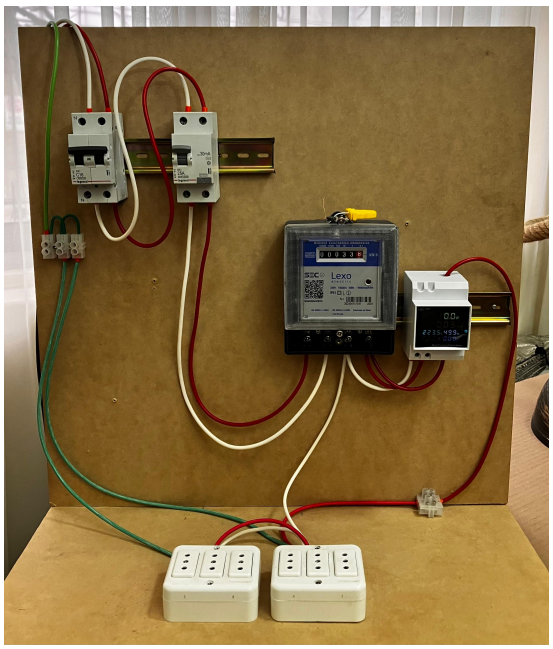


Figura 4.4: Vista general del tablero de pruebas finalizado.

La integración de todos los componentes de protección, medición y cableado dio como resultado el entorno de pruebas robusto que se aprecia en la Figura 4.4, listo para la instalación y evaluación de los prototipos. Ademas, en la Figura 4.5 se muestra el diagrama eléctrico del tablero de pruebas.

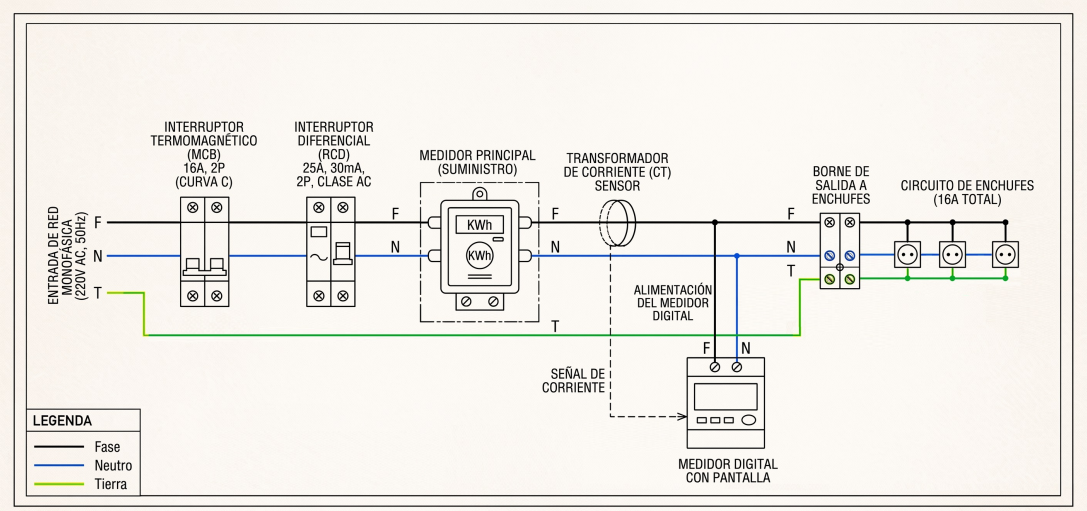


Figura 4.5: Diagrama eléctrico del tablero de pruebas finalizado.

4.2 Implementación del Prototipo Óptico (BPW34)

4.2.1 Acondicionamiento de Hardware y Circuitaje

El circuito de detección óptica se basó en el fotodiodo BPW34, cuyo terminal positivo fue conectado directamente al pin de alimentación (VCC) de la placa ESP32. El segundo terminal del sensor se conectó al pin analógico GPIO 32. Para lograr un acondicionamiento adecuado de la señal y forzar un estado lógico bajo por defecto, se implementó una configuración *pull-down* utilizando tres resistencias de $330\ k\Omega$ conectadas en serie entre el GPIO 32 y la referencia de tierra (GND). La medición física de este arreglo resistivo arrojó un valor total de $1000\ k\Omega$ ($1\ M\Omega$). Adicionalmente, el circuito incorpora dos indicadores visuales de estado: un LED azul conectado al GPIO 2 para monitorear la conexión Wi-Fi, y un LED rojo en el GPIO 19 destinado a replicar físicamente el parpadeo registrado por el medidor. En la Figura 4.6 se ilustra

el esquema de conexiones empleado, mientras que la Figura 4.7 muestra el montaje físico finalizado.

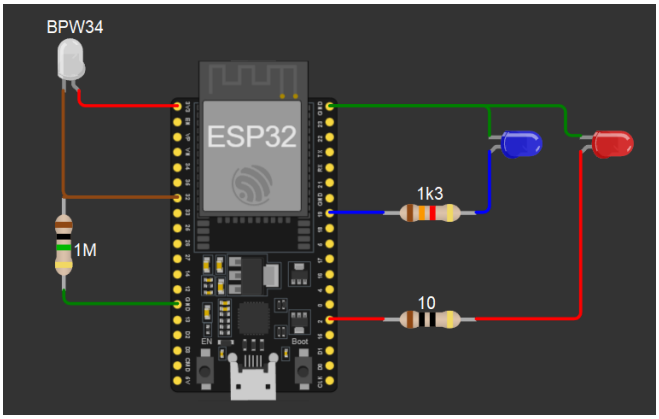


Figura 4.6: Diagrama de conexión del sensor BPW34 y ESP32.

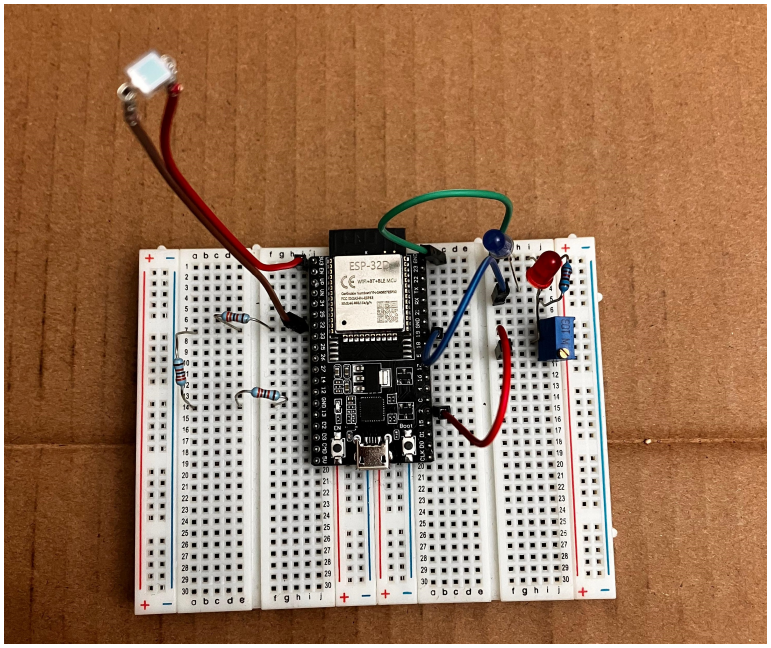


Figura 4.7: Implementación física del prototipo óptico montado.

4.2.2 Validaciones de Detección

Para comprobar la viabilidad del circuito acondicionador y la respuesta del fotodiodo BPW34 frente a estímulos lumínicos, se diseñó una etapa de pruebas controladas

dividida en dos fases.

En la primera fase, se conectó el sensor al microcontrolador y se ejecutó un *script* de monitoreo a través del puerto serial, el cual se expone en el Código 6.1. Utilizando una fuente de luz externa (linterna), se iluminó el fotodiodo hasta alcanzar su punto de saturación analógica. Tal como se observa en la Figura 4.8, el sensor respondió adecuadamente a los cambios de intensidad, validando su capacidad inicial de detección.

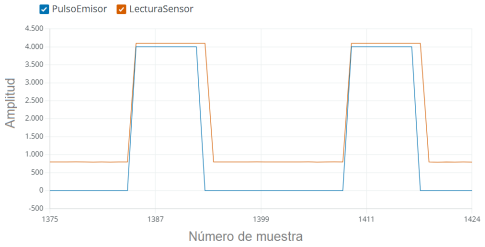


Figura 4.8: Prueba de saturación del fotodiodo mediante estimulación con linterna.

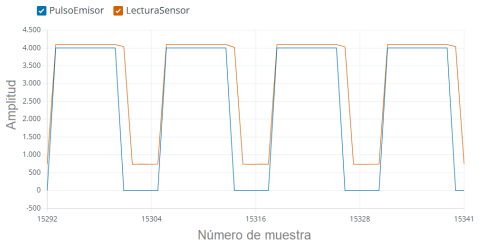
Las gráficas presentadas, obtenidas mediante el monitor serial, exponen la lectura del sensor en el eje de las ordenadas. Este eje representa el valor digitalizado de la señal (en cuentas ADC), con un rango de 0 a 4095, propio de la resolución de 12 bits del microcontrolador. Por su parte, el eje de las abscisas representa el número de muestra discreta capturada y transmitida por el puerto serial, ilustrando la evolución cronológica del experimento por cada iteración del código.

La segunda fase consistió en una prueba dinámica para evaluar la respuesta en frecuencia del sistema. Se implementó un circuito auxiliar con un LED emisor programado, en el Código 6.1, para generar destellos con una duración fija de 80 ms. En la Figura 4.9 se ilustra el comportamiento del sensor frente a un barrido de frecuencias. El sistema logró registrar de forma precisa los estímulos a 4 Hz, 8 Hz y 10 Hz. Sin embargo, al incrementar la frecuencia a 11 Hz, el fotodiodo alcanzó su límite operativo, registrando el estímulo como un pulso continuo debido a la incapacidad de

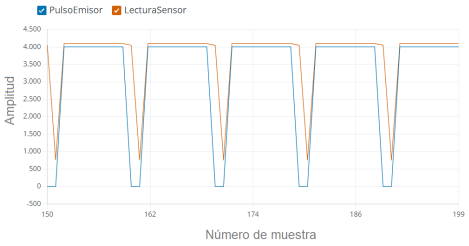
distinguir los intervalos sin luz.



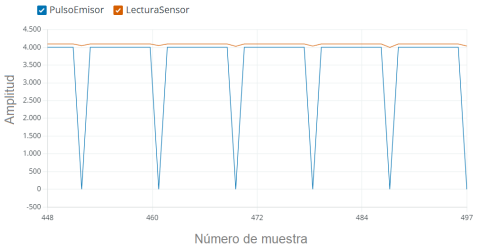
(a) Respuesta a 4 Hz



(b) Respuesta a 8 Hz



(c) Respuesta a 10 Hz



(d) Respuesta a 11 Hz (Saturación)

Figura 4.9: Comportamiento del fotodiodo ante distintas frecuencias de estimulación.

Para corroborar la fiabilidad del algoritmo de conteo, se realizó una medición específica a 8 Hz. La Figura 4.10 demuestra la sincronía exacta entre el pulso emisor (estado lógico) y el incremento unitario en el contador de pulsos, comprobando que la lógica *anti-rebote* programada funciona correctamente y no se generan falsos positivos por un mismo destello.

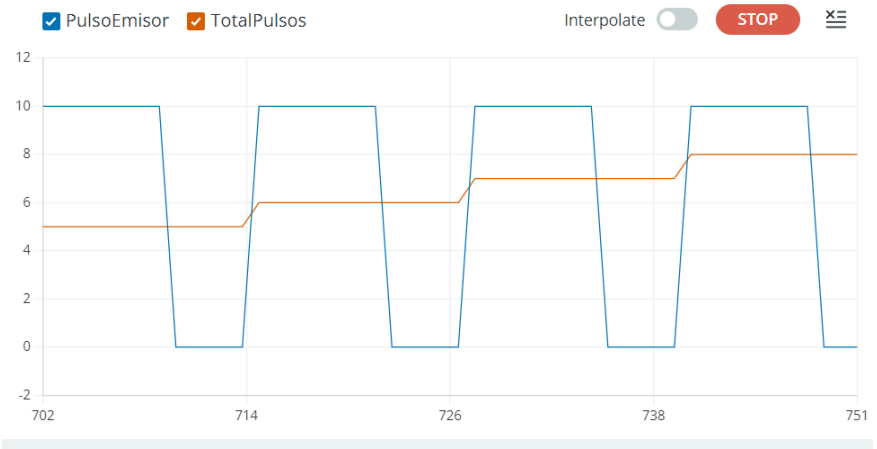


Figura 4.10: Validación del contador de pulsos versus pulso emisor a 8 Hz.

4.2.3 Instalación sobre Medidor Eléctrico

Tras validar la capacidad de detección y los límites operacionales en un entorno controlado, se procedió a la instalación física del prototipo sobre el medidor eléctrico certificado (LEXO DDS608).

El factor crítico en esta etapa fue garantizar un aislamiento óptico absoluto, dado que cualquier filtración de luz ambiental superaría el umbral de detección definido y alteraría el conteo. Para solucionar esto, se fabricó un encapsulado artesanal utilizando una tapa de botella plástica. El fotodiodo fue insertado en esta carcasa y sellado herméticamente con silicona en su parte posterior.



Figura 4.11: Encapsulado para garantizar un aislamiento óptico absoluto.

Finalmente, como se aprecia en la Figura 4.12, el conjunto fue posicionado directamente sobre el LED indicador de impulsos del medidor y adherido con huincha aisladora. Este montaje aseguró la alineación perfecta con la fuente de luz y bloqueó de forma efectiva cualquier interferencia externa.



Figura 4.12: Instalación física y acople óptico del fotodiodo sobre el medidor eléctrico.

4.2.4 Lógica del Firmware

A nivel de *software*, tras la configuración inicial de los periféricos y el establecimiento de la resolución analógica en 12 bits (véase el Código 6.2), el sistema realiza lecturas continuas del pin analógico. La detección de un pulso luminoso ocurre cuando la lectura del fotodiodo supera un umbral empírico de 4000. Para evitar conteos múltiples provenientes de un mismo destello, se implementó una lógica de control de estado mediante una variable booleana (`pulsoDetectado`) que restringe el incremento del contador a un único evento por cada ciclo de luz, tal como se expone en el Código 6.3. Posteriormente, los pulsos registrados son convertidos a energía (kWh) utilizando la constante de calibración de 1600 pulsos/kWh.

Para dotar al sistema de tolerancia frente a caídas de red, el *firmware* incluye una estructura de *buffer* (`RegistroOffline`) con una capacidad máxima de 24 registros. El sistema consolida y almacena la información de consumo en intervalos fijos de

15 minutos, lógica cuya implementación se detalla en el Código 6.4. Una vez que se establece o recupera la conexión Wi-Fi, la placa sincroniza su reloj interno mediante el protocolo NTP (`pool.ntp.org`) y procede a transmitir los registros almacenados hacia la plataforma ThingSpeak mediante peticiones HTTP GET. Esta transmisión de datos atrasados se ejecuta de forma dosificada, realizando un envío cada 16 segundos para cumplir con las restricciones de la API externa, proceso que se evidencia en el Código 6.5.

4.3 Implementación del Prototipo de Visión Artificial (ESP32-CAM)

4.3.1 Instalación Física, Enfoque y Control del Flash LED

El módulo de visión seleccionado para la digitalización de la lectura fue la placa ESP32-CAM (modelo AI-Thinker). Para llevar a cabo la carga del *firmware* y la alimentación eléctrica inicial del microcontrolador de forma estable mediante un puerto MicroUSB estándar, se empleó una placa adaptadora o base de programación para ESP32-CAM, componente fundamental que facilitó la interfaz de comunicación serial sin requerir conexiones cableadas complejas sobre los pines GPIO. En la Figura 4.13 se muestra la ESP32-CAM montada sobre su base adaptadora conectada por MicroUSB.

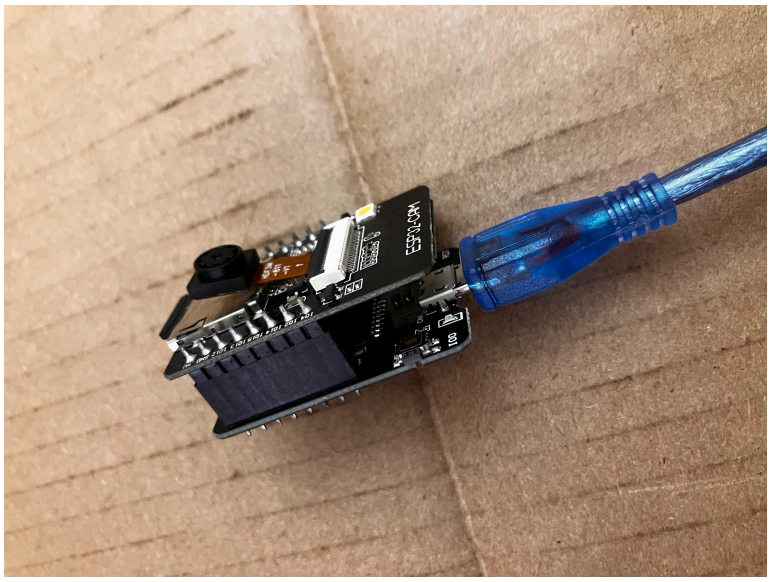


Figura 4.13: ESP32-CAM montada sobre su base adaptadora.

Dado que este módulo cuenta con un lente enfocado al infinito desde fábrica, se determinó empíricamente una distancia focal óptima de aproximadamente 25 cm entre el lente de la cámara y la pantalla del medidor eléctrico.

Para mantener esta distancia de forma estable y permitir ajustes en la alineación, se empleó un brazo articulado flexible diseñado originalmente para la sujeción de teléfonos móviles. Este soporte, compuesto por un núcleo de alambre plastificado y pinzas de presión en ambos extremos, permitió fijar firmemente el módulo de visión frente al equipo de medida, garantizando la repetibilidad geométrica de las capturas fotográficas. El resultado de este montaje se puede apreciar en la Figura 4.14.

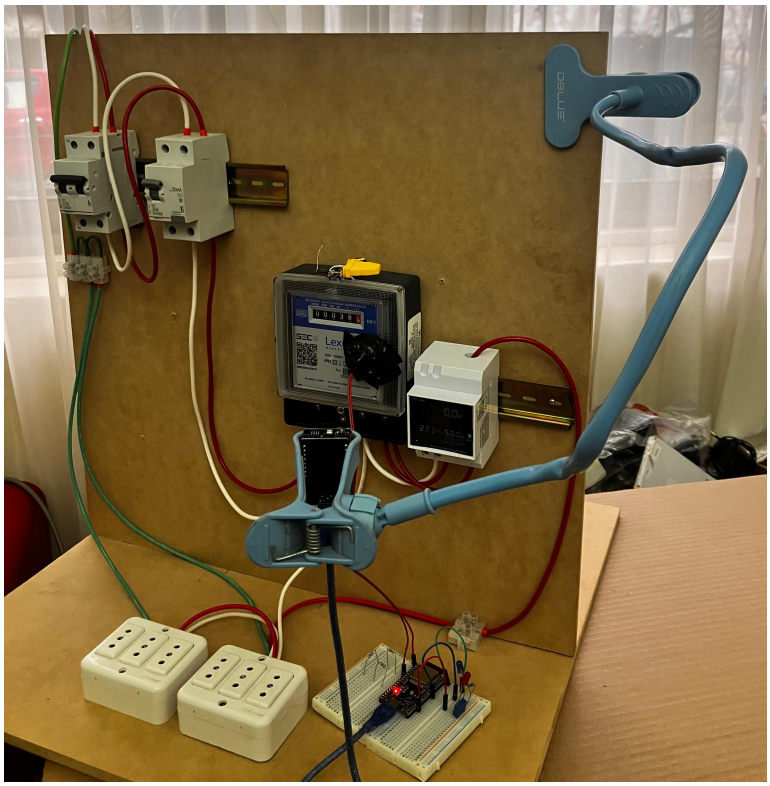


Figura 4.14: Montaje de la ESP32-CAM frente al medidor utilizando brazo articulado a 25 cm de distancia.

Para asegurar una captura nítida en distintas condiciones lumínicas, se hizo uso del Flash LED integrado en la placa (GPIO 4). Mediante el *firmware* (véase el Código 6.6), el sistema enciende el Flash una fracción de segundo antes de tomar la fotografía para permitir que el sensor ajuste la exposición, y lo apaga inmediatamente después de almacenar la imagen para evitar el sobrecalentamiento del componente. Un desafío importante durante la instalación física fue el deslumbramiento; dado que la cubierta del medidor tiende a reflejar la luz, una captura completamente frontal generaba un brillo intenso que ocultaba los dígitos y anulaba el procesamiento OCR. Para solucionar esto, la cámara no se instaló directamente frente a la pantalla, sino que se posicionó con un ligero ángulo en diagonal. Esta inclinación permitió que el rebote directo de la luz del flash se desviara fuera del campo de visión del lente, garantizando una imagen limpia, contrastada y libre de reflejos parásitos.

4.3.2 Algoritmo de Preprocesamiento de Imagen y Tesseract OCR

Para la extracción de los dígitos del medidor se optó por un procesamiento completamente local utilizando el motor Tesseract OCR, evitando la dependencia de servicios en la nube para reducir latencias operativas y garantizar el funcionamiento *offline*.

El proceso inicia con una fase de calibración mediante la interfaz gráfica de OpenCV, donde el operador define manualmente el área de interés (ROI, *Region of Interest*) que engloba los seis dígitos del medidor (véase el Código 6.9). Esta selección, visualizada en la Figura 4.15, se segmenta dinámicamente en tres recortes específicos para facilitar la lectura individual: tres dígitos para los enteros mayores, dos dígitos intermedios y un dígito decimal.



Figura 4.15: Interfaz de selección de la Región de Interés (ROI) sobre los dígitos del medidor.

Para maximizar la tasa de éxito en el reconocimiento de caracteres, se implementó un algoritmo iterativo de barrido de umbrales. La rutina aplica una binarización de imagen combinada con una inversión de colores, iterando un rango de umbrales en pasos de 5 unidades. Un ejemplo de la binarización resultante entregada al OCR se muestra en la Figura 4.16.

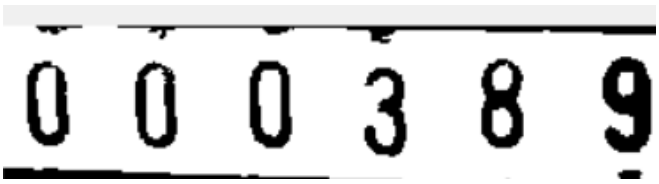


Figura 4.16: Procesamiento de imagen: binarización e inversión de colores previa a la lectura OCR.

4.3.3 Lógica de Validación y Manejo de Errores (Modo Ráfaga)

Para garantizar la alta fiabilidad de los datos extraídos mediante Visión Artificial antes de su transmisión a la plataforma ThingSpeak, el *software* implementa un sistema de validación secuencial estructurado en tres niveles: estadístico, histórico y de contingencia ante transiciones mecánicas.

El primer filtro es el nivel de validación estadística, el cual actúa directamente sobre los resultados del motor OCR. Dado que la iluminación y el contraste pueden variar, el sistema no confía en una única captura estática. En su lugar, realiza un barrido con múltiples umbrales de binarización. Una vez finalizado este ciclo de procesamiento, se aplica un filtro estadístico riguroso: el algoritmo exige un mínimo de 4 lecturas válidas (que cumplan con la cantidad de dígitos esperada) durante el barrido. Utilizando la clase `Counter` de la librería estándar `collections` de Python, se agrupan los resultados y se determina el valor más frecuente. Si la coincidencia de este valor supera un umbral de confianza del 65 %, el dígito se da por válido y avanza a la siguiente etapa; de lo contrario, se descarta por falta de certeza (véase el Código 6.8).

Superada la etapa estadística, la lectura ensamblada se somete a un nivel de validación histórica (véase el Código 6.10). Cada nueva medición se compara con el máximo histórico almacenado de forma persistente y local en un archivo JSON. Debido a que el registro de energía eléctrica es de naturaleza estrictamente acumulativa, cualquier medición que presente un valor inferior al último registro exitoso es automáticamente descartada. Asimismo, el algoritmo calcula de forma dinámica un "salto máximo"permitido en función del tiempo transcurrido desde la última lectura válida. Si la diferencia excede este límite físico o lógico, se asume que hubo un error de interpretación óptica y el registro se bloquea, protegiendo así la integridad de la base de datos y evitando picos en el monitoreo remoto.

Finalmente, el sistema cuenta con un nivel de contingencia operativa denominado "Modo Ráfaga"(véase el Código 6.11), diseñado específicamente para resolver la

principal limitación del medidor LEXO: el movimiento de sus rodillos mecánicos. Durante la rotación del tambor, los números quedan en posiciones intermedias, lo que provoca que el OCR falle inevitablemente. Si el algoritmo registra tres intentos fallidos de manera consecutiva, evalúa mediante OpenCV si la imagen actual presenta una variación estructural (superior al 3 % de los píxeles) respecto a la última captura. Esta diferencia confirma que el medidor no está estático, sino en plena transición mecánica. Al detectar esta condición, el sistema modifica su comportamiento y reduce su intervalo de captura normal de 900 segundos a solo 60 segundos, durante un máximo de 12 intentos. Esta aceleración temporal permite al *software* hacer un seguimiento continuo y cazar el momento exacto en que el rodillo finaliza su recorrido y se estabiliza, para luego registrar el nuevo valor y retornar automáticamente a su estado de bajo consumo y monitoreo espaciado.

4.3.4 Transmisión de la Imagen al Servidor Local

La arquitectura de red del sistema se diseñó bajo un modelo cliente-servidor en una red WiFi local. La ESP32-CAM actúa como un servidor HTTP, inicializando su servicio mediante la API `esp_http_server` y habilitando el *endpoint* `/capture` para atender peticiones de tipo GET. Al recibir una petición, la placa procesa la fotografía, enciende el flash, empaqueta la imagen en formato JPEG (configurado en calidad 10 y resolución UXGA) y la transmite al cliente.

Del lado del servidor de procesamiento, un *script* desarrollado en Python se encarga de solicitar la imagen utilizando la librería `requests`. Para dotar al sistema de resiliencia frente a intermitencias en la red, el código incorpora una función de descarga con hasta 3 intentos de captura, donde el tiempo de espera se incrementa de forma lineal si el módulo no responde, asegurando la decodificación exitosa de la imagen a un arreglo matricial de `numpy`.

4.4 Comunicación con la Nube y Visualización

4.4.1 Configuración de Canales y Campos

Una vez que el algoritmo de Visión Artificial (OCR local) o el prototipo óptico validan exitosamente una lectura de consumo, la información debe ser transmitida a la plataforma en la nube ThingSpeak para su almacenamiento y procesamiento analítico. El sistema utiliza la conexión Wi-Fi del microcontrolador ESP32 para realizar una petición HTTP (POST) a la API REST de la plataforma, enviando el dato consolidado.

La estructura del canal en la nube se configuró para recibir la métrica principal de energía en su campo primario (`Field 1`). Este campo almacena el consumo eléctrico de forma secuencial en kilowatts-hora (kWh), permitiendo a la plataforma contar con una base de datos histórica, estructurada y disponible en tiempo real para la ejecución de *scripts* de análisis y visualización remota.

4.4.2 Métricas e Indicadores de Actividad

Para transformar los datos energéticos crudos en información útil para el monitoreo remoto de rutinas (orientado al cuidado de adultos mayores), se desarrollaron diversos *scripts* de análisis utilizando el motor de MATLAB integrado en ThingSpeak. Estos códigos procesan la serie temporal almacenada en el `Field 1` y generan seis visualizaciones clave que se actualizan dinámicamente en el panel principal del sistema:

1. **Consumo Eléctrico Últimas 24 Horas (`Field 1 Chart`):** Es la visualización directa de los datos crudos entrantes. Muestra en un gráfico de barras rojas de alta frecuencia el consumo registrado (en kWh) a lo largo del día, permitiendo verificar de un vistazo la correcta recepción y continuidad de los paquetes enviados por el microcontrolador.

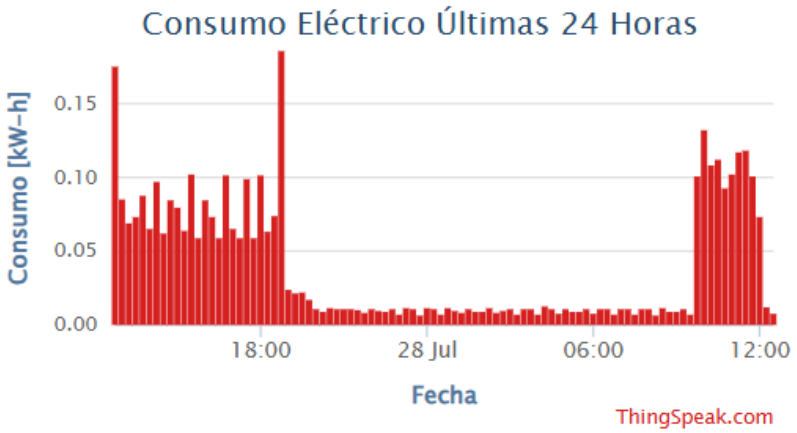


Figura 4.17: Visualización en tiempo real del consumo eléctrico de las últimas 24 horas.

2. **Consumo Basal:** Este gráfico agrupa el consumo por hora y superpone una línea de tendencia crítica: el consumo basal. Matemáticamente, este valor se calcula extrayendo la mediana de los datos de consumo registrados exclusivamente en el bloque horario de 02:00 a 06:00 de la madrugada. Esto permite aislar el consumo automático del hogar (como el ciclo de refrigeradores) del consumo atribuible a la acción humana directa.

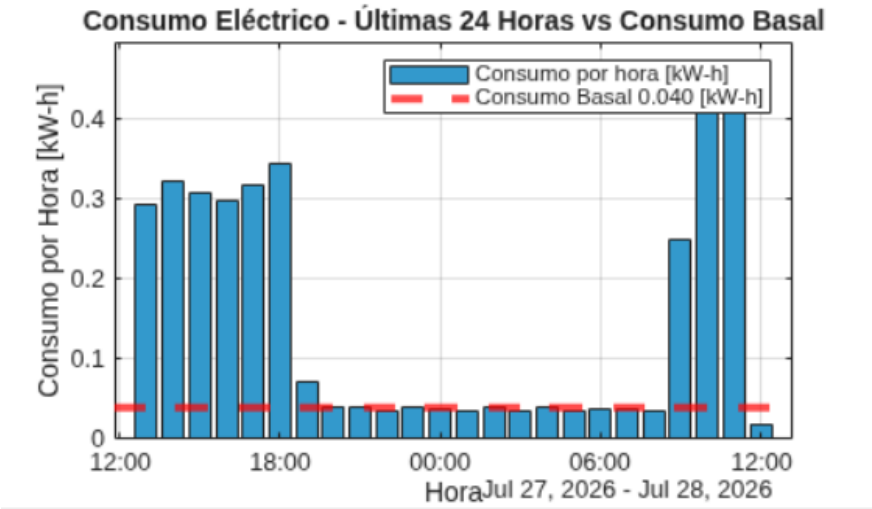


Figura 4.18: Consumo por hora versus consumo basal del hogar.

3. **Monitoreo de Actividad en el Hogar:** Basado en el cálculo del gráfico anterior,

este indicador transforma la energía en estado de actividad. Si el consumo en un intervalo determinado supera el umbral basal establecido, el sistema cambia a estado Activo; si decae al nivel basal, retorna a Inactivo. Este gráfico de escalón es el núcleo del monitoreo de bienestar, ya que evidencia de forma binaria si hay movimiento e interacción con electrodomésticos en la vivienda.

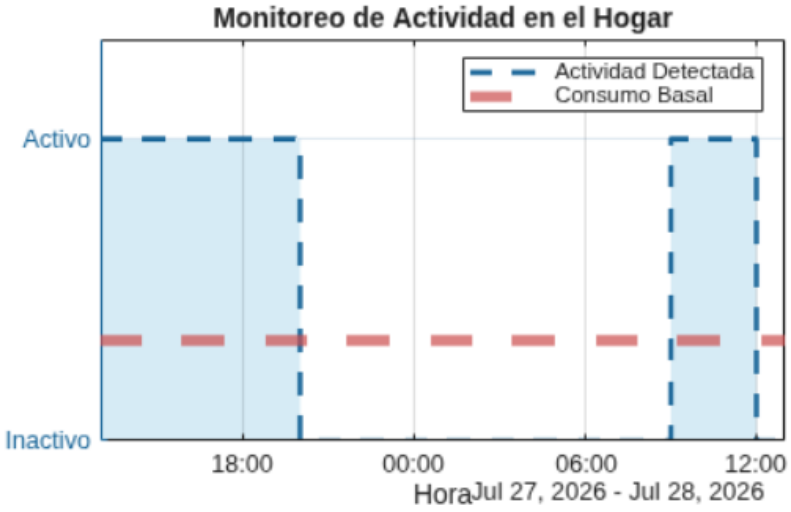


Figura 4.19: Indicador binario de actividad en el hogar basado en el consumo basal.

4. **Heatmap de Consumo por Hora (Últimos 7 Días):** Consiste en una matriz térmica donde el eje vertical representa los días de la semana y el horizontal las horas del día. La intensidad del color azul mapea el nivel de consumo en kWh. Esta herramienta analítica es fundamental para identificar patrones de comportamiento a mediano plazo y detectar anomalías en la rutina semanal del usuario.

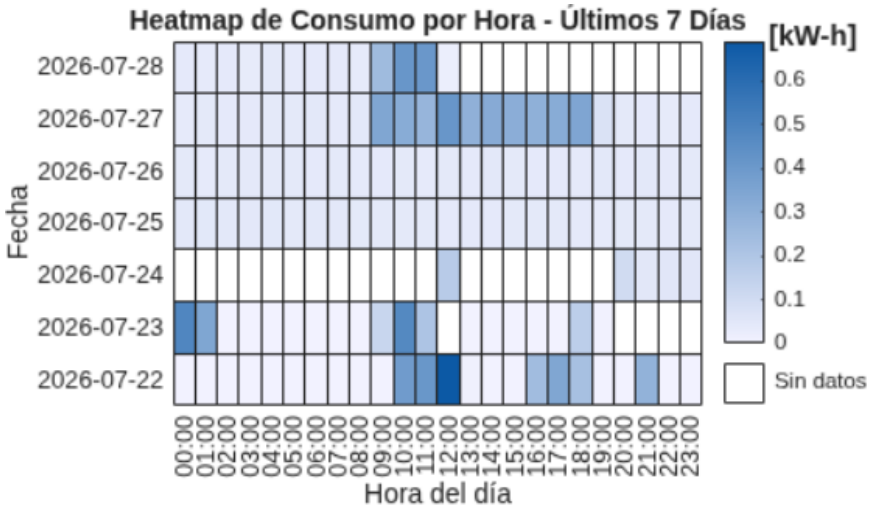


Figura 4.20: Mapa de calor (Heatmap) que ilustra los patrones de consumo de los últimos 7 días.

5. **Análisis Histórico a Corto Plazo (Últimas 48 Horas):** Panel de gráficos de barras apiladas verticalmente que compara el perfil de consumo por hora del día actual (Hoy) respecto al día inmediatamente anterior (Ayer). Al mantener los ejes *Y* normalizados, se facilita la comparación visual rápida.

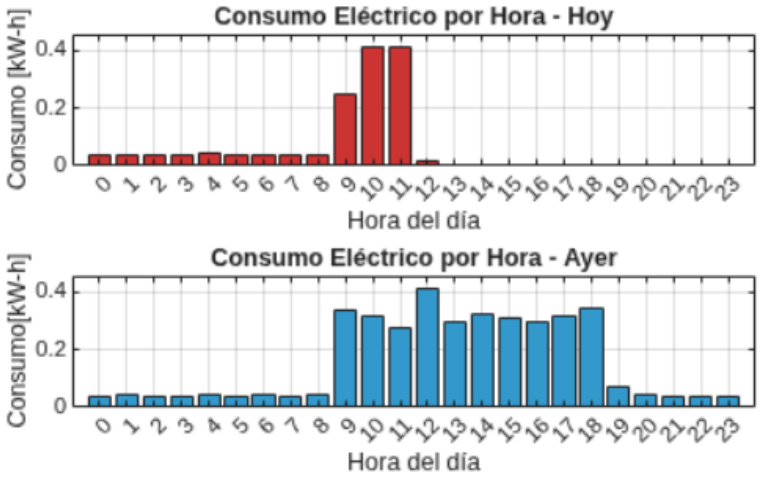


Figura 4.21: Comparativa del perfil de consumo por hora de las últimas 48 horas.

6. **Análisis Histórico Extendido (Últimas 72 Horas):** Expansión del análisis anterior que incluye un tercer día (Antier) para una visión más amplia de la ten-

dencia reciente del hogar, permitiendo determinar si el comportamiento del día en curso se alinea con la rutina habitual.

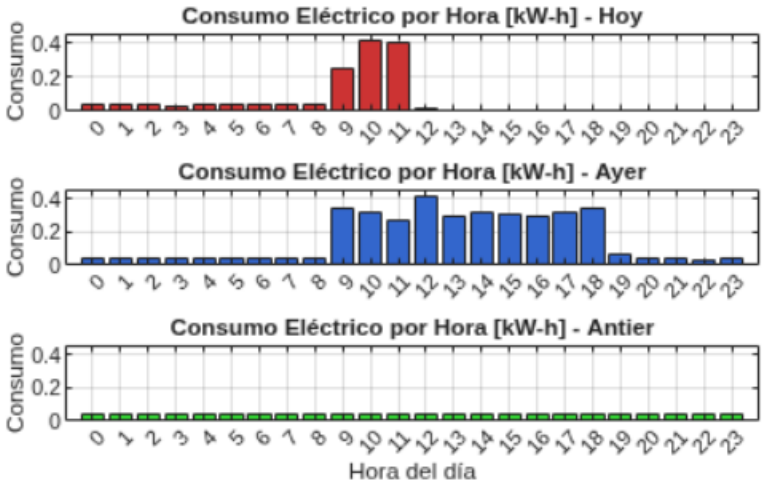


Figura 4.22: Comparativa extendida del perfil de consumo por hora de las últimas 72 horas.

4.4.3 Sistema de Alerta Integrado y Control Temporal

Para complementar la monitorización pasiva a través de los paneles visuales en ThingSpeak, se implementó un sistema de notificaciones proactivo utilizando la herramienta integrada *MATLAB Analysis* y la API de *ThingSpeak Alerts*. El propósito fundamental de este módulo es detectar automáticamente la ausencia prolongada de rutinas y enviar un correo electrónico de advertencia a los cuidadores, operando como un mecanismo de teleasistencia de respuesta rápida.

El algoritmo (véase el Código 6.12), se ejecuta de forma programada en la nube y basa su funcionamiento en cuatro etapas lógicas sucesivas:

1. **Extracción de Datos Temporales:** El script recupera los registros del canal correspondientes a dos ventanas de tiempo: los últimos 7 días (para establecer una base estadística) y las últimas 25 horas (para evaluar el comportamiento reciente).

2. **Cálculo Dinámico del Consumo Basal:** Para evitar falsas alarmas provocadas por el ciclo automático de electrodomésticos (como refrigeradores), el sistema aísla exclusivamente los datos registrados en la madrugada (entre las 02:00 y las 06:00 horas). Posteriormente, agrupa este consumo por hora y calcula la mediana estadística (`median`), obteniendo un valor basal depurado y representativo de la vivienda en estado de reposo absoluto.

3. **Detección de Inactividad Reciente:** Una vez definido el basal, el sistema establece un umbral de actividad empírico, fijado en un 30 % por sobre el nivel de reposo ($consumo_basal \times 1{,}3$). El algoritmo segmenta las últimas 25 horas en bloques horarios exactos y busca iterativamente el último bloque que haya superado dicho umbral. Al restar la marca de tiempo de esa última actividad con la hora actual, el script cuantifica con precisión la cantidad de horas continuas de inactividad del usuario.

4. **Control Temporal y Disparo de Alertas:** Para prevenir la saturación de correos electrónicos (*spam*) en la bandeja de entrada del cuidador ante un evento prolongado, se implementó una estricta lógica de control temporal. Las sentencias condicionales operan bajo ventanas de activación cerradas. Por ejemplo, la alerta de 12 horas solo se dispara si la inactividad es mayor o igual a 12, pero estrictamente menor a 13 horas. Este diseño garantiza que, durante ese lapso de inactividad, la petición HTTP POST hacia la URL de *ThingSpeak Alerts* (`webwrite`) se ejecute una única vez por cada nivel de criticidad (12 horas y 24 horas).

Capítulo 5

Resultados

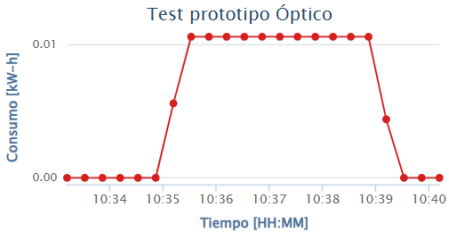
En esta sección se presentan los resultados obtenidos al someter los dos sistemas desarrollados (el prototipo óptico y el prototipo de visión artificial) a una serie de pruebas estandarizadas. El objetivo es contrastar el rendimiento, la fiabilidad y el consumo de recursos de ambos enfoques bajo condiciones operativas idénticas y escenarios de estrés.

5.1 Análisis de Frecuencia de Muestreo

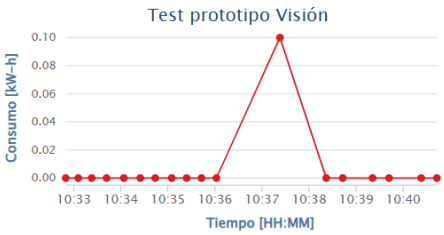
En esta prueba se evaluó el comportamiento de captura y transmisión de ambos dispositivos (prototipo óptico y prototipo de visión) frente a cargas eléctricas controladas. Para esto, se utilizaron dos niveles de potencia estandarizados: 2000 W y 3000 W. Durante el ensayo, se configuraron tres intervalos de muestreo distintos (20, 30 y 60 segundos) para cada nivel de carga, con el objetivo de registrar el desempeño de ambos sistemas operando en paralelo bajo las mismas condiciones.

5.1.1 Frecuencia de muestreo: 20 segundos

Carga de 2000 W



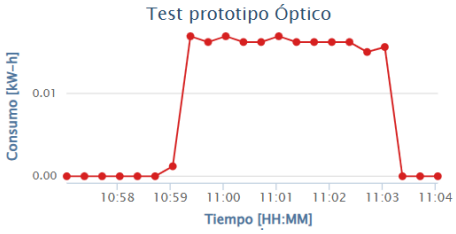
(a) Prototipo óptico.



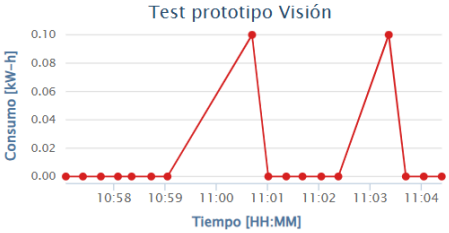
(b) Prototipo de visión.

Figura 5.1: Registro comparativo a 20s de muestreo y carga de 2000W.

Carga de 3000 W



(a) Prototipo óptico.

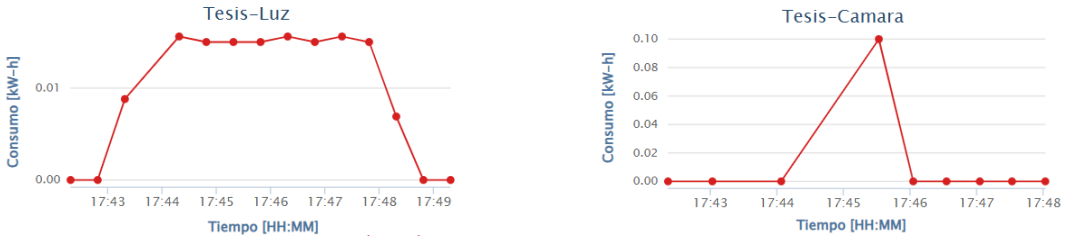


(b) Prototipo de visión.

Figura 5.2: Registro comparativo a 20s de muestreo y carga de 3000W.

5.1.2 Frecuencia de muestreo: 30 segundos

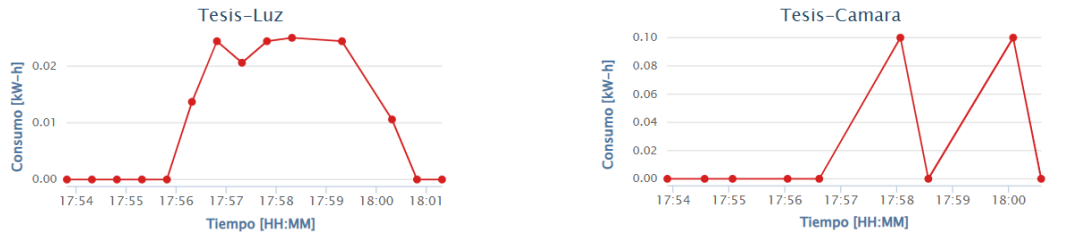
Carga de 2000 W



(a) Prototipo óptico. (b) Prototipo de visión.

Figura 5.3: Registro comparativo a 30s de muestreo y carga de 2000W.

Carga de 3000 W

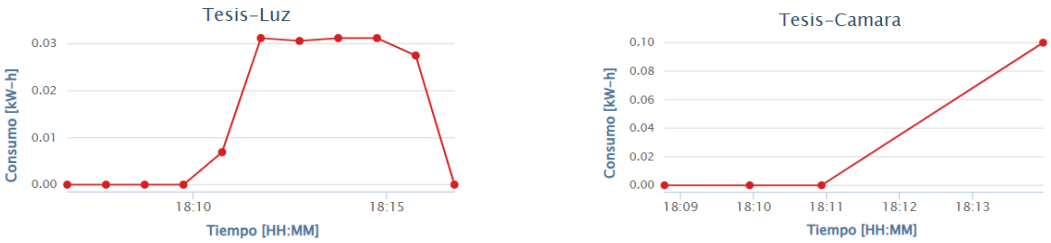


(a) Prototipo óptico. (b) Prototipo de visión.

Figura 5.4: Registro comparativo a 30s de muestreo y carga de 3000W.

5.1.3 Frecuencia de muestreo: 60 segundos

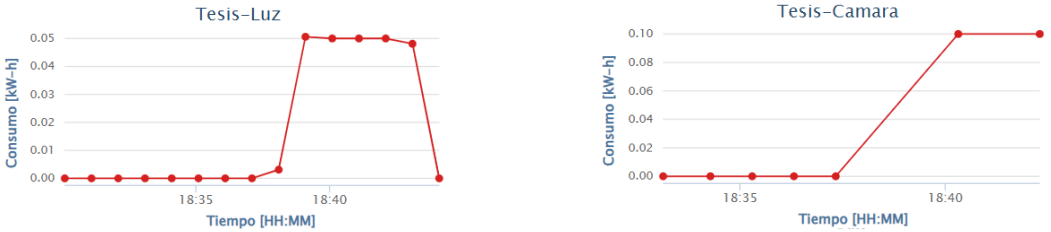
Carga de 2000 W



(a) Prototipo óptico. (b) Prototipo de visión.

Figura 5.5: Registro comparativo a 60s de muestreo y carga de 2000W.

Carga de 3000 W



(a) Prototipo óptico. (b) Prototipo de visión.

Figura 5.6: Registro comparativo a 60s de muestreo y carga de 3000W.

A partir del análisis de las pruebas realizadas, se identifican tres diferencias operativas fundamentales entre ambas arquitecturas de medición, las cuales se evidencian claramente al observar el comportamiento gráfico de los datos.

En primer lugar, existe una diferencia crítica en la resolución de las mediciones. El prototipo óptico presenta una sensibilidad notablemente mayor, lo que le permite generar perfiles de consumo continuos y estables. Esto se aprecia con claridad en la Figura 5.1, donde el sensor óptico dibuja un escalón casi perfecto y sostenido con incrementos de 0.0001 kWh. Por el contrario, en esa misma figura, el prototipo de visión se ve limitado por la escala física del odómetro, entregando variaciones mucho más toscas y discretas con saltos abruptos de 0.1 kWh.

En segundo lugar, el procesamiento local de caracteres del prototipo de visión presenta vulnerabilidades durante las transiciones mecánicas, lo que se traduce en pérdida de información. Al observar las pruebas con carga de 3000 W a altas frecuencias, específicamente en las Figuras 5.2 y 5.4, se aprecian grandes vacíos y caídas drásticas a cero en pleno periodo de actividad del hervidor. Este fenómeno ocurre porque los rodillos numéricos se encuentran en movimiento (entre dos números), generando una imagen ambigua que el motor OCR descarta por baja fiabilidad. Al no validarse el dato, se omite el envío a ThingSpeak, generando los quiebres observados en la gráfica de visión, un problema del cual el prototipo óptico, con su transmisión ininterrumpida, está exento.

Finalmente, el tiempo de respuesta frente a la conexión de la carga varía drásticamente. El sensor óptico exhibe una reacción prácticamente instantánea al captar el aumento en la frecuencia de parpadeo del LED metrológico. Esto queda en evidencia, por ejemplo, en la Figura 5.6 (muestreo a 60 segundos), donde el prototipo óptico comienza a registrar el alza de manera progresiva inmediatamente tras encender el hervidor. En contraste, el sistema de visión presenta una latencia mecánica importante; en la misma gráfica se observa cómo la lectura se mantiene plana en cero durante varios intervalos antes de que el rodillo gire lo suficiente para que la cámara valide el nuevo dígito.

5.2 Consumo Energético de los Prototipos

Un factor crítico en el despliegue de nodos IoT es su autonomía y eficiencia energética. Para evaluar rigurosamente este parámetro, se diseñó un esquema de medición físico que permitió registrar la corriente eléctrica consumida por cada microcontrolador operando a su voltaje nominal.

Para llevar a cabo las mediciones empíricas, se adaptó un cable de alimentación estándar USB tipo A a Micro-USB. La metodología consistió en interrumpir la línea de alimentación de voltaje, derivando sus terminales hacia una placa de pruebas (*pro-*

toboard). Mediante esta configuración, que se ilustra en la Figura 5.7, se conectó un multímetro digital en serie con el circuito, lo que hizo posible obtener lecturas directas de la corriente consumida por los módulos.

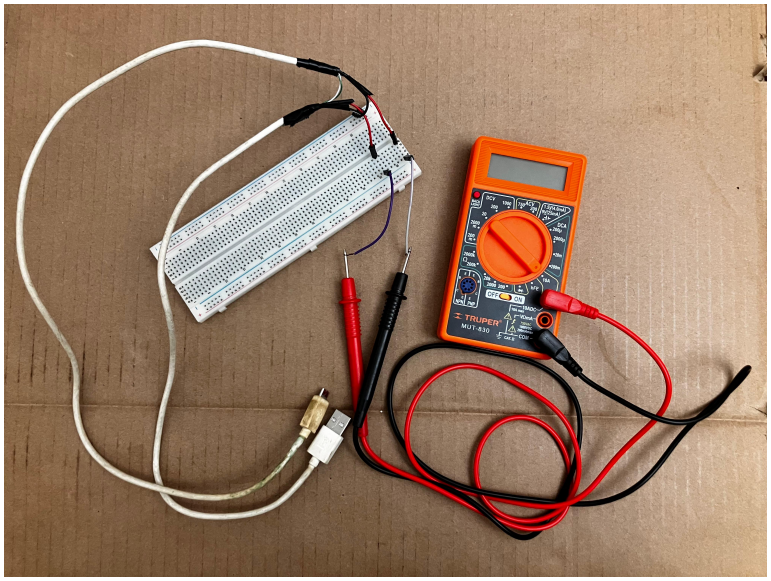
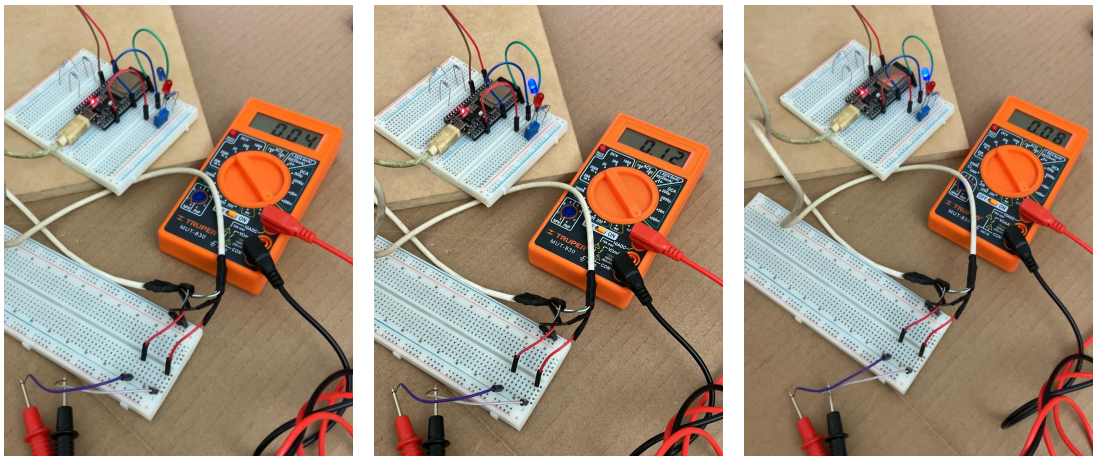


Figura 5.7: Esquema del banco de pruebas: multímetro conectado en serie en la *protoboard* mediante la interrupción del cable USB.

Bajo esta metodología, se registraron tres estados operativos fundamentales para cada prototipo: el estado de reposo (monitoreo pasivo), la fase de conexión a la red (búsqueda de Wi-Fi) y el estado de carga máxima o *peak* (procesamiento y transmisión).

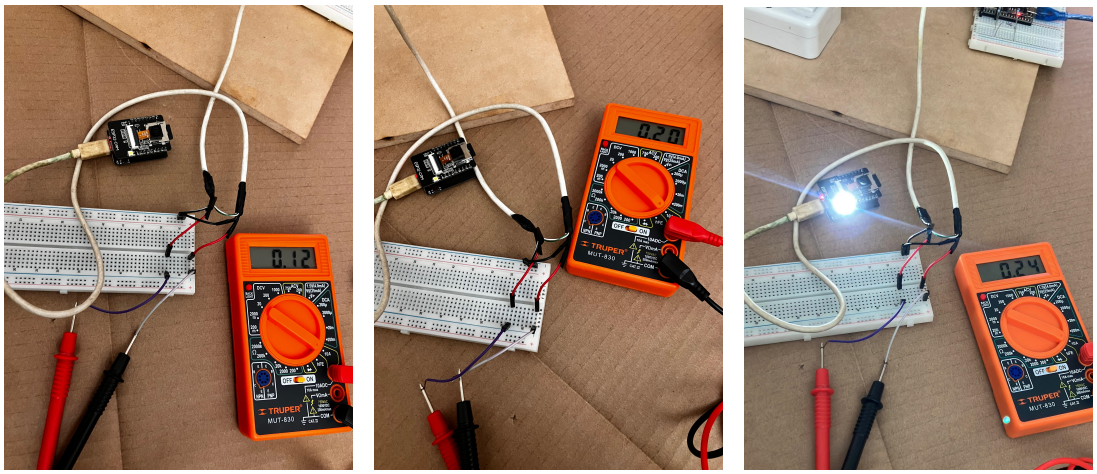
En el caso del prototipo óptico, cuyos resultados se detallan en la Figura 5.8, el sistema presentó un consumo basal en estado de reposo de **0.04 A**, asociado a la lectura continua del sensor BWP34. Durante la fase inicial de búsqueda y conexión a la red Wi-Fi, el consumo se elevó a **0.12 A**. Finalmente, al detectar un pulso luminoso e iniciar la transmisión del dato hacia la plataforma ThingSpeak, el *peak* de corriente alcanzó los **0.08 A**.



(a) Reposo. (b) Buscando Wi-Fi. (c) Peak de uso.

Figura 5.8: Registro del consumo de corriente (mA) para los tres estados operativos del prototipo óptico.

Por su parte, el **prototipo de visión** (basado en la placa ESP32-CAM) registró un consumo en estado de inactividad de **0.12 A**. Durante su respectiva rutina de conexión a la red inalámbrica, el módulo demandó **0.20 A**. El cambio más drástico ocurrió en su fase crítica de operación, la cual involucra la activación del periférico de la cámara, el encendido del Flash LED y la posterior transmisión de la imagen donde el *peak* de corriente se elevó a un máximo de **0.24 A**. Esta progresión de consumo se consolida en la Figura 5.9, evidenciando la exigencia energética superior de la adquisición y procesamiento fotográfico.



(a) Reposo. (b) Buscando Wi-Fi. (c) Peak de uso

Figura 5.9: Registro del consumo de corriente (mA) para los tres estados operativos del prototipo de visión.

5.3 Pruebas de Estrés Lumínico

Dado que los medidores suelen ubicarse en exteriores o en nichos con iluminación variable, se evaluó la robustez de ambos sistemas ante distintas condiciones ambientales. Se capturaron datos bajo tres escenarios: total oscuridad, iluminación natural directa e iluminación artificial (simulada con linterna de alta intensidad).

Los resultados evidenciaron un desempeño contrastante entre las tecnologías. Mientras que el sensor óptico reaccionó de manera estable frente a los cambios lumínicos, el único método que presentó problemas fue el prototipo de visión. Al someter el equipo a la prueba de iluminación artificial intensa, el reflejo directo de la linterna sobre el cristal del medidor generó zonas de alta saturación. En ciertos ángulos, este resplandor impidió que los algoritmos de umbralización limpiaran adecuadamente la imagen antes de la extracción de texto, lo que imposibilitó al motor de reconocimiento Tesseract, aislar y realizar una lectura apropiada de los dígitos. Este efecto de degradación visual y obstrucción por destello se puede apreciar claramente en la Figura 5.10.



Figura 5.10: Reflejo producido sobre el medidor.

5.4 Sincronización y Desfase

Para validar la precisión del registro acumulado, se realizó una prueba continua de 48 horas contrastando los datos enviados a la plataforma por ambos sistemas. Durante este periodo, el prototipo óptico, registró un consumo total acumulado de 5.8116 kWh. Por su parte, el prototipo de visión cuantificó un total de 5.8000 kWh.

Esta comparación arrojó una diferencia absoluta de apenas 0.0116 kWh, lo que se traduce en un error porcentual relativo del 0.20 %. Este margen de error resulta estadísticamente insignificante para efectos de facturación o monitoreo residencial. El ligero desfase se atribuye principalmente a la latencia inherente del procesamiento de imágenes y a los breves periodos en los que el medidor de visión se ve obligado a descartar capturas por falta de claridad. Esto retrasa fraccionalmente la actualización de la energía consumida en la plataforma en comparación con la inmediatez del pulso óptico. Sin embargo, a nivel macro, ambos sistemas logran una sincronización exitosa y totalizan prácticamente la misma energía.

5.5 Tasa de Éxito de Captura

Para determinar la viabilidad operativa y la estabilidad de los algoritmos bajo un régimen de muestreo continuo, se analizó el desempeño de ambos prototipos. La métrica

principal evaluada fue el porcentaje de lecturas exitosas subidas a la nube respecto al total de intervalos teóricos exigidos por el tiempo de prueba.

El prototipo óptico demostró una fiabilidad absoluta, logrando registrar de manera exitosa 233 capturas de 233 esperadas, lo que equivale a una tasa de éxito del 100 %. Su mecanismo de detección infrarroja de pulsos no presenta vulnerabilidades frente a los estados intermedios del medidor mecánico.

Por el contrario, el prototipo de visión alcanzó una tasa de éxito del 75.97 %, concretando 177 lecturas válidas de las 233 esperadas. Al analizar los registros de procesamiento del microcontrolador, se determinó que el 24.03 % de las capturas perdidas (56 intervalos) no se deben a fallas de conectividad o de hardware, sino a limitaciones algorítmicas durante las transiciones mecánicas de los rodillos del medidor. Cuando los dígitos se encuentran a mitad de giro, el motor Tesseract OCR local arroja niveles de confianza de extracción que oscilan entre el 25 % y el 60 %, cayendo por debajo del umbral de aceptación predefinido del 65 %.

Además, la validación del sistema descarta automáticamente aquellas lecturas que, por un error de interpretación visual del OCR, resultan ser numéricamente inferiores al máximo histórico registrado. Para mitigar estas zonas ciegas visuales, el algoritmo detecta la imagen estática y entra en un estado de alta frecuencia iterando capturas continuas hasta que el rodillo finaliza su transición. Esto explica por qué, a pesar de tener una menor densidad de puntos de datos enviados, el volumen de energía acumulada total a largo plazo coincide con el sensor óptico.

5.6 Consumo de Ancho de Banda y Tráfico de Red

El volumen de datos transmitidos hacia la red difiere radicalmente entre ambas arquitecturas, convirtiéndose en un factor crítico al evaluar la escalabilidad de cada solución. El prototipo óptico fundamenta su comunicación en el envío de tramas de texto plano hacia la nube (ThingSpeak). Considerando la carga útil y el overhead de los protocolos TCP/HTTP, cada transmisión presenta un peso aproximado de 4.25 KB.

Operando bajo el intervalo estándar de 15 minutos (96 envíos diarios), este sistema genera un tráfico acumulado marginal de apenas 408 KB por jornada.

En contraste, el sistema de visión requiere una arquitectura de red mucho más exigente. Al operar de manera distribuida, el microcontrolador ESP32-CAM debe transmitir cada captura fotográfica hacia el servidor local para su procesamiento OCR, para luego subir el resultado a la nube. Basado en la configuración del hardware, que captura imágenes en formato JPEG a una resolución UXGA (1600x1200) y baja compresión, cada archivo tiene un peso promedio de 250 KB. Puesto que este prototipo también envía el registro final a ThingSpeak, la carga de red por cada intervalo asciende a 254.25 KB.

Bajo la misma frecuencia de muestreo de 96 capturas diarias, el prototipo de visión inyecta un tráfico total de 23.84 MB diarios a la red local (un incremento cercano al 5882 % respecto al sensor óptico). Si bien en un escenario residencial aislado este consumo es manejable, los datos evidencian que escalar el enfoque de visión artificial masivamente requeriría infraestructura de red dedicada, mientras que el modelo óptico podría operar sobre redes de baja potencia y bajo ancho de banda (LPWAN) sin riesgo de congestión.

5.7 Detección de Actividades y Monitoreo en Entorno Real

Para validar la aplicabilidad práctica de la arquitectura propuesta en el contexto del cuidado de adultos mayores o personas que viven solas, el sistema de medición fue instalado en el medidor de un entorno residencial real. Dicha vivienda cuenta con una ocupación única (un solo residente), lo que representa un escenario ideal para aislar y correlacionar los eventos de consumo eléctrico con las rutinas y la presencia física del individuo, eliminando el ruido estadístico que generaría la interferencia de múltiples habitantes.

Los datos recolectados por el prototipo óptico fueron transmitidos a la plataforma ThingSpeak, permitiendo generar representaciones gráficas de la actividad a lo largo de varias semanas. En la Figura 5.11 se presenta un mapa de calor que ilustra el consumo eléctrico por hora durante un periodo de 21 días (desde el 21 de julio hasta el 10 de agosto de 2026). La escala de colores representa la intensidad del consumo, donde los tonos cercanos al blanco indican el nivel basal del hogar y los tonos azules oscuros evidencian un consumo intensivo (alcanzando peaks superiores a 0.6 kWh por intervalo).

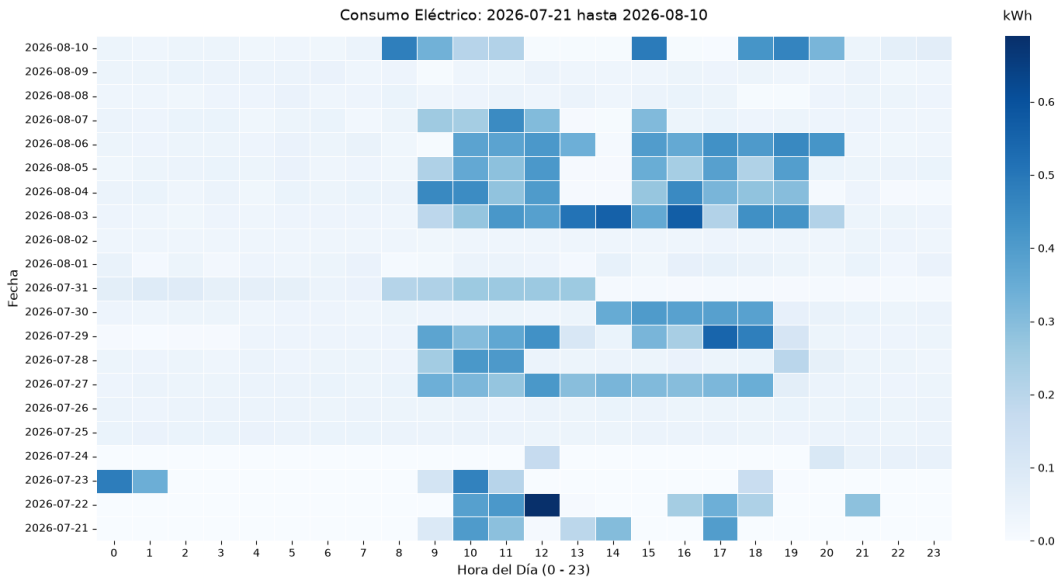


Figura 5.11: Mapa de calor del consumo eléctrico por hora en vivienda de ocupación única.

El análisis detallado de esta matriz térmica permite extraer conclusiones directas sobre los hábitos del residente. Como se destaca en la Figura 5.12, el sistema logra delimitar visualmente el ciclo de descanso diario. Se observa un patrón claro y repetitivo de inactividad durante las horas de la madrugada, específicamente en la franja comprendida entre las 00:00 y las 08:00 horas. Durante este bloque, el consumo decae estrictamente a niveles basales. Adicionalmente, el recuadro derecho de la misma figura evidencia una segunda franja de baja actividad entre las 20:00 y las 23:00 horas, lo que sugiere un cese en el uso de electrodomésticos de alta potencia hacia el final de

la jornada. Esta caracterización horaria es fundamental para el sistema de teleasistencia, ya que permite definir la "normalidad"del usuario y evitar falsas alarmas durante sus horas de sueño.

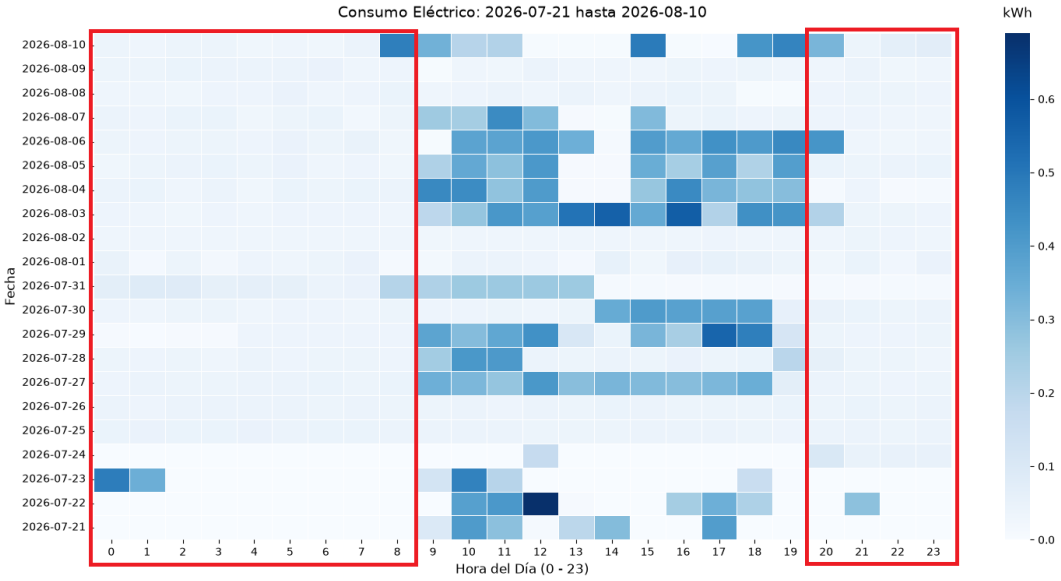


Figura 5.12: Franjas horarias de inactividad asociadas al ciclo de descanso.

Más allá de los hábitos diarios, el monitoreo longitudinal revela patrones macroscópicos de comportamiento. La Figura 5.13 exhibe una serie de anomalías donde existe una ausencia casi total de actividad que supere el consumo basal. Al contrastar estas fechas con el calendario, los bloques vacíos corresponden exactamente a los fines de semana: 25 y 26 de julio, 1 y 2 de agosto, y 8 y 9 de agosto.

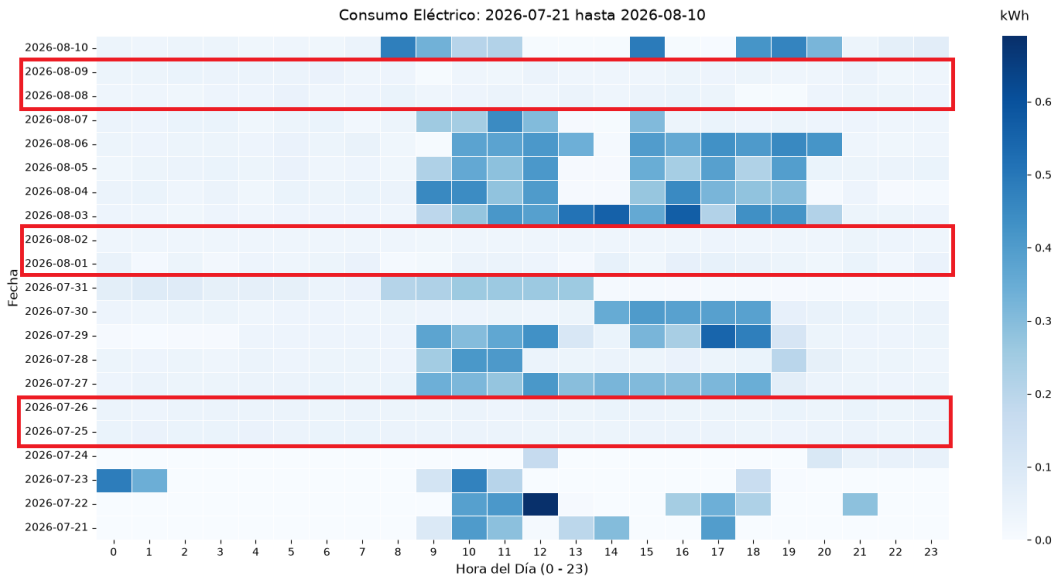


Figura 5.13: Detección de ausencia prolongada en la vivienda durante los fines de semana.

Estos bloques de bajo consumo demuestran empíricamente que el sistema es capaz de inferir de manera 100 % no intrusiva cuándo el residente abandona la vivienda por periodos prolongados. En el contexto del cuidado de un adulto mayor, esta capacidad de detección es crítica. Si un vacío de esta naturaleza ocurriera de manera atípica a mitad de semana, activaría exitosamente la lógica de alertas temporizadas descrita en la sección anterior (desencadenando las notificaciones de 12 y 24 horas de inactividad), advirtiendo a los familiares sobre una posible emergencia, caída prolongada o ausencia no planificada, cumpliendo así el objetivo principal de este proyecto.

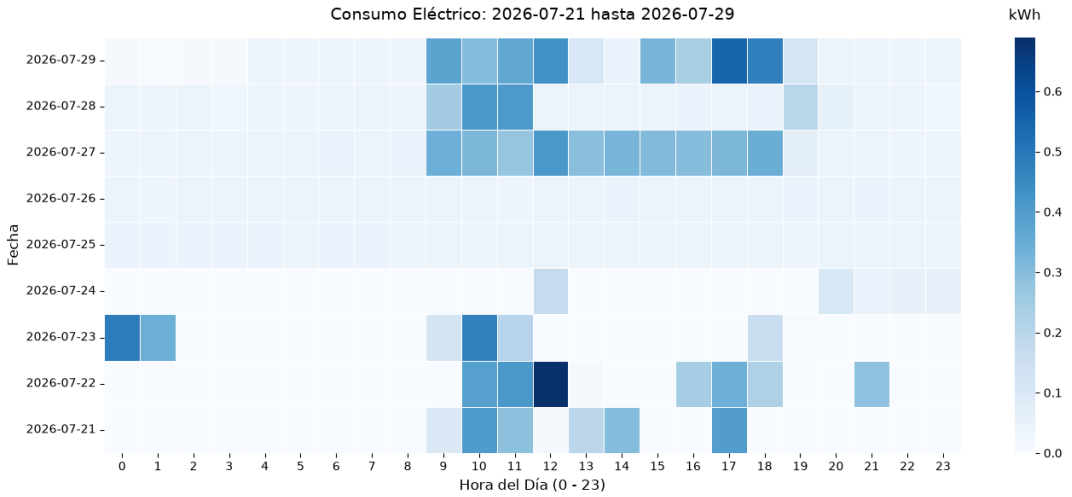
5.7.1 Optimización de la Frecuencia de Captura en el Prototipo de Visión

Como se evidenció en las pruebas de muestreo de alta frecuencia (20, 30 y 60 segundos), el prototipo de visión artificial basado en ESP32-CAM y Tesseract OCR local presentó severas limitaciones operativas. La naturaleza mecánica de los rodillos del medidor generó un alto índice de capturas descartadas cuando los dígitos se encontra-

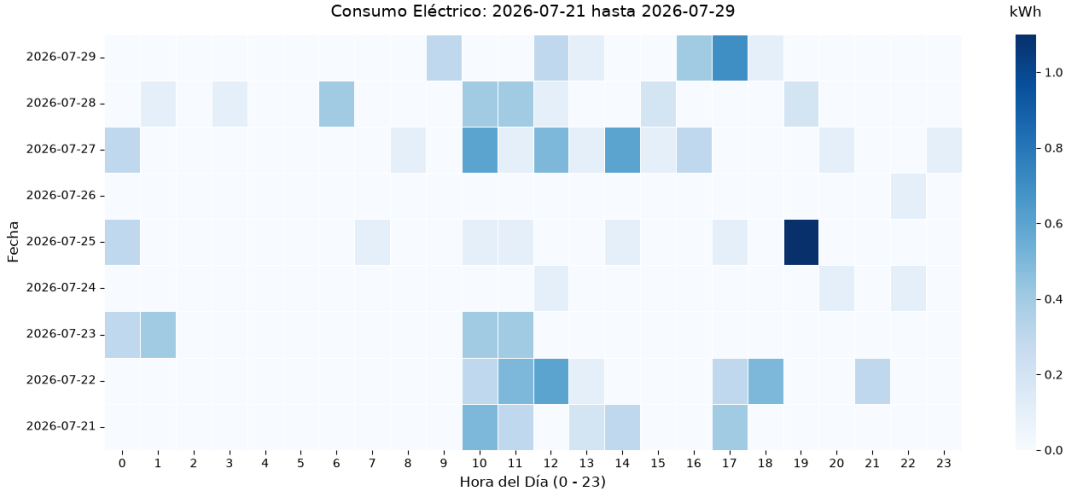
ban en transición, invalidando los datos continuos.

Para dimensionar el impacto real de esta limitación en la inferencia de rutinas, se realizó una prueba comparativa instalando ambos prototipos simultáneamente en el mismo medidor eléctrico residencial durante un periodo de 9 días (del 21 al 29 de julio). Al operar bajo el mismo consumo y en la misma ventana de tiempo, los datos transmitidos a la plataforma ThingSpeak revelaron una disparidad absoluta en la calidad de la información.

La Figura 5.14 contrasta los mapas de calor generados por ambos sistemas. Se evidencia claramente que el dispositivo óptico (Figura 5.14a) resulta inmensamente superior para deducir la actividad del usuario, delineando bloques de consumo consistentes y patrones diarios lógicos. En contraparte, el prototipo de visión (Figura 5.14b) arroja una matriz altamente fragmentada y errática. Las fallas de lectura del OCR no solo generan grandes vacíos de información (celdas en blanco donde sí hubo actividad), sino que las lecturas retrasadas provocan la acumulación artificial de energía en un solo intervalo, generando falsos picos (superiores a 1.0 kWh) que distorsionan por completo la realidad del hogar.



(a) Prototipo Óptico (Lectura fluida y fiable).



(b) Prototipo de Visión (Lectura fragmentada).

Figura 5.14: Comparativa de detección de actividad entre ambos prototipos.

Para mitigar este problema y evaluar si la arquitectura de visión era viable bajo un régimen menos estricto, se modificó la frecuencia de muestreo, espaciando las capturas fotográficas a intervalos de 15, 30 y 60 minutos.

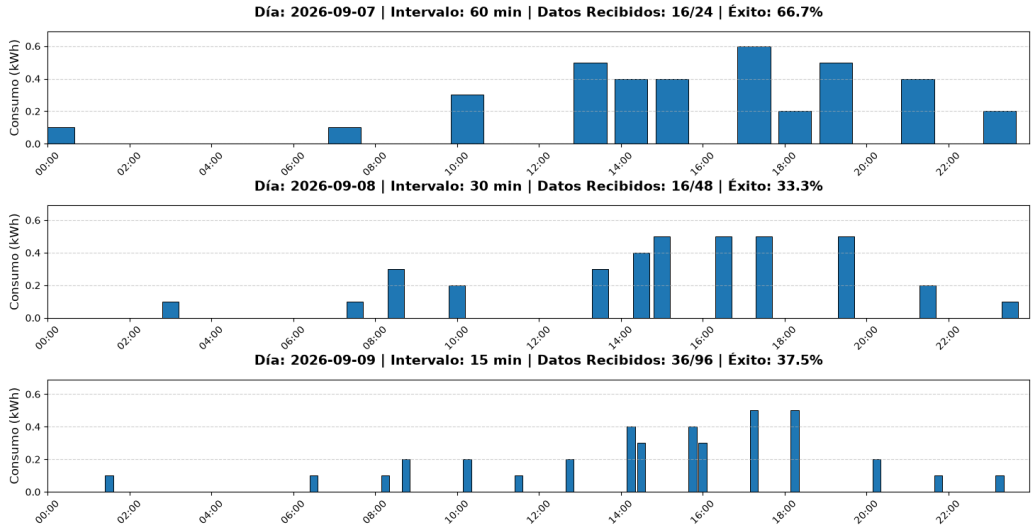


Figura 5.15: Comparativa de lecturas válidas versus esperadas para frecuencias de 60, 30 y 15 minutos.

Como se detalla en la Figura 5.15, el rendimiento del sistema se evaluó contrastando la cantidad de lecturas válidas recibidas en la plataforma frente al total teórico esperado para cada jornada. Operando a un intervalo de 60 minutos, el sistema exige 24 capturas diarias; en este escenario, se lograron registrar 16 datos, alcanzando un 66.7 % de éxito. Sin embargo, al exigir una mayor resolución temporal, el rendimiento del OCR decae drásticamente: al muestrear cada 30 minutos (48 capturas teóricas), la tasa de éxito se desplomó a un 33.3 % (solo 16 datos válidos). Finalmente, en el intervalo estándar de 15 minutos (96 capturas esperadas), el acierto se estancó en un deficiente 37.5 % (36 datos recibidos).

Al espaciar la ventana temporal entre capturas a 60 minutos, se observó estadísticamente una leve disminución en la probabilidad de fotografiar el rodillo en pleno movimiento, logrando ese tope del 66.7 % de efectividad. Sin embargo, si bien se nota una mejora marginal en la estabilidad de los datos respecto a frecuencias mayores, este avance está muy lejos de ser suficiente para concluir que la opción basada en la cámara sea la más viable para un ecosistema de monitoreo de salud.

Los ensayos demostraron que la tasa de descarte a los 15 y 30 minutos sigue siendo excesivamente alta (rondando el 65 % de pérdida), inutilizando dichos intervalos

para una detección continua y confiable. Por lo tanto, para poder emitir una conclusión definitiva sobre si el *hardware* fotográfico posee viabilidad a largo plazo, sería estrictamente necesario realizar campañas de muestreo por semanas o meses, operando de forma exclusiva a 60 minutos. No obstante, al someter el sistema a estos intervalos tan prolongados —obteniendo con suerte 16 puntos de datos útiles al día—, se sacrificaría irrevocablemente la granularidad temporal de la información. Esta baja densidad de datos impide identificar acciones cortas e interacciones vitales cotidianas, lo que termina por desvirtuar el propósito central del reconocimiento de Actividades de la Vida Diaria (ADL).

Capítulo 6

Conclusiones y trabajo futuro

Se implementó un sistema IoT no intrusivo orientado a la teleasistencia de adultos mayores, evaluando empíricamente dos enfoques de captura de datos en el borde (*Edge Computing*). El proyecto integró la adquisición de parámetros eléctricos domiciliarios, su procesamiento local y la transmisión hacia una plataforma en la nube (ThingSpeak) para su visualización. Se construyeron dos prototipos: uno basado en visión artificial (ESP32-CAM con Tesseract OCR) y otro de sensado óptico (BPW34 acoplado a un ESP32). Tras someterlos a rigurosas pruebas estandarizadas de frecuencia, estrés lumínico y sincronización de 48 horas en un laboratorio aislado, se validó la capacidad del sistema para inferir rutinas de actividad de manera segura, económica y respetando la absoluta privacidad del usuario.

6.1 Conclusiones

La evaluación empírica de las tecnologías de adquisición de datos evidenció diferencias operativas críticas entre ambos enfoques, consolidando al prototipo óptico como la solución definitiva y superior para el monitoreo domiciliario no intrusivo. A pesar de que el modelo de visión artificial logró digitalizar el consumo energético aislando el procesamiento de la nube, su desempeño real se vio severamente mermado por las características físicas del medidor tradicional; las transiciones de los rodillos mecánicos

provocaron significativas caídas en la tasa de éxito, la cual alcanzó apenas un 75.97 % frente al escenario de prueba. Esta limitación obligó al sistema de visión a recurrir a ráfagas de captura aceleradas para recuperar la sincronía, lo que a su vez incrementó exponencialmente la demanda energética, alcanzando peaks de 240 mA, y multiplicó el consumo de ancho de banda de la red local, superando en un 5882 % el tráfico generado por el sensor de luz. En contraste directo, el fotodiodo BPW34 demostró un rendimiento impecable, con una fiabilidad absoluta del 100 % en sus lecturas, un error de desfase acumulado insignificante del 0.20 % tras 48 horas de monitoreo continuo, y una exigencia de red marginal, validándose como la alternativa óptima para ecosistemas IoT con recursos limitados.

Desde la perspectiva de la arquitectura de software y el diseño del hardware, el proyecto cumplió a cabalidad con los Requisitos Funcionales (RF) y No Funcionales (RNF) definidos inicialmente. Se satisficieron los requerimientos asociados a la captura, procesamiento y transmisión de datos a la plataforma ThingSpeak, tanto para el modelo de visión (RF1 al RF4) como para el modelo de detección de pulsos (RF5 al RF7). La robustez estructural del sistema quedó demostrada al operar de forma estable y cumplir con los requisitos críticos de contingencia: la sincronización temporal precisa mediante servidores NTP (RF9) y la reconexión automática (RF10), mecanismos que garantizaron la integridad de los registros frente a intermitencias de la red WiFi. A nivel de la capa de aplicación, el panel en la nube satisfizo los RF11, RF12 y RF13 al generar de manera automatizada visualizaciones históricas, mapas de calor semanales y un indicador binario de actividad en el hogar basado en la discriminación analítica del consumo basal.

En lo que respecta a las estrictas restricciones de diseño impuestas por el contexto médico y residencial, el sistema respetó en su totalidad los Requisitos No Funcionales. La instalación se ejecutó de forma 100 % no intrusiva mediante un simple acople óptico superficial sobre el medidor, eliminando cualquier riesgo eléctrico por intervención galvánica (RNF1). Fundamentalmente, se dio cabal cumplimiento al mandato ético de privacidad (RNF2) al confinar todo el procesamiento analítico en el microcontrolador

local (*Full Edge*). El envío exclusivo de valores numéricos abstractos (kWh) a través de la red WiFi estándar (RNF3), en lugar de transmitir fotografías crudas o firmas eléctricas de alta resolución a servidores externos, protege la intimidad del adulto mayor y ayuda a disipar el estigma y la sensación de vigilancia que suele provocar el rechazo de este tipo de tecnologías asistenciales.

El logro medular de este proyecto radica en comprobar científicamente que no es necesario implementar complejas arquitecturas de desagregación de electrodomésticos en la nube para extraer información clínicamente valiosa sobre el bienestar de una persona. La transformación del consumo agregado en índices de regularidad y mapas térmicos resulta ser una herramienta poderosa y escalable. En conclusión, el monitoreo pasivo de Actividades de la Vida Diaria mediante la lectura óptica de medidores es una solución tecnológicamente madura, operativamente confiable, segura y altamente económica para apoyar el cuidado a distancia de la creciente población mayor.

6.2 Trabajo Futuro

A partir de los resultados experimentales obtenidos y la validación de la presente arquitectura IoT, se desprenden diversas líneas de investigación y desarrollo orientadas a robustecer el sistema y escalarlo hacia aplicaciones clínicas más complejas. En primer lugar, a nivel de la capa de hardware y captura de datos, resulta imperativo evolucionar desde el actual prototipo ensamblado en placa de pruebas (*protoboard*) hacia el diseño y fabricación de un circuito impreso (PCB) dedicado. Esta transición no solo minimizará la huella física del dispositivo, facilitando su instalación en nichos de medidores confinados, sino que aumentará significativamente su durabilidad ante factores ambientales. Además, para satisfacer cabalmente la exigencia de una operación prolongada e ininterrumpida por baterías (como lo demanda el RF8), será necesario optimizar el *firmware*, incorporando rutinas de estado de suspensión profunda (*Deep Sleep*) en el microcontrolador ESP32 durante los lapsos de inactividad, maximizando así la eficiencia energética del nodo sensor.

En el ámbito de la transmisión de datos y escalabilidad comunitaria, si bien el uso de WiFi y peticiones HTTP demostró ser altamente efectivo para validaciones de concepto en una vivienda aislada, su dependencia de la infraestructura del residente representa un punto único de fallo. Por consiguiente, el trabajo futuro debe contemplar la integración del hardware con tecnologías de redes de área amplia y bajo consumo (LPWAN), específicamente LoRaWAN, tal como se evaluó en el análisis inicial de alternativas. Esta migración permitiría establecer una topología de red en estrella donde múltiples hogares transmitan su telemetría de forma autónoma hacia un único *gateway* central, independizando el monitoreo médico de los cortes de internet domiciliarios.

Desde la perspectiva del análisis de la señal y el reconocimiento de patrones, el enfoque binario implementado (identificación de actividad frente a consumo basal) estableció un marco de teleasistencia robusto. Sin embargo, la alta resolución y precisión lograda por el fotodiodo abre la puerta a la implementación de un paradigma de análisis en tiempo real mediante algoritmos de *Machine Learning* ejecutados en el borde. Como horizonte de desarrollo, se propone entrenar modelos ligeros, como *Random Forest* o redes neuronales adaptativas, capaces de identificar transiciones y escalones de potencia sin requerir una desagregación exhaustiva y propensa a sobreajustes. Siguiendo los hallazgos en la literatura de reconocimiento de actividad, agrupar las variaciones de carga en categorías funcionales amplias —por ejemplo, un "grupo de cocina" para el uso del hervidor o microondas, y un "grupo de entretenimiento" para televisores— permitiría inferir de forma mucho más rica el tipo de Actividad de la Vida Diaria (ADL) que realiza el usuario. Adicionalmente, la exploración de modelos predictivos, como *signal2point*, facilitaría la anticipación de umbrales de consumo y la detección temprana de anomalías conductuales.

Por último, la validación definitiva del *framework* desarrollado demanda su despliegue más allá de un ambiente controlado. Se proyecta la ejecución de estudios longitudinales y proyectos piloto en hogares reales de adultos mayores que viven solos. Correlacionar los mapas de calor y los perfiles de consumo generados por la plataforma en la nube con diarios de actividad clínica y evaluaciones geriátricas, permitirá

calibrar empíricamente las alarmas del sistema. Este cruce de datos interdisciplinario transformará la arquitectura técnica aquí probada en una herramienta sociosanitaria preventiva y esencial para la preservación de la autonomía en la tercera edad.

Bibliografía

[1] Instituto Nacional de Estadísticas (INE). Proyecciones de Población. https://www.ine.cl/estadisticas/sociales/demografia-y-vitales/proyecciones-de-poblacion, 2024. Accedido: 15-05-2024.

[2] 2020health Workshop. Smart Future of Healthcare. https://2020health.org/wp-content/uploads/2020/11/2020health-SFoH-Full-report.pdf, 2020. Accedido: 15-05-2024.

[3] P. Franco, F. Condón, J. M. Martínez, and M. A. Ahmed. Enabling Remote Elderly Care: Design and Implementation of a Smart Energy Data System with Activity Recognition. *Sensors*, 23(18):7936, 2023. doi:10.3390/s23187936.

[4] Ministerio de Energía de Chile. Ministra de energía: "los medidores inteligentes tendrán múltiples beneficios para los clientes". https://energia.gob.cl/noticias/nacional/ministra-de-energia-los-medidores-inteligentes-tendran-multiples-beneficios-para-los-clientes, 2024. Accedido: 20-05-2024.

[5] Compañía General de Electricidad (CGE). Medición inteligente. https://www.cge.cl/datos-utiles/medicion-inteligente/, 2024. Accedido: 22-09-2025.

[6] Enel Distribución Chile. Medición inteligente. https://www.enel.cl/es/clientes/informacion-util/medicion-inteligente.html, 2024. Accedido: 22-09-2025.

[7] George W. Hart. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12):1870–1891, 1992. doi:10.1109/5.192069.

[8] Ahmad Zoha, Alexander Gluhak, Muhammad A. Imran, and Sutharshan Rajasegarar. Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey. *Sensors*, 12(12):16838–16866, 2012. doi:10.3390/s121216838.

[9] Jana Huchtkoetter and Andreas Reinhardt. A study on the impact of data sampling rates on load signature event detection. *Energy Informatics*, 2(Suppl 1):24, 2019. doi:10.1186/s42162-019-0096-9.

[10] Riccardo Iacobucci, Fabio Turrini, Stefano Rinaldi, and Stefano Panzieri. Data collection and cloud processing architecture applied to nilm techniques for independent living. In *2021 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 1–6. IEEE, 2021. doi:10.1109/I2MTC50364.2021.9460046.

[11] Yu Zhuang, Peng Chen, Yang Zhao, and Kai Li. Towards energy-efficient smart homes via precise nilm. *International Journal of Energy Research*, 47(5):1851–1868, 2023. doi:10.1002/er.8837.

[12] Comisión Nacional de Energía (CNE). Informe sobre medidores inteligentes y su impacto en tarifas. Technical report, Comisión Nacional de Energía de Chile, 2019. Accedido: 22-09-2025.

[13] Iec ts 63297:2021 – sensing devices for non-intrusive load monitoring (nilm) systems. Technical Specification IEC TS 63297:2021, International Electrotechnical Commission, Geneva, Switzerland, 2021. Edition 1.0. URL: https://webstore.iec.ch/publication/62918.

[14] Norbert Noury, Marc Berenguer, Henri Teyssier, Marie-Jeanne Bouzid, and Michel Giordani. Building an index of activity of inhabitants from their activity on

the residential electrical power line. *IEEE Transactions on Information Technology in Biomedicine*, 15(5):758–766, 2011.

[15] Jiajia Gao, Weijia Kong, Xiaoxiang Jiang, Zhe Wang, and Youjie Ma. Event detection and load identification for non-intrusive load monitoring using a wdsdm-based approach and weighted knn. *Energies*, 13(4):888, 2020. doi:10.3390/en13040888.

[16] Álvaro Hernández, Rubén Nieto, Laura de Diego-Otón, María Carmen Pérez-Rubio, José M. Villadangos-Carrizo, Daniel Pizarro, and Jesús Ureña. Detection of anomalies in daily activities using data from smart meters. *Sensors*, 24(2):515, 2024. doi:10.3390/s24020515.

[17] Álvaro Hernández, Rubén Nieto, Laura de Diego-Otón, José M. Villadangos-Carrizo, Daniel Pizarro, David Fuentes, and María C. Pérez-Rubio. Monitoring daily activities in households by means of energy consumption measurements from smart meters. *Journal of Sensor and Actuator Networks*, 14(2):25, 2025. doi:10.3390/jsan14020025.

[18] Mateusz Brzozowski and Artur Janicki. Real-time energy measurement for non-intrusive well-being monitoring of elderly people – a case study. *arXiv preprint arXiv:2407.00524*, 2024. arXiv:2407.00524.

[19] Bhekumuzi M. Mathunjwa, Yu-Fen Chen, Tzung-Cheng Tsai, and Yeh-Liang Hsu. A lifestyle monitoring system for older adults living independently using low-resolution smart meter data. *Sensors*, 24(11):3662, 2024. doi:10.3390/s24113662.

[20] Annalisa Barsotti, Gerl Armin, Wilhelm Sebastian, Massimiliano Donati, Stefano Dalmiani, and Claudio Passino. The complexity of ehealth architecture: Lessons learned from application use cases. *Computers*, 14(9):371, 2025. doi:10.3390/computers14090371.

[21] Victoria Hall, Richard Wong, Louise Rogerson, Phillip Stimpson, and Bertha Ochieng. The howz (smart meter) project: A planned service evaluation pilot study using smart analytics to monitor changes in the functional ability in the daily lives of older adults. Technical report, Centre for Primary Care Research at De Montfort University, June 2023. Funded by the East Midlands Academic Health Science Network.

[22] Voltaware. Amazon shop - energy monitoring solutions. https://voltaware.com/amazon-shop, 2026. Accedido el 13 de agosto de 2026.

[23] Fludia. FM432e – IoT electricity consumption sensor (LoRaWAN). https://shop.fludia.com/shop/en/isoFR/4-fm432e.html, 2026. Accedido el 13 de agosto de 2026.

[24] MCI Electronics. Cámara de detección inalámbrica WiFi de bajo consumo – Milesight X1 Sensing Camera. https://mcielectronics.cl/shop/product/camara-de-deteccion-x1-milesight/, 2026. Accedido el 13 de agosto de 2026.

[25] EDF Group. Howz: A smart home for the elderly – EDF Pulse Awards Winners 2017. https://www.edf.fr/en/pulse/edf-pulse-awards-winners-2017-howz, 2017. Accedido el 13 de agosto de 2026.

[26] Smilics Technologies. *Línea de Productos Wibeee 2024*. Smilics Technologies S.L., November 2024. Catálogo de productos. Accedido el 13 de agosto de 2026. URL: https://smilics.com/wp-content/uploads/2024/11/Wibeee_LineadeProductos_2024.pdf.

[27] Peacefair Electronic. *PZEM-004T V3.0 Datasheet and User Manual*. Peacefair Electronic, 2019. Consultado: Octubre 2025. URL: https://innovatorsguru.com/wp-content/uploads/2019/06/PZEM-004T-V3.0-Datasheet-User-Manual.pdf.

[28] MechatronicStore. Voltímetro amperímetro pzem-004t 260v con salida ttl. Página de producto. URL: https://www.mechatronicstore.cl/voltimetro-amperimetro-pzem-004t-260v-con-salida-ttl/.

[29] Analog Devices. *ADE9153A Datasheet*. Analog Devices, 2021. Consultado: Octubre 2025. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/ade9153a.pdf.

[30] Analog Devices. Ev-ade9153ashieldz - placas y kits de evaluación - módulos de shield. Página de producto. URL: https://www.mouser.cl/ProductDetail/Analog-Devices/EV-ADE9153ASHIELDZ.

[31] Generalitat de Catalunya. *Manual de Usuario CERMI*. Generalitat de Catalunya, 2020. Consultado: Octubre 2025. URL: https://web.gencat.cat/web/shared/GENCAT/que_cal_fer_si/serveis_centrals/que_fer_quan_especials/queferquan/vull-consumir-energia-responsable/reduir-factura/model-facturacio-hores/Manual-CERM1-1.pdf.

[32] Vishay Telefunken. *Silicon PIN Photodiode: BPW34*, 1999. Document Number 81521, Rev. 2, 20-May-99.

[33] Global Electrónica. Foto diodo bpw34, 2026. URL: https://globalelectronica.cl/881-foto-diodos/bpw34.html.

[34] Shenzhen Ai-Thinker Technology Co., Ltd. *ESP32-CAM Wi-Fi + BT SoC Module V1.0 Product Specifications*, 2017. Document version V1.0.

[35] Altronics. Placa esp32-cam, 2026. URL: https://altronics.cl/placa-esp32-cam.

[36] Roy T. Fielding, Mark Nottingham, and Julian Reschke. Http semantics. RFC 9110, 2022. URL: https://www.rfc-editor.org/info/rfc9110.

[37] LoRa Alliance. Lorawan technology overview, 2024. Consultado: Octubre 2025. URL: https://lora-alliance.org/about-lorawan.

[38] RAKwireless. Wisduo lpwan module - rak3172. Página de producto. URL: https://store.rakwireless.com/products/wisduo-lpwan-module-rak3172.

[39] Zamper. Gateway lorawan lps8n (interior). Página de producto. URL: https://zamper.cl/gateway-lorawan/gateway-lorawan-lps8n.

[40] Monografías. Zigbee estándar domótico e inmótica, 2023. Consultado: Octubre 2025. URL: https://www.monografias.com/trabajos61/zigbee-estandar-domotico-inmotica/zigbee-estandar-domotico-inmotica.

[41] MCI Electronics. Dongle usb sonoff zigbee 3.0 plus. Página de producto. URL: https://mcielectronics.cl/shop/product/dongle-usb-sonoff-zigbee-3-0-plus/.

[42] Espressif Systems. *ESP32 Development Board*. Espressif Systems, 2024. Consultado: Octubre 2025. URL: https://mcielectronics.cl/shop/product/tarjeta-de-desarrollo-de-esp-32-esp32-29541/.

[43] MCI Electronics. Tarjeta de desarrollo de esp-32 (esp32). Página de producto. URL: https://mcielectronics.cl/shop/product/tarjeta-de-desarrollo-de-esp-32-esp32-29541/.

[44] Amazon Web Services. *AWS IoT Developer Guide*. Amazon Web Services, 2024. Consultado: Octubre 2025. URL: https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html.

[45] Microsoft. *Azure IoT Hub Concepts*. Microsoft, 2024. Consultado: Octubre 2025. URL: https://learn.microsoft.com/es-es/azure/iot-hub/iot-concepts-and-iot-hub.

[46] IBM. *IBM Internet of Things Solutions*. IBM, 2024. Consultado: Octubre 2025. URL: https://www.ibm.com/es-es/solutions/internet-of-things.

[47] Raspberry Pi Foundation. *Raspberry Pi 4 Model B 8GB RAM*. Raspberry Pi Foundation, 2024. Consultado: Octubre 2025. URL: https://raspberrypi.cl/producto/raspberry-pi-4-modelo-b-8gb-ram.

[48] MathWorks. Thingspeak iot analytics. Plataforma IoT. URL: https://thingspeak.com/.

Anexos

Testeo del Prototipo Óptico

```
#define PIN_FOTODIODO 32
#define PIN_LED_EMISOR 19
// --- MODIFICA ESTE VALOR PARA CAMBIAR LA FRECUENCIA ---
const float FRECUENCIA_HZ = 8.0;

const unsigned long TIEMPO_ENCENDIDO = 80; // Duración fija del pulso (80 ms)
unsigned long periodoActual;
unsigned long tiempoInicioCiclo = 0;

// --- VARIABLES DEL CONTADOR ---
long pulsosTotales = 0;
bool pulsoDetectado = false;

void setup() {
    Serial.begin(115200);
    analogReadResolution(12);
    pinMode(PIN_LED_EMISOR, OUTPUT);
    pinMode(PIN_LED_INDICADOR, OUTPUT);
    // Calcula cuánto dura el ciclo completo en milisegundos
    periodoActual = 1000.0 / FRECUENCIA_HZ;
}

void loop() {
    unsigned long tiempoActual = millis();

    // --- 1. GENERACIÓN DEL PULSO ---
    unsigned long tiempoTranscurrido = tiempoActual - tiempoInicioCiclo;
    bool estadoLED = false;

    if (tiempoTranscurrido >= periodoActual) {
        tiempoInicioCiclo = tiempoActual;
        tiempoTranscurrido = 0;
    }

    if (tiempoTranscurrido < TIEMPO_ENCENDIDO) {
        estadoLED = true;
    } else {
        estadoLED = false;
    }
    digitalWrite(PIN_LED_EMISOR, estadoLED);

    // --- 2. LECTURA Y CONTEO ---
    int valor_luz = analogRead(PIN_FOTODIODO);
    int senalLimpia = 0;

    if (valor_luz > 4000) {
        senalLimpia = 10;
        digitalWrite(PIN_LED_INDICADOR, HIGH);

        // Cuenta el pulso solo una vez mientras la luz se mantenga alta
        if (!pulsoDetectado) {
            pulsosTotales++;
```

```
            pulsoDetectado = true;
        }
    } else {
        senalLimpia = 0;
        digitalWrite(PIN_LED_INDICADOR, LOW);
        pulsoDetectado = false;
    }

    // --- 3. GRAFICACIÓN Y MONITOREO ---
    // Imprimir para ver en el Serial Plotter (escala unificada)
    Serial.print("PulsoEmisor:");
    Serial.print(estadoLED ? 10 : 0);
    Serial.print(",");
    Serial.print("TotalPulsos:");
    Serial.println(pulsosTotales);

    delay(10);
}
```

Código 6.1: Código fuente para la prueba de frecuencia y saturación del fotodiodo.

Firmware del Prototipo Óptico

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <time.h>

#define PIN_FOTODIODO 32
#define PIN_LED_INDICADOR 19
#define PIN_LED_AZUL 2

const char* ssid = "iPhone de Bruno";
const char* password = "qqqqqqqq";
String apiKey = "ERLPBY5CGAI3HYJ1";

// --- Contadores y Variables de Estado ---
long pulsosTotales = 0;
long pulsosIntervalo = 0;
bool pulsoDetectado = false;

unsigned long tiempoInicio = 0;
const unsigned long INTERVALO_15MIN = 15*60*1000;
const unsigned long INTERVALO_5MIN  =  5*60*1000;
const unsigned long INTERVALO = INTERVALO_15MIN;

// --- BUFFER DE DATOS OFFLINE ---
const int MAX_BUFFER = 24; // Guarda hasta 24 datos
struct RegistroOffline {
    float energia;
    unsigned long tiempoMillis;
};
RegistroOffline bufferDatos[MAX_BUFFER];
int cantidadBuffer = 0;

unsigned long ultimoIntentoWiFi = 0;
unsigned long ultimoEnvioHTTP = 0;
bool ntpSincronizado = false;

void setup() {
    Serial.begin(115200);
    analogReadResolution(12);
    pinMode(PIN_LED_INDICADOR, OUTPUT);
    pinMode(PIN_LED_AZUL, OUTPUT);
    Serial.println("Iniciando sistema. Buscando WiFi...");
    WiFi.begin(ssid, password);
    tiempoInicio = millis();
}
```

Código 6.2: Configuración Inicial y Setup del ESP32 para Prototipo Óptico

```
void loop() {
    unsigned long tiempoActual = millis();

    // --- 0. CONTROL DE ESTADO WIFI Y LED AZUL ---
    if (WiFi.status() != WL_CONNECTED) {
        if ((tiempoActual / 500) % 2 == 0) {
            digitalWrite(PIN_LED_AZUL, HIGH);
        } else {
            digitalWrite(PIN_LED_AZUL, LOW);
        }

        if (tiempoActual - ultimoIntentoWiFi >= 30000) {
            Serial.println("Reintentando conexion WiFi...");
            WiFi.begin(ssid, password);
            ultimoIntentoWiFi = tiempoActual;
        }
        ntpSincronizado = false;
    } else {
        digitalWrite(PIN_LED_AZUL, LOW);

        if (!ntpSincronizado) {
            configTime(0, 0, "pool.ntp.org");
            ntpSincronizado = true;
            Serial.println("WiFi Conectado. Sincronizando reloj NTP...");
        }
    }

    // --- 1. LECTURA Y CONTEO ---
    int valor_luz = analogRead(PIN_FOTODIODO);
    int senalLimpia = 0;
    if (valor_luz > 4000) {
        senalLimpia = 4000;
        digitalWrite(PIN_LED_INDICADOR, HIGH);

        if (!pulsoDetectado) {
            pulsosTotales++;
            pulsosIntervalo++;
            pulsoDetectado = true;
        }
    } else {
        senalLimpia = 0;
        digitalWrite(PIN_LED_INDICADOR, LOW);
        pulsoDetectado = false;
    }
    float energiaKWh = (float)pulsosTotales / 1600.0;
```

Código 6.3: Gestión de conexión WiFi, sincronización NTP y lectura del BPW34

```
    // --- 2. GENERAR DATO POR INTERVALO Y GUARDAR EN BUFFER ---
    if (tiempoActual - tiempoInicio >= INTERVALO) {
        float energiaIntervalo = (float)pulsosIntervalo / 1600.0;

        if (cantidadBuffer < MAX_BUFFER) {
            bufferDatos[cantidadBuffer].energia = energiaIntervalo;
            bufferDatos[cantidadBuffer].tiempoMillis = tiempoActual;
            cantidadBuffer++;
            Serial.println("Guardado en Buffer.");
        } else {
            Serial.println("ALERTA Buffer lleno. Eliminando dato mas antiguo
                para guardar el nuevo.");

            for (int i = 0; i < MAX_BUFFER - 1; i++) {
                bufferDatos[i] = bufferDatos[i + 1];
            }
            bufferDatos[MAX_BUFFER - 1].energia = energiaIntervalo;
            bufferDatos[MAX_BUFFER - 1].tiempoMillis = tiempoActual;
        }

        pulsosIntervalo = 0;
        tiempoInicio = tiempoActual;
    }
```

Código 6.4: Lógica de almacenamiento en búfer circular offline

```
        // --- 3. ENVIAR BUFFER A THINGSPEAK (DOSIFICADO CADA 16 SEGS) ---
        if (WiFi.status() == WL_CONNECTED && cantidadBuffer > 0 && (tiempoActual -
            ultimoEnvioHTTP >= 16000)) {
            time_t now;
            time(&now);
            if (now > 1000000000) {
                unsigned long antiguedadDato = tiempoActual -
                    bufferDatos[0].tiempoMillis;
                time_t epochDelDato = now - (antiguedadDato / 1000);

                struct tm timeinfo;
                gmtime_r(&epochDelDato, &timeinfo);
                char timeString[25];
                strftime(timeString, sizeof(timeString), "%Y-%m-%d%%20%H:%M:%S",
                    &timeinfo);

                String url = "http://api.thingspeak.com/update?api_key=" + apiKey +
                "&field1=" + String(bufferDatos[0].energia, 4) +
                "&created_at=" + String(timeString);

                Serial.print(">>> Subiendo a TS [Buffer restan: ");
                Serial.print(cantidadBuffer - 1);
                Serial.println("] <<<");

                HTTPClient http;
                http.begin(url);
                int httpCode = http.GET();
                http.end();

                if (httpCode > 0) {
                    Serial.print("Exito HTTP: "); Serial.println(httpCode);

                    for (int i = 0; i < 3; i++) {
                        digitalWrite(PIN_LED_AZUL, LOW); delay(50);
                        digitalWrite(PIN_LED_AZUL, HIGH); delay(50);
                    }

                    for (int i = 0; i < cantidadBuffer - 1; i++) {
                        bufferDatos[i] = bufferDatos[i + 1];
                    }
                    cantidadBuffer--;
                } else {
                    Serial.println("Error subiendo dato. Se reintentara.");
                }

                ultimoEnvioHTTP = millis();
            }
        }
        // --- 4. IMPRESION CONSTANTE DE TODA LA DATA ---
        Serial.print("LuzCruda:");
        Serial.print(valor_luz);
        Serial.print(",PulsoLimpio:");
        Serial.print(senalLimpia);
        Serial.print(",PulsosTot:");
        Serial.print(pulsosTotales);
        Serial.print(",Energia_kWh:");
        Serial.println(energiaKWh, 4);

        delay(20);
    }
```

Código 6.5: Transmisión dosificada de datos a ThingSpeak mediante HTTP

Firmware del Módulo ESP32-CAM

```
#include "esp_camera.h"
#include <WiFi.h>
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "fb_gfx.h"
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "esp_http_server.h"

// --- 1. CONFIGURACIÓN DE WIFI ---
const char* ssid = "WiFi_Mesh-541486";
const char* password = "D7zEZDN2";

// --- 2. DEFINICIÓN DE PINES (MODELO AI THINKER) ---
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22
#define FLASH_GPIO_NUM     4

int flash_intensity = 150;

// --- 3. FUNCIONES DEL SERVIDOR WEB ---
// Tomar y enviar la foto (Aquí se enciende el flash)
esp_err_t capture_handler(httpd_req_t *req) {
    // 1. Encender el flash si la intensidad es mayor a 0
    if (flash_intensity > 0) {
        ledcWrite(FLASH_GPIO_NUM, flash_intensity);
        delay(1000); // Dar tiempo a la cámara para ajustar la exposición y balance
            de blancos
    }
    // 2. Tomar la foto
    camera_fb_t *fb = esp_camera_fb_get();
    // 3. Apagar el flash inmediatamente para no gastar energía ni generar calor
    ledcWrite(FLASH_GPIO_NUM, 0);
    if (!fb) {
        Serial.println("Error: Falló la captura de la cámara");
        httpd_resp_send_500(req);
        return ESP_FAIL;
    }

    httpd_resp_set_type(req, "image/jpeg");
    httpd_resp_set_hdr(req, "Content-Disposition", "inline; filename=captura.jpg");
    httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
    esp_err_t res = httpd_resp_send(req, (const char *)fb->buf, fb->len);
    esp_camera_fb_return(fb);
    return res;
}

void startCameraServer() {
    httpd_config_t config = HTTPD_DEFAULT_CONFIG();
    httpd_uri_t capture_uri = { .uri = "/capture", .method = HTTP_GET, .handler =
        capture_handler, .user_ctx = NULL };
    httpd_handle_t stream_httpd = NULL;
    if (httpd_start(&stream_httpd, &config) == ESP_OK) {
        httpd_register_uri_handler(stream_httpd, &capture_uri);
    }
}
```

```
// --- 4. CONFIGURACIÓN INICIAL ---
void setup() {
    WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
    Serial.begin(115200);
    Serial.setDebugOutput(true);
    // Configuración del Flash usando PWM (Sintaxis actualizada v3.x.x)
    ledcAttach(FLASH_GPIO_NUM, 5000, 8);
    ledcWrite(FLASH_GPIO_NUM, 0); // Iniciar apagado

    camera_config_t config;
    config.ledc_channel = LEDC_CHANNEL_0;
    config.ledc_timer = LEDC_TIMER_0;
    config.pin_d0 = Y2_GPIO_NUM;
    config.pin_d1 = Y3_GPIO_NUM;
    config.pin_d2 = Y4_GPIO_NUM;
    config.pin_d3 = Y5_GPIO_NUM;
    config.pin_d4 = Y6_GPIO_NUM;
    config.pin_d5 = Y7_GPIO_NUM;
    config.pin_d6 = Y8_GPIO_NUM;
    config.pin_d7 = Y9_GPIO_NUM;
    config.pin_xclk = XCLK_GPIO_NUM;
    config.pin_pclk = PCLK_GPIO_NUM;
    config.pin_vsync = VSYNC_GPIO_NUM;
    config.pin_href = HREF_GPIO_NUM;
    config.pin_sscb_sda = SIOD_GPIO_NUM;
    config.pin_sscb_scl = SIOC_GPIO_NUM;
    config.pin_pwdn = PWDN_GPIO_NUM;
    config.pin_reset = RESET_GPIO_NUM;
    config.xclk_freq_hz = 20000000;
    config.pixel_format = PIXFORMAT_JPEG;

    if(psramFound()){
        config.frame_size = FRAMESIZE_UXGA;
        config.jpeg_quality = 10;
        config.fb_count = 2;
    } else {
        config.frame_size = FRAMESIZE_SVGA;
        config.jpeg_quality = 12;
        config.fb_count = 1;
    }
    esp_err_t err = esp_camera_init(&config);
    if (err != ESP_OK) {
        Serial.printf("Error al iniciar cámara: 0x%x", err);
        return;
    }
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi Conectado");
    Serial.print("Dirección IP: ");
    Serial.println(WiFi.localIP());

    startCameraServer();

    Serial.println("Servidor Listo Entra a http://IP/capture");
}

void loop() {
    delay(10000);
}
```

Código 6.6: Servidor web, control PWM de flash y captura en ESP32-CAM.

Códigos de Visión Artificial (Script de Procesamiento Local)

```
    import cv2
    import numpy as np
    import requests
    import pytesseract
    import csv
    import os
    import json
    import time
    from datetime import datetime
    from collections import Counter

    # --- CONFIGURACIÓN ---
    # Se pueden sobreescribir con variables de entorno sin tocar el código
    ip_esp32 = os.environ.get("ESP32_IP", "192.168.1.43")
    url_captura = f"http://{ip_esp32}/capture"
    archivo_csv = "historico_lecturas.csv"
    archivo_estado = "estado_medidor.json"
    carpeta_fallos = "fallos_capturas"

    # Tiempo exacto que quieres que pase entre el inicio de cada foto (modo normal)
    INTERVALO_SEGUNDOS = 900.0

    # --- MODO RÁFAGA (captura acelerada tras fallos consecutivos) ---
    FALLOS_PARA_RAFAGA = 3          # a partir de cuántos fallos seguidos se evalúa
        activar ráfaga
    INTERVALO_RAFAGA = 60.0         # segundos entre capturas mientras esté en modo
        ráfaga
    MAX_INTENTOS_RAFAGA = 12        # tope de intentos en ráfaga antes de volver a
        la normalidad
    UMBRAL_CAMBIO_IMAGEN = 0.03     # % de píxeles distintos para considerar que "la
        imagen cambió"

    # --- SALTO MÁXIMO DINÁMICO ---
    SALTO_BASE = 0.3                # salto máximo normal (medidor leído cada
        INTERVALO_SEGUNDOS)
    SALTO_MAXIMO_CAP = 3.0          # tope absoluto, no importa cuántos fallos lleve
        acumulados

    # --- REINTENTOS DE RED ---
    INTENTOS_CAPTURA = 3
    ESPERA_BASE_REINTENTO = 2  # segundos, crece linealmente por intento

    # --- CONFIGURACIÓN THINGSPEAK ---
    TS_WRITE_API_KEY = os.environ.get("TS_WRITE_API_KEY", "DUZ0O6KF4CK55PF1")
    TS_CHANNEL_ID = os.environ.get("TS_CHANNEL_ID", "3432080")
    TS_URL = "https://api.thingspeak.com/update"

    # DIRECCION TESSERACT
    pytesseract.pytesseract.tesseract_cmd = r'C:\Program
        Files\Tesseract-OCR\tesseract.exe'
```

Código 6.7: Configuración general, parámetros de red y variables de entorno para OCR.

```
#------------------------- FUNCIONES ---------------------------------------

def nothing(x):
pass


def obtener_mejor_lectura(imagen_gris, nombre_zona, digitos_esperados, mini, maxi,
    lectura_esperada=None):
"""Realiza un barrido completo de umbrales y retorna la lectura más frecuente."""
resultados = []
for umbral in range(mini, maxi, 5):
# Procesamiento
_, binaria = cv2.threshold(imagen_gris, umbral, 255, cv2.THRESH_BINARY)
binaria_inv = cv2.bitwise_not(binaria)
```

```
# Configuración OCR
config_tess = f'--psm 7 -c tessedit_char_whitelist=0123456789'
lectura = pytesseract.image_to_string(binaria_inv, config=config_tess).strip()

# Validamos que tenga la cantidad de dígitos correcta
if len(lectura) == digitos_esperados:
# Si coincide con lo esperado, retornamos inmediatamente (restaurado)
if lectura == lectura_esperada:
return lectura

resultados.append(lectura)

# --- LÓGICA DE VALIDACIÓN ESTADÍSTICA ---
if len(resultados) >= 4:
conteo = Counter(resultados)
valor_mas_comun, frecuencia = conteo.most_common(1)[0]

porcentaje_confianza = (frecuencia / len(resultados)) * 100
if porcentaje_confianza > 65:
return valor_mas_comun
else:
print(f"--> [!] Confianza insuficiente para {nombre_zona}
    ({porcentaje_confianza:.2f}% <= 65%)")
return []

else:
print(f"--> [!] No se alcanzó el mínimo de 4 lecturas válidas (Total:
    {len(resultados)})")
return []


def procesar_lectura_memoria(valor_float):
valor_formateado = str(int(float(valor_float) * 10)).zfill(6)
esperado_3 = valor_formateado[:3]
esperado_2 = valor_formateado[3:5]
esperado_1 = valor_formateado[5:]
return esperado_3, esperado_2, esperado_1


def enviar_a_thingspeak(valor_kwh, delta_kwh=0.0):
try:
valor_str = "{:.1f}".format(valor_kwh)
delta_str = "{:.1f}".format(delta_kwh)

payload = {
    'api_key': TS_WRITE_API_KEY,
    'field1': valor_str,
    'field2': delta_str
}

requests.post(TS_URL, data=payload, timeout=5)
print(f">>> [Cloud] Enviado a TS - Lectura: {valor_str} kWh | Delta: {delta_str}
    kWh")
except Exception as e:
print(f"[!] Error de conexión TS: {e}")


def obtener_captura(intentos=INTENTOS_CAPTURA, espera_base=ESPERA_BASE_REINTENTO,
    timeout=15):
"""Descarga y decodifica una imagen de la ESP32-CAM, con reintentos ante fallas de
    red."""
for intento in range(1, intentos + 1):
try:
respuesta = requests.get(url_captura, timeout=timeout)
arr_imagen = np.array(bytearray(respuesta.content), dtype=np.uint8)
img = cv2.imdecode(arr_imagen, -1)
if img is not None:
return img
print(f"[!] Intento {intento}/{intentos}: imagen no decodificable.")
except Exception as e:
print(f"[!] Intento {intento}/{intentos} fallido al capturar: {e}")

if intento < intentos:
time.sleep(espera_base * intento)
```

```
return None


def hubo_cambio(imagen_a, imagen_b, umbral_pixeles=UMBRAL_CAMBIO_IMAGEN):
"""Compara dos recortes en escala de gris/BGR para saber si el medidor cambió
    visualmente."""
if imagen_a is None or imagen_b is None:
# Sin referencia previa no podemos comparar; asumimos que sí hubo cambio
# para no bloquear la activación de ráfaga innecesariamente.
return True
if imagen_a.shape != imagen_b.shape:
return True

diff = cv2.absdiff(imagen_a, imagen_b)
if len(diff.shape) == 3:
diff = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)
_, diff_bin = cv2.threshold(diff, 25, 255, cv2.THRESH_BINARY)
porcentaje_cambio = np.count_nonzero(diff_bin) / diff_bin.size
return porcentaje_cambio > umbral_pixeles


def calcular_salto_maximo(tiempo_ultimo_exito, intervalo_normal,
    salto_base=SALTO_BASE, cap=SALTO_MAXIMO_CAP):
"""Escala el salto máximo permitido según cuánto tiempo real ha pasado sin una
    lectura válida."""
if tiempo_ultimo_exito is None:
return salto_base

tiempo_transcurrido = time.time() - tiempo_ultimo_exito
ciclos_equivalentes = max(1.0, tiempo_transcurrido / intervalo_normal)
salto = salto_base * ciclos_equivalentes
return min(salto, cap)


def guardar_imagen_fallo(recorte, motivo):
try:
os.makedirs(carpeta_fallos, exist_ok=True)
ts = datetime.now().strftime("%Y%m%d_%H%M%S")
ruta = os.path.join(carpeta_fallos, f"{ts}_{motivo}.png")
cv2.imwrite(ruta, recorte)
except Exception as e:
print(f"[!] No se pudo guardar imagen de fallo: {e}")


def registrar_csv(timestamp, lectura, delta, estado, fallos_seguidos):
existe = os.path.exists(archivo_csv)
try:
with open(archivo_csv, "a", newline="") as f:
writer = csv.writer(f)
if not existe:
writer.writerow(["timestamp", "lectura_kwh", "delta_kwh", "estado",
    "fallos_consecutivos"])
writer.writerow([timestamp, lectura, delta, estado, fallos_seguidos])
except Exception as e:
print(f"[!] Error escribiendo CSV: {e}")


def guardar_estado(estado):
try:
with open(archivo_estado, "w") as f:
json.dump(estado, f)
except Exception as e:
print(f"[!] No se pudo guardar el estado: {e}")


def cargar_estado():
if os.path.exists(archivo_estado):
try:
with open(archivo_estado) as f:
return json.load(f)
except Exception as e:
print(f"[!] No se pudo leer el estado previo: {e}")
return None
```

Código 6.8: Funciones de captura, validación estadística y persistencia de estado.

```
    # --- INICIO DEL PROGRAMA ---
    X_FIXED = Y_FIXED = W_FIXED = H_FIXED = None
    estado_previo = cargar_estado()
    usar_estado_previo = False

    if estado_previo and all(k in estado_previo for k in ("X_FIXED", "Y_FIXED",
        "W_FIXED", "H_FIXED")):
    print(f"[*] Se encontró un estado guardado de una ejecución anterior:")
    print(f"    Máximo histórico: {estado_previo.get('max_en_memoria')} kWh | "
    f"Fallos seguidos: {estado_previo.get('fallos_consecutivos')} | "
    f"ROI: ({estado_previo['X_FIXED']}, {estado_previo['Y_FIXED']}, "
    f"{estado_previo['W_FIXED']}, {estado_previo['H_FIXED']})")
    resp = input("Retomar desde ese estado y omitir la calibración manual (s/n):
        ").strip().lower()
    if resp == 's':
    usar_estado_previo = True
    X_FIXED = estado_previo["X_FIXED"]
    Y_FIXED = estado_previo["Y_FIXED"]
    W_FIXED = estado_previo["W_FIXED"]
    H_FIXED = estado_previo["H_FIXED"]

    ultima_lectura_calibracion = None

    if not usar_estado_previo:
    print("1. Descargando imagen inicial para calibración...")
    try:
    img_original = obtener_captura()

    if img_original is None:
    raise Exception("No se pudo descargar/decodificar la imagen de la cámara tras
        varios intentos.")

    print("\n2. SELECCIÓN DE ÁREA -> Dibuja un rectángulo bien ajustado sobre los 6
        números y presiona ENTER.")
    roi = cv2.selectROI("SELECCIONA LOS NUMEROS", img_original, showCrosshair=True,
        fromCenter=False)
    cv2.destroyWindow("SELECCIONA LOS NUMEROS")

    if roi[2] > 0 and roi[3] > 0:
    X_FIXED, Y_FIXED, W_FIXED, H_FIXED = [int(v) for v in roi]
    recorte_inicial = img_original[Y_FIXED:Y_FIXED + H_FIXED, X_FIXED:X_FIXED +
        W_FIXED]
    gris_inicial = cv2.cvtColor(recorte_inicial, cv2.COLOR_BGR2GRAY)

    # --- MODO CALIBRACIÓN ---
    nombre_ventana = "Calibrador (r = Leer, q = Iniciar Sistema)"
    cv2.namedWindow(nombre_ventana)
    cv2.createTrackbar("Umbral", nombre_ventana, 130, 255, nothing)
    cv2.createTrackbar("Invertir", nombre_ventana, 1, 1, nothing)

    print("\n--- MODO CALIBRACIÓN PREVIA ---")
    print("- Ajusta el Umbral para ver cómo afecta la imagen.")
    print("- Presiona 'r' para ver qué detecta Tesseract con ese ajuste.")
    print("- Presiona 'q' cuando estés conforme para arrancar la recolección
        automática.")

    while True:
    val_umbral = cv2.getTrackbarPos("Umbral", nombre_ventana)
    val_invertir = cv2.getTrackbarPos("Invertir", nombre_ventana)

    _, binaria_calib = cv2.threshold(gris_inicial, val_umbral, 255,
        cv2.THRESH_BINARY)
    if val_invertir == 1:
    binaria_calib = cv2.bitwise_not(binaria_calib)

    cv2.imshow(nombre_ventana, binaria_calib)

    k = cv2.waitKey(1) & 0xFF
    if k == ord('q'):
```

```
    break
    elif k == ord('r'):
    config_tess = '--psm 7 -c tessedit_char_whitelist=0123456789'
    texto = pytesseract.image_to_string(binaria_calib, config=config_tess).strip()

    # Filtrar solo números para limpiar ruidos extraños
    texto_limpio = ''.join(filter(str.isdigit, texto))
    print(f"LECTURA DE PRUEBA (Umbral {val_umbral}): '{texto_limpio}'")

    if len(texto_limpio) == 6:
    valor_float = float(texto_limpio[:5]) + (float(texto_limpio[5]) / 10.0)
    ultima_lectura_calibracion = valor_float
    print(f"  [OK] Lectura válida guardada temporalmente: {valor_float} kWh")
    else:
    print(f"  [!] Ojo: Se leyeron {len(texto_limpio)} dígitos. Necesitamos 6. Sigue
        ajustando el umbral.")

    cv2.destroyAllWindows()
    print(f"\n[OK] Coordenadas fijadas: X={X_FIXED}, Y={Y_FIXED}, W={W_FIXED},
        H={H_FIXED}")

    else:
    print("[!] No seleccionaste nada. Cerrando programa.")
    exit()

    except Exception as e:
    print(f"Error en la calibración inicial: {e}")
    exit()

    # --- PREPARACIÓN DE DATOS E INICIO ---
    if usar_estado_previo:
    max_en_memoria = estado_previo.get("max_en_memoria", 0.0)
    fallos_consecutivos = estado_previo.get("fallos_consecutivos", 0)
    tiempo_ultimo_exito = estado_previo.get("tiempo_ultimo_exito", time.time())
    modo_rafaga = estado_previo.get("modo_rafaga", False)
    intentos_rafaga = estado_previo.get("intentos_rafaga", 0)
    print(f"\n[*] Retomando con máximo histórico: {max_en_memoria} kWh, "
    f"fallos consecutivos previos: {fallos_consecutivos}")
    elif ultima_lectura_calibracion is not None:
    print(f"\n[*] Usando última lectura de calibración como valor actual:
        {ultima_lectura_calibracion} kWh")
    max_en_memoria = ultima_lectura_calibracion
    delta_enviar = 0
    fallos_consecutivos = 0
    tiempo_ultimo_exito = time.time()
    modo_rafaga = False
    intentos_rafaga = 0

    lectura_str_inicial = f"{int(max_en_memoria * 10):06d}"
    enviar_a_thingspeak(max_en_memoria, delta_enviar)
    registrar_csv(datetime.now().isoformat(), max_en_memoria, delta_enviar,
        "OK_INICIAL", 0)
    else:
    print("\n[!] No se logró una lectura válida de 6 dígitos durante la
        calibración.")
    entrada_usuario = input("Por favor, ingresa la lectura actual del medidor
        manualmente (ej. 12345.6): ")
    delta_usuario = input("Por favor, ingresa delta(0.3): ")
    try:
    max_en_memoria = float(entrada_usuario)
    delta_enviar = float(delta_usuario)
    except ValueError:
    print("[!] Entrada inválida. Se iniciará en 0.0")
    max_en_memoria = 0.0
    delta_enviar = 0.0

    fallos_consecutivos = 0
    tiempo_ultimo_exito = time.time()
    modo_rafaga = False
    intentos_rafaga = 0

    enviar_a_thingspeak(max_en_memoria, delta_enviar)
    registrar_csv(datetime.now().isoformat(), max_en_memoria, delta_enviar,
        "OK_INICIAL", 0)
```

```
    recorte_anterior = None

    print(f"\n[*] Sistema listo para monitoreo. Máximo histórico actual fijado en:
        {max_en_memoria} kWh")
    print(f"[*] Intervalo configurado: {INTERVALO_SEGUNDOS} segundos por captura
        (modo normal).")
    print(f"[*] Modo ráfaga: se activa tras {FALLOS_PARA_RAFAGA} fallos seguidos con
        cambio de imagen, "
    f"cada {INTERVALO_RAFAGA}s, máximo {MAX_INTENTOS_RAFAGA} intentos.\n")
```

Código 6.9: Rutina de calibración manual, selección de ROI y recuperación de estado previo.

```
    # --- CICLO PRINCIPAL ---
    try:
    while True:
    tiempo_inicio_ciclo = time.time()
    ahora_momento = datetime.now().strftime("%H:%M:%S")
    etiqueta_modo = " [MODO RÁFAGA]" if modo_rafaga else ""
    print(f"[{ahora_momento}]{etiqueta_modo} Iniciando captura y procesamiento...")

    exito_lectura = False
    recorte_actual = None

    try:
    img_original = obtener_captura()

    if img_original is not None:
    recorte_actual = img_original[Y_FIXED:Y_FIXED + H_FIXED, X_FIXED:X_FIXED +
        W_FIXED]
    gris_completo = cv2.cvtColor(recorte_actual, cv2.COLOR_BGR2GRAY)

    mitad_w = W_FIXED // 2
    parte_izquierda = gris_completo[:, :mitad_w]
    parte_derecha = gris_completo[:, mitad_w:]

    w_sub = parte_derecha.shape[1] // 3
    zona_2 = parte_derecha[:, :w_sub * 2]
    zona_3 = parte_derecha[:, w_sub * 2:]

    esp1, esp2, esp3 = procesar_lectura_memoria(max_en_memoria)

    mejor_p1 = obtener_mejor_lectura(parte_izquierda, "ZONA 1", 3, 70, 170, esp1)
    mejor_p2 = obtener_mejor_lectura(zona_2, "ZONA 2", 2, 70, 170, esp2)
    # Zona 3 se deja sin lectura_esperada a propósito
    mejor_p3 = obtener_mejor_lectura(zona_3, "ZONA 3", 1, 120, 240)

    if (mejor_p1 and mejor_p2 and mejor_p3):
    mas_frecuente = mejor_p1 + mejor_p2 + mejor_p3
    lectura_float = float(mas_frecuente[:5]) + (float(mas_frecuente[5]) / 10.0)

    salto_maximo = calcular_salto_maximo(tiempo_ultimo_exito, INTERVALO_SEGUNDOS)

    if lectura_float >= max_en_memoria:
    if lectura_float <= max_en_memoria + salto_maximo:
    delta = round(lectura_float - max_en_memoria, 1)
    enviar_a_thingspeak(lectura_float, delta)
    registrar_csv(datetime.now().isoformat(), lectura_float, delta, "OK",
        fallos_consecutivos)

    max_en_memoria = lectura_float
    exito_lectura = True
    else:
    print(f"[!] Descartado: Salto ilógico ({lectura_float} >>> {max_en_memoria}, "
    f"máximo permitido ahora: {salto_maximo:.2f})")
    registrar_csv(datetime.now().isoformat(), lectura_float, None, "SALTO_ILOGICO",
        fallos_consecutivos)
    else:
    print(f"[!] Descartado: Lectura menor al histórico ({lectura_float} <
        {max_en_memoria})")
    registrar_csv(datetime.now().isoformat(), lectura_float, None,
        "MENOR_A_HISTORICO", fallos_consecutivos)
```

```
    else:
    print("\n[!] No se detectaron los 6 dígitos con suficiente confianza.")
    registrar_csv(datetime.now().isoformat(), None, None, "OCR_FALLIDO",
        fallos_consecutivos)
    else:
    print("[!] Error al decodificar/obtener imagen.")
    registrar_csv(datetime.now().isoformat(), None, None, "SIN_IMAGEN",
        fallos_consecutivos)

    except Exception as e:
    print(f"[!] Error en el ciclo: {e}")
    registrar_csv(datetime.now().isoformat(), None, None, f"ERROR:{e}",
        fallos_consecutivos)
```

Código 6.10: Bucle principal de captura periódica y validación de saltos en la lectura del medidor.

```
    # --- CONTADOR DE FALLOS Y MODO RÁFAGA ---
    if exito_lectura:
    fallos_consecutivos = 0
    tiempo_ultimo_exito = time.time()
    if modo_rafaga:
    print("[*] Lectura recuperada. Saliendo de modo ráfaga, se vuelve al intervalo
        normal.\n")
    modo_rafaga = False
    intentos_rafaga = 0
    else:
    fallos_consecutivos += 1
    if recorte_actual is not None:
    guardar_imagen_fallo(recorte_actual, "rafaga" if modo_rafaga else "normal")

    if fallos_consecutivos >= FALLOS_PARA_RAFAGA:
    cambio = hubo_cambio(recorte_anterior, recorte_actual)
    if cambio:
    if not modo_rafaga:
    print(f"[!] {fallos_consecutivos} fallos seguidos y la imagen SÍ cambió -> "
    f"posible desincronización de tiempos. Activando modo ráfaga.")
    modo_rafaga = True
    intentos_rafaga = 0
    intentos_rafaga += 1
    if intentos_rafaga >= MAX_INTENTOS_RAFAGA:
    print(f"[!] Se alcanzó el máximo de intentos en modo ráfaga
        ({MAX_INTENTOS_RAFAGA}). "
    f"Se vuelve al intervalo normal para no saturar la cámara.")
    modo_rafaga = False
    intentos_rafaga = 0
    else:
    if modo_rafaga:
    print(f"[!] {fallos_consecutivos} fallos seguidos, la imagen dejó de cambiar -> "
    f"medidor probablemente en transición mecánica. Saliendo de modo ráfaga.")
    else:
    print(f"[!] {fallos_consecutivos} fallos seguidos, la imagen no cambió respecto
        a la "
    f"anterior (medidor entre dos dígitos, no es un problema de sincronización).")
    modo_rafaga = False
    intentos_rafaga = 0

    if recorte_actual is not None:
    recorte_anterior = recorte_actual

    guardar_estado({
        "max_en_memoria": max_en_memoria,
        "fallos_consecutivos": fallos_consecutivos,
        "tiempo_ultimo_exito": tiempo_ultimo_exito,
        "modo_rafaga": modo_rafaga,
        "intentos_rafaga": intentos_rafaga,
        "X_FIXED": X_FIXED,
        "Y_FIXED": Y_FIXED,
        "W_FIXED": W_FIXED,
        "H_FIXED": H_FIXED,
    })

    tiempo_fin_ciclo = time.time()
```

```
    duracion_ciclo = tiempo_fin_ciclo - tiempo_inicio_ciclo
    print(f"[*] Duración del procesamiento: {duracion_ciclo:.2f} segundos")

    intervalo_actual = INTERVALO_RAFAGA if modo_rafaga else INTERVALO_SEGUNDOS
    tiempo_espera = intervalo_actual - duracion_ciclo

    if tiempo_espera > 0:
    print(f"[*] Durmiendo por {tiempo_espera:.2f} segundos...\n")
    time.sleep(tiempo_espera)
    else:
    print(f"[!] Advertencia: El procesamiento tomó más de {intervalo_actual}s.
        Iniciando siguiente de inmediato.\n")

    except KeyboardInterrupt:
    print("\n[!] Monitoreo detenido por el usuario.")
```

Código 6.11: Lógica de control de errores, activación del modo ráfaga y temporización del ciclo.

```
    % Parametros del canal y API
    channelID = 3427558;
    readAPIKey = 'EECMDJX58MGEW8NG';
    alertsAPIKey = 'TAK4yh6k+5GbQ1E0lkv';
    alertUrl = 'https://api.thingspeak.com/alerts/send';

    % 1. Extraer los datos (Ultimos 7 dias para basal, Ultimas 25h para actividad)
    tiempo_fin = datetime('now');
    hora_actual_cerrada = dateshift(tiempo_fin, 'start', 'hour');
    tiempo_inicio_25h = hora_actual_cerrada - hours(24);
    tiempo_inicio_7d = tiempo_fin - days(2);

    [data_25h, time_25h] = thingSpeakRead(channelID, 'Fields', 1, 'DateRange', ...
    [tiempo_inicio_25h, tiempo_fin], 'ReadKey', readAPIKey);
    [data_7d, time_7d] = thingSpeakRead(channelID, 'Fields', 1, 'DateRange', ...
    [tiempo_inicio_7d, tiempo_fin], 'ReadKey', readAPIKey);

    % 2. CALCULO DEL BASAL (Mediana de la madrugada de los ultimos 7 dias)
    consumo_basal = 0;
    if ~isempty(data_7d)
    horas_7d = hour(time_7d);
    idx_mad = (horas_7d >= 2) & (horas_7d < 6);
    time_mad = time_7d(idx_mad);
    data_mad = data_7d(idx_mad);

    time_mad_horas = dateshift(time_mad, 'start', 'hour');
    horas_unicas_mad = unique(time_mad_horas);
    consumo_mad_por_hora = zeros(length(horas_unicas_mad), 1);

    for i = 1:length(horas_unicas_mad)
    consumo_mad_por_hora(i) = sum(data_mad(time_mad_horas == horas_unicas_mad(i)),
        'omitnan');
    end

    if ~isempty(consumo_mad_por_hora)
    consumo_basal = median(consumo_mad_por_hora, 'omitnan');
    end
    end

    % 3. LOGICA DE ACTIVIDAD EN LAS ULTIMAS 25 HORAS
    if ~isempty(data_25h) && consumo_basal > 0
    umbral_actividad = consumo_basal * 1.3;

    % Agrupar datos por hora para comparar consumos equivalentes
    bordes_horas = tiempo_inicio_25h : hours(1) : hora_actual_cerrada;
    consumo_25h = zeros(25, 1);

    for i = 1:25
    h_in = bordes_horas(i);
    idx = (time_25h >= h_in) & (time_25h < (h_in + hours(1)));
    consumo_25h(i) = sum(data_25h(idx), 'omitnan');
    end

    % Buscar el ultimo bloque de hora que supero el umbral
```

```
    idx_actividad = find(consumo_25h > umbral_actividad, 1, 'last');

    if isempty(idx_actividad)
    horas_inactivo = 25;
    else
    ultima_actividad = bordes_horas(idx_actividad);
    horas_inactivo = hours(hora_actual_cerrada - ultima_actividad);
    end

    % 4. LOGICA DE DISPARO DE ALERTAS CON CONTROL TEMPORAL
    options = weboptions('HeaderFields', {'ThingSpeak-Alerts-API-Key',
        alertsAPIKey});

    % Solo envia el correo si esta en la ventana exacta para evitar saturacion
    if (horas_inactivo >= 12) && (horas_inactivo < 13)
    subject = 'Alerta de Inactividad: 12 Horas';
    body = sprintf('El sistema no ha registrado actividad electrica superior al
        consumo basal (%.3f) en las ultimas 12 horas.', consumo_basal);
    webwrite(alertUrl, 'body', body, 'subject', subject, options);

    elseif (horas_inactivo >= 24) && (horas_inactivo < 25)
    subject = 'Alerta Critica: 24 Horas sin actividad';
    body = 'No se ha detectado ninguna variacion electrica en 24 horas. Revisar
        estado de las conexiones y de la vivienda.';
    webwrite(alertUrl, 'body', body, 'subject', subject, options);
    end
    end
```

Código 6.12: Script de MATLAB para la detección de inactividad y envío de alertas temporizadas.