INSTITUTO UNIVERSITARIO AERONÁUTICO

SISTEMAS EN TIEMPO REAL

Proyecto integrador 2026

"Sistema de monitoreo en tiempo real de alumbrado público"

GRUPO 8

Cernik, Joaquin

Friesen, Eduardo

Bidone, Andres

Mousist, Jorge Martin

1. Índice

1. Índice....2
2. Introducción....3
2.1. Contexto....3
2.2. Motivación....4
3. Definición de problema....5
4. Justificación....6
5. Objetivo general....8
6. Metodología....8
7. Hardware....9
8. Software: Código del microcontrolador....15
8.1. Gestión de Tiempo Real con FreeRTOS....16
Arquitectura General....16
Tipo de Sistema en Tiempo Real: Suave (Soft Real-Time)....17
Sincronía vs Asincronía: Event-Driven Asíncrono....17
Patrón de uso de colas....19
Medición de Corriente y Tensión (ADS1115)....20
Hardware....20
Cálculo RMS (ventana deslizante de 20ms)....20
Filtrado EMA (Exponential Moving Average)....21
Consideraciones sobre el Tiempo Real....21
9. Software: Infraestructura del server de la nube....22
Envío de datos mediante mqtt....22
Nodered....23
Influxdb....24
Grafana....25
10. Conclusión....28

2. Introducción

Se presenta una propuesta para el desarrollo de un prototipo funcional de un sistema embebido e integral diseñado para la empresa ICOR SAS, organización dedicada a la gestión, operación y mantenimiento de infraestructura de alumbrado público. Esta prueba de concepto permite sensar, monitorear y generar reportes en tiempo real de variables eléctricas (tensión y corriente), además de supervisar el estado físico de los tableros mediante sensores de apertura de puerta y reconocimiento facial.

Dada la naturaleza de la ingeniería informática, la solución se enfoca prioritariamente en la arquitectura de software y la seguridad de la información. El sistema integra una plataforma web para la visualización dinámica de datos y una gestión de accesos robusta mediante políticas de autenticación de múltiple factor (MFA). Asimismo, implementa una lógica de notificaciones automáticas a través de Telegram, que permite el envío de alertas ante anomalías eléctricas y la remisión inmediata de registros fotográficos al identificarse una apertura del tablero. Se plantea también un sistema de reconocimiento facial el cual envía la alerta dependiendo si el modelo reconoce al operario que abrió el tablero.

2.1. Contexto

ICOR SAS es una empresa radicada en la ciudad de Córdoba, integrada por ingenieros con experiencia en obras y servicios públicos y privados. Su enfoque es brindar soluciones completas: desde el diseño del proyecto hasta su ejecución, puesta en marcha y operación de sistemas eléctricos, electrónicos y de iluminación.

En el marco de sus actividades, ICOR opera y mantiene tableros de alumbrado público distribuidos geográficamente en rutas y autopistas. Cada tablero contiene la electrónica y protecciones necesarias para alimentar el alumbrado de una zona determinada. La gestión actual de estos tableros se basa en reportes reactivos: ante un reclamo por alumbrado cortado en una ruta, se despacha una cuadrilla técnica al lugar para diagnosticar y reparar la falla.

Un problema recurrente y costoso en este esquema es que la cuadrilla llega al sitio y constata que el corte de alumbrado no es un problema interno del tablero ni del sistema de alumbrado público, sino que se debe a un corte en la alimentación general de la red eléctrica (responsabilidad de la distribuidora). En ese caso, el desplazamiento resultó completamente innecesario y sus costos no son recuperables.

2.2. Motivación

El caso de uso central que motiva este proyecto es el siguiente: se recibe un reporte de que una ruta o autopista tiene el alumbrado público cortado. Ante la ausencia de información remota sobre el estado del sistema, la única respuesta posible es despachar una cuadrilla técnica al lugar. Al llegar, la cuadrilla determina que el

problema no es del tablero de alumbrado sino de un corte en la alimentación eléctrica general, ajeno a su área de responsabilidad.

Este escenario implica costos concretos y evitables:

- Traslado de personal (horas de viaje, horas extras si es fuera de horario).
- Consumo de combustible y desgaste de vehículos.
- Inmovilización de una cuadrilla que podría atender otras tareas.
- Demora en la resolución real del problema, ya que el equipo correcto (la distribuidora) no es notificado con la suficiente anticipación.

Con el sistema propuesto, antes de despachar cualquier cuadrilla, el operador puede consultar el panel y verificar si el tablero de la zona afectada registra tensión de alimentación. Si la tensión está en cero, la causa es externa (corte de red general) y corresponde contactar a la distribuidora, no enviar una cuadrilla de alumbrado. Si la tensión es normal pero hay anomalía de corriente o el contactor está desactivado, entonces sí corresponde la intervención técnica sobre el tablero.

Esta capacidad de diagnóstico remoto previo convierte al sistema en una herramienta de ahorro operativo directo y medible para ICOR SAS.

3. Definición de problema

El proceso operativo actual de ICOR SAS carece de visibilidad remota sobre el estado de sus tableros. Ante un reporte de alumbrado cortado, el único curso de acción disponible es el despacho físico de una cuadrilla, sin poder determinar previamente si el problema es:

A. interno al tablero (falla eléctrica, contactor disparado, vandalismo).
B. externo (corte en la alimentación general de la distribuidora eléctrica).

Esta imposibilidad de diagnóstico remoto previo genera desplazamientos innecesarios con costos de personal, combustible y desgaste vehicular no recuperables.

En términos más amplios, el sistema actual carece de capacidad para:

1. Monitorear en tiempo real las variables eléctricas (tensión, corriente) de cada tablero.

2. Distinguir remotamente si un corte de alumbrado es causa interna o externa al tablero.

3. Detectar y registrar la apertura física de los tableros con evidencia visual (foto).

4. Actuar de forma remota sobre la carga conectada (comando de contactor).

5. Centralizar la información de múltiples tableros distribuidos geográficamente en una plataforma accesible para distintos niveles de usuario.

6. Generar alarmas automáticas ante condiciones fuera de rango o eventos de seguridad.

7. Garantizar el funcionamiento del sistema de monitoreo durante cortes de energía eléctrica.

La ausencia de estas capacidades impacta negativamente en la eficiencia operativa, la seguridad de la infraestructura y la calidad del servicio de alumbrado público.

4. Justificación

El desarrollo de este sistema se justifica desde cuatro perspectivas:

- **Técnica:**

Las tecnologías de IoT (Internet of Things) actuales permiten implementar soluciones de monitoreo remoto con bajo costo de hardware, y plataformas de visualización de datos open source como grafana. La integración de cámaras de bajo costo con detección de eventos agrega valor sin complejidad excesiva. Para la orquestación de estos servicios y la gestión de la lógica de eventos en tiempo real, se integra node-red, herramienta que actúa como el motor de flujos para automatizar la comunicación fluida entre el hardware de campo y los servicios de backend.

Un pilar fundamental que otorga solidez académica a esta propuesta es el enfoque en la seguridad informática, disciplina central de la ingeniería informática. La trascendencia técnica del proyecto radica en asegurar la integridad de los datos desde su origen mediante el protocolo MQTT y el servidor mosquitto, hasta su gestión centralizada. Este diseño no solo resuelve la falta de visibilidad operativa de ICOR SAS, sino que establece un marco de trabajo seguro para la protección de infraestructura crítica.

- **Operativa:**

Un sistema centralizado con alertas automáticas reduce el tiempo de detección y respuesta ante fallas eléctricas, actos de vandalismo y manipulación indebida. La posibilidad de actuar remotamente sobre los contactores evita desplazamientos innecesarios.

- **Organizacional:**

La gestión multiusuario con distintos niveles de acceso (administrador, usuario por tablero, grupos) permite delegar la supervisión de sectores específicos sin comprometer la seguridad global del sistema. El mapa de estado con Google Maps API facilita la supervisión territorial.

La propuesta de solución es algo completamente viable, ya que el hardware a utilizar tiene un precio sustancialmente bajo, y brinda una solución práctica y eficiente a un problema recurrente de la empresa. También facilita la gestión de estos tableros con solo mirar en un dashboard centralizado pudiendo verificar su estado en tiempo real.

- **Perspectiva Social y de Seguridad:**

Desde un punto de vista social, la mejora en la gestión del alumbrado público contribuye directamente a la seguridad ciudadana en rutas y autopistas, garantizando que el servicio se restablezca con mayor celeridad. Asimismo, el registro fotográfico ante aperturas de puertas protege los activos contra el vandalismo y la manipulación no autorizada.

Una vez finalizado el alcance de este proyecto, la empresa podrá realizar el despliegue físico del hardware en sus tableros industriales de alta tensión sin requerir modificaciones estructurales en el sistema de backend, ya que la lógica de procesamiento de datos y el panel de visualización geográfica se encuentran plenamente operativos.

Asimismo, se plantean como futuras líneas de investigación y mejora:

- Implementación de Algoritmos de Mantenimiento Predictivo: Explotar el repositorio histórico de variables de tensión y corriente para el entrenamiento de modelos de aprendizaje automático orientados a la detección temprana de anomalías operativas y la anticipación de fallas en la red de distribución eléctrica

Siguiendo el lineamiento del paper (Pasolini et al., 2019, p. 3), vemos la trascendencia del monitoreo de alumbrado público en ciudades inteligentes, donde se podría profundizar mucho más en el tema, como por ejemplo regulando la intensidad de la iluminación y tendría resultados relevantes en el consumo optimizando mucho más el uso de los recursos.

5. Objetivo general

Diseñar e implementar un prototipo de un sistema IoT de monitoreo para tableros de alumbrado público para ICOR SAS con la finalidad de monitorear el estado en tiempo real de este, facilitando la capacidad de reacción ante inconvenientes habituales de la empresa.

6. Metodología

Para llevar a cabo este proyecto se eligió la metodología scrum, debido a su agilidad y a la familiaridad de cada uno de los integrantes del grupo con ella.

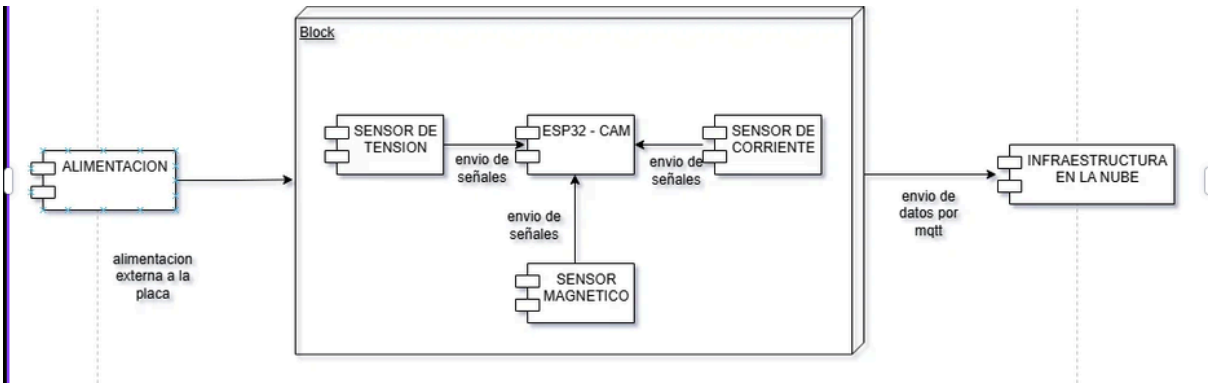
Debido a que no disponíamos de mucho tiempo, decidimos fijar el sprint en 2 días. Apuntando a terminar el proyecto en 14 días (7 sprints) aproximadamente.

Decidimos enfocarnos en cada sprint a una etapa en específico

Etapa	Actividad	Alias	Sprint (2 días)
Requerimientos	Análisis del alcance del proyecto y relevamiento de información	R1	1
Análisis	Refinamiento de requerimientos	A1	2
Diseño	Búsqueda y compra de hardware a utilizar	D1	3
Diseño	Elaboración de diagramas UML y pseudocódigo	D2	4
Implementación	Calibración y conexión de sensores junto con desarollo correcto con tareas e interrupciones	I1	5
Implementación	Deploy e integración de infraestructura del servidor en nube con nodered, grafana, mosquitto e influxdb	I2	6
Prueba	Ejecucion de casos de prueba	P1	7

Adjuntamos algunos de los entregables de cada etapa:

- **Etapa de diseño: Diagrama de bloques**



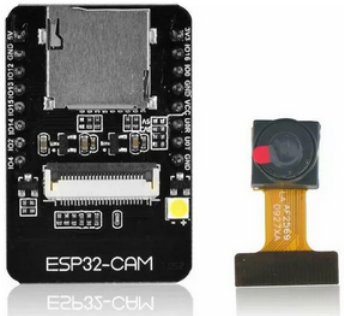
- **Etapa de pruebas: Casos de prueba**

ID	Componente	Escenario de Prueba	Entrada de Datos (MQTT)	Resultado Esperado	Objetivo de la Prueba
CP-01	Filtro de Tensión	Ruido aislado de voltaje (Pico falso)	Un envío de 250.0 V, seguido de valores normales (220.0 V)	El flujo ignora el dato de forma silenciosa. Sin registros ni alertas.	Validar la inmunidad al ruido eléctrico transitorio.
CP-02	Filtro de Tensión	Sobretensión real sostenida	Tres envíos consecutivos de 250.0 V (Intervalo	Inserción en tabla 'alertas_voltaje' y envío de	Verificar la activación ante anomalías reales
CP-03	Módulo Iluminación	Foco quemado / Circuito abierto	Estado del foco: 1 Corriente: 0.0 A	Notificación única de falla en Telegram.	Evaluar la lógica combinacional (Interruptor activo
CP-04	Módulo Iluminación	Fuga de energía / Cortocircuito	Estado del foco: 0 Corriente: 3.5 A	Notificación crítica de fuga en Telegram	Detectar condiciones de riesgo (Corriente
CP-05	Módulo Iluminación	Restauración del circuito	Retorno a parámetros normales desde	Envío de un único mensaje de normalización.	Comprobar el reinicio de las variables de

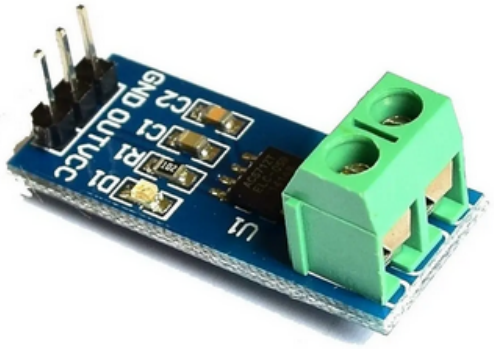
7. Hardware

Para este proyecto se utilizaron los siguientes componentes

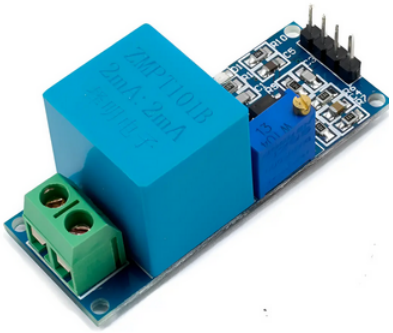
- **Esp32 Cam Camara Modulo Wifi Bt Ov2640 2mp Esp 32 Arduino**



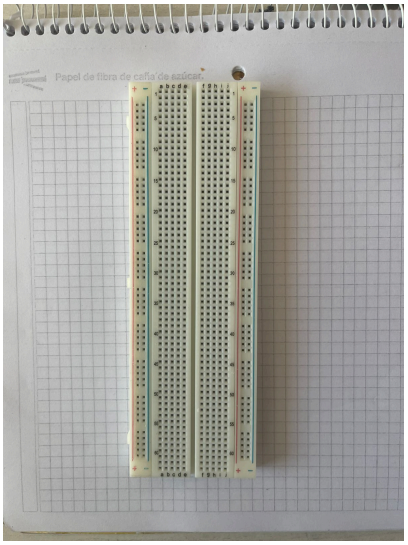
- **Modulo Sensor De Corriente Acs712 ± 30a Efecto Hall Arduino**



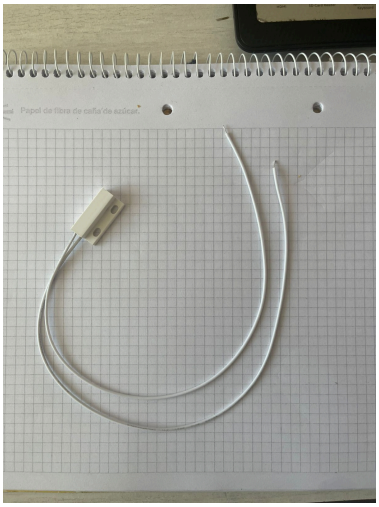
- **Sensor De Voltaje Corriente Alterna Ac 250v Zmpt101b Arduino Zmpt101b-isi**



- **Protoboard**



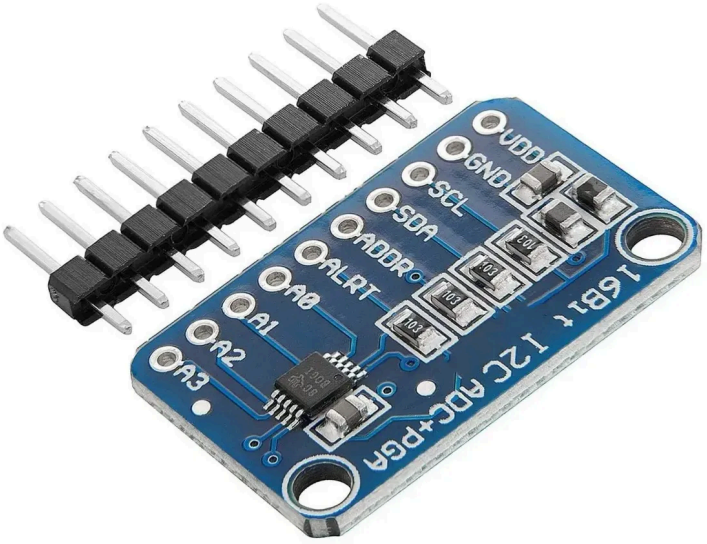
- **Sensor magnético**



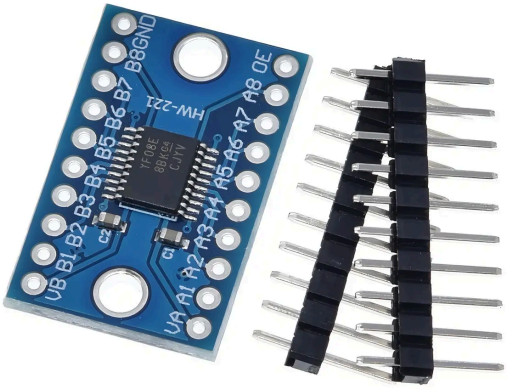
A medida que se fue avanzando con el trabajo se fueron adquiriendo los siguientes componentes:

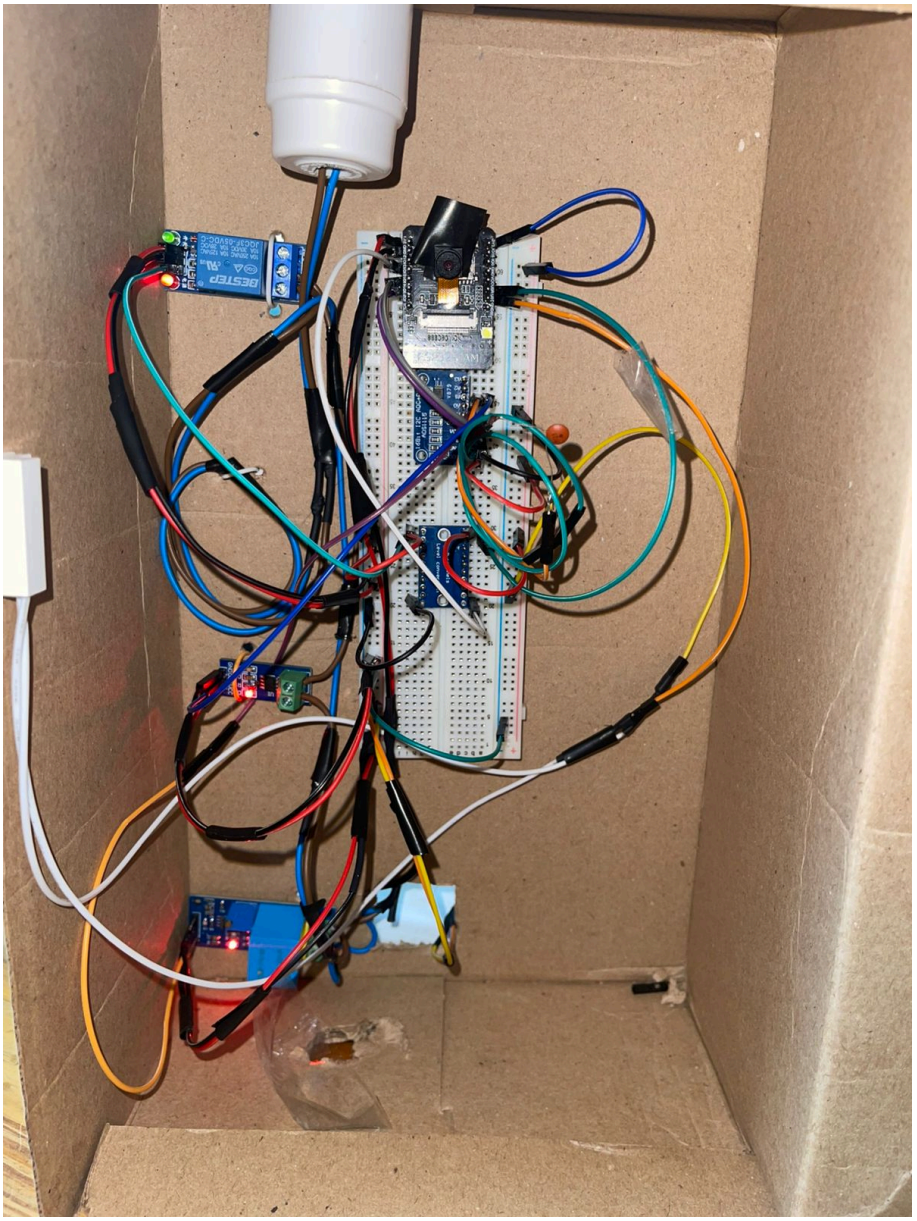
- **Modulo Relay Rele De 4 Canales 5v 10a Compatible con Arduino**

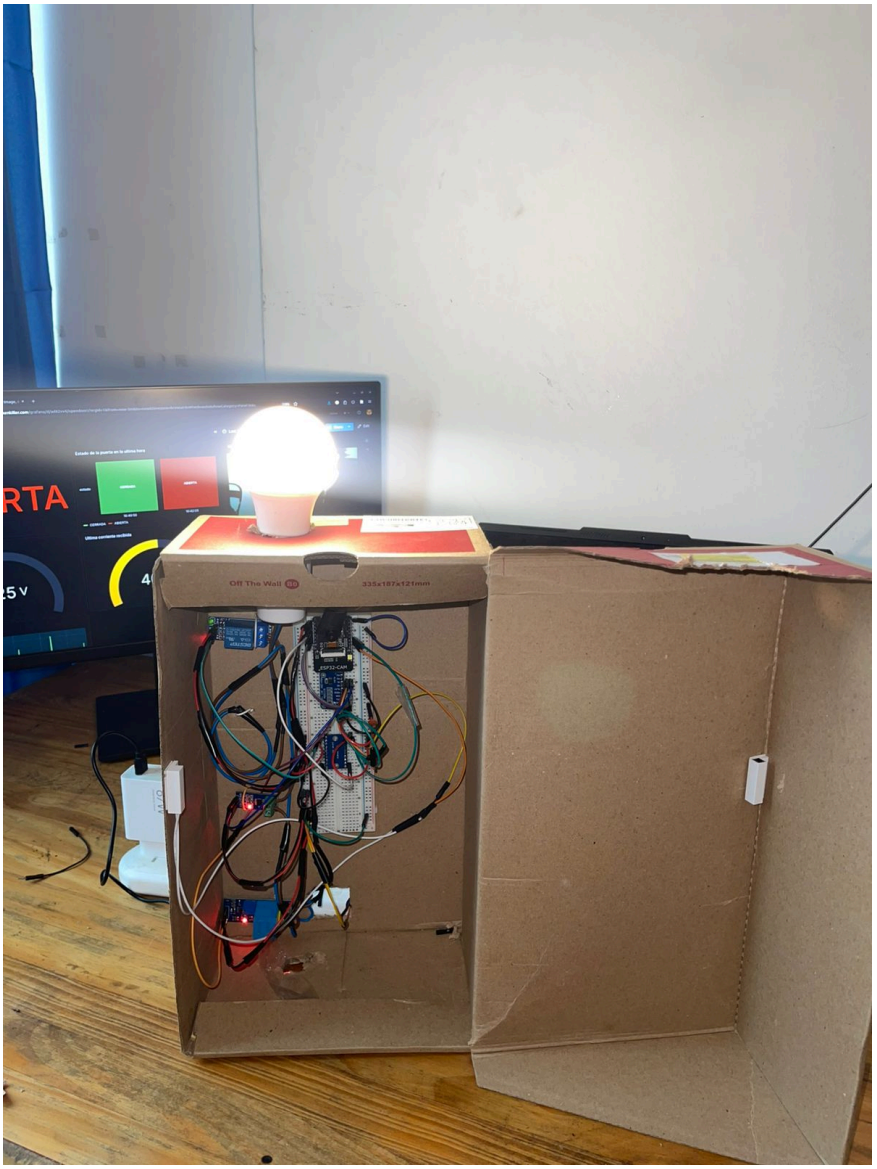
- **Conversor Analogico A Digital Adc Ads1115 16bit I2c Hobb**



- **Conversor De Nivel Arduino Txs0108e Hw221 5v 3 3v 8ch**

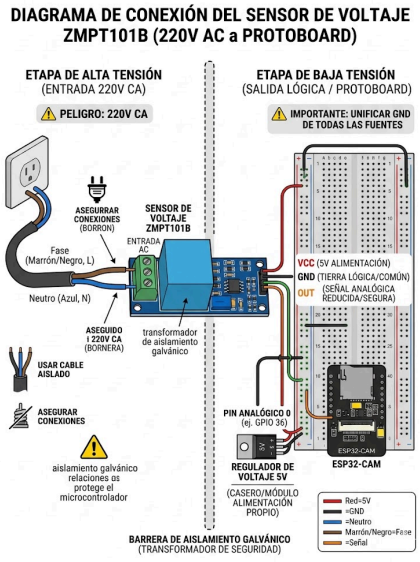




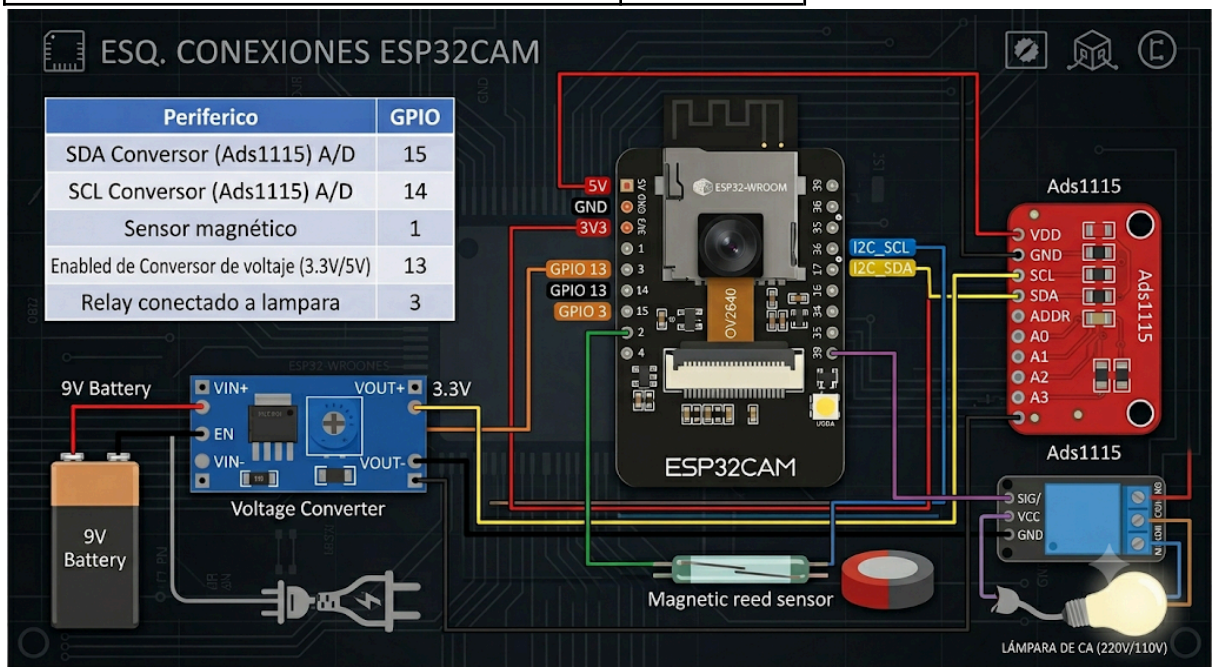


El grafico de abajo fui utilizado también para conectar al sensor de tensión

Se conectaron los siguientes pines a la esp32cam



Periferico	GPIO
SDA Conversor (Ads115) A/D	**15**
SCL Conversor (Ads115) A/D	**14**
Sensor magnético	**1**
Enabled de Conversor de voltaje 3.3V - 5V	**13**
Relay conectado a lampara	**3**



Posteriormente los sensores de tensión y corriente fueron conectados al conversor analógico digital.

8. Software: Código del microcontrolador

Dividimos la parte de programación del software en dos: Programación del código de la esp 32 y Desarrollo de infraestructura del server en cloud

Pasemos a hablar de la primera parte:

8.1. Gestión de Tiempo Real con FreeRTOS

Debido a lo investigado y aprendido en la materia, la opción elegida fue utilizar FreeRTOS y aplicar un RTS soft, ya que no necesitamos que sea estricto en nuestro caso.

Arquitectura General

El sistema ejecuta 6 tareas FreeRTOS más una ISR, todas creadas en **createRtTasks()** (tasks.cpp:141). El planificador de FreeRTOS utiliza scheduling apropiativo (preemptive) con round-robin entre tareas de igual prioridad

Cada tarea se asigna a un núcleo específico mediante **xTaskCreatePinnedToCore(),** aprovechando la arquitectura dual-core del ESP32. El Core 0 (pro-core) ejecuta las tareas críticas de sensado y captura; el Core 1 (app-core) ejecuta las tareas de comunicación red y logging.

Tareas (Hilos)

Tarea	Prioridad	Core	Stack (bytes)	Comportamiento
`SensorTask`	5 (máxima)	0	2048	Bloqueada en un semáforo binario. La ISR la despierta al detectar un flanco en el sensor magnético (`CHANGE`). Aplica debounce de 200ms, filtra cambios redundantes, y encola eventos en `sensorEventQueue` y `mqttEventQueue`.
`CaptureTask`	4	0	4096	Bloqueada en `sensorEventQueue`. Al recibir un evento HIGH: enciende LED 150ms, captura foto (polling 20×100ms = 2s timeout), envía el buffer a `captureQueue`. En LOW: solo loguea.
`TelnetTask`	3	1	4096	Polling cada 10ms (`vTaskDelay(10)`). Procesa `telnet.loop()` y comandos entrantes. No es crítica en tiempo real.
`HttpTask`	2	1	8192	Bloqueada en `captureQueue`. Al recibir un frame, construye multipart/form-data y hace HTTP POST con 3 reintentos, 1s entre reintentos, 15s de timeout. Usa `WiFiClientSecure::setInsecure()`.
`MqttTask`	1 (mínima)	1	4096	Bucle permanente: mantiene conexión MQTT (PubSubClient), recibe de `mqttEventQueue` con timeout de 100ms, publica `ABIERTA`/`CERRADA` en `estado-puerta` (QoS 0). También publica tensión y corriente desde `adsDataQueue`.
`AdsTask`	1	1	4096	Cada 600ms lee el ADC ADS1115 por I2C (cálculo RMS en ventana de 20ms), filtra con EMA, envía a `adsDataQueue`.

- **Apropiativo por prioridad:** una tarea de mayor prioridad (ej. SensorTask, prio 5) desaloja inmediatamente a una de menor prioridad (ej. MqttTask, prio 1) en cuanto puede ejecutarse.

- **Round-robin entre igual prioridad:** si varias tareas comparten prioridad, el tick las turna en tiempo compartido. En este proyecto no hay dos tareas con la misma prioridad, por lo que el round-robin no se aplica.

- **Tareas bloqueadas:** la mayoría de las tareas pasan la mayor parte del tiempo bloqueadas en colas (xQueueReceive con portMAX_DELAY) o semáforos (xSemaphoreTake), lo que permite que el idle task consuma CPU o que el ESP32 entre en modo de bajo consumo.

- No se utiliza vTaskSuspend / vTaskResume ni notificaciones directas; toda la sincronización se basa en primitivas de FreeRTOS (colas y semáforos).

Tipo de Sistema en Tiempo Real: Suave (Soft Real-Time)

El sistema es de tiempo real suave (soft real-time) por las siguientes razones:

- La violación ocasional de un deadline no causa fallo catastrófico. Si la captura falla (timeout 2s), solo se pierde una foto. Si el HTTP POST falla tras 3 reintentos, se descarta el frame.

- No hay mecanismos de temporización explícitos (xTimer, xTaskDelayUntil) para garantizar periodicidad estricta.

- El sistema tolera la pérdida de eventos: si sensorEventQueue está llena (10 elementos), el nuevo evento se descarta y se loguea una advertencia.

- No es un sistema hard real-time porque no existen deadlines de los que dependa la seguridad o integridad del sistema.

Sincronía vs Asincronía: Event-Driven Asíncrono

El sistema es completamente asíncrono y manejado por eventos (event-driven):

- ISR → SensorTask: el sensor magnético genera una interrupción por flanco (CHANGE). La ISR libera un semáforo binario (sensorSemaphore). Este es el único punto de entrada de eventos externos.
- SensorTask → CaptureTask: los eventos se propagan mediante sensorEventQueue (cola de mensajes con sensor_event_t{state, timestamp_ms}).
- SensorTask → MqttTask: paralelamente, el mismo evento se envía a mqttEventQueue.
- CaptureTask → HttpTask: los punteros a camera_fb_t se envían mediante captureQueue.
- No existe un loop central sincrónico ni polling de sensores (salvo TelnetTask y AdsTask, que son las únicas tareas basadas en temporización periódica).

### Task	Tipo
`SensorTask`	Event-driven (semáforo desde ISR)
`CaptureTask`	Event-driven (cola de mensajes)
`HttpTask`	Event-driven (cola de mensajes)
`MqttTask`	Híbrido: event-driven (cola) + polling MQTT (loop + delay 100ms)
`TelnetTask`	Time-driven (polling cada 10ms)
`AdsTask`	Time-driven (delay fijo de 600ms)

Primitivas de Sincronización

Primitiva	Tipo	Propósito	Creada en
`sensorSemaphore`	Semáforo binario	Despertar a `SensorTask` desde la ISR. Modelo productor-consumer (ISR → tarea).	`tasks.cpp:142`
`sensorEventQueue`	Cola (10 × `sensor_event_t`)	Enviar eventos de sensor desde `SensorTask` → `CaptureTask`.	`tasks.cpp:143`
`captureQueue`	Cola (10 × `camera_fb_t*`)	Transferir buffers de cámara desde `CaptureTask` → `HttpTask`.	`tasks.cpp:144`
`mqttEventQueue`	Cola (5 × `sensor_event_t`)	Enviar eventos desde `SensorTask` → `MqttTask`.	`tasks.cpp:145`
`adsDataQueue`	Cola (5 × `ads_data_t`)	Enviar lecturas del ADS1115 desde `AdsTask` → `MqttTask`.	`tasks.cpp:146`

Patrón de uso de colas

- xQueueSend con timeout finito (100–1000ms) en lugar de portMAX_DELAY para evitar deadlocks si la cola destino está llena. tasks.cpp:40,43,68.
- xQueueReceive con portMAX_DELAY en las tareas consumer (CaptureTask, HttpTask) para bloquear indefinidamente hasta que lleguen datos. tasks.cpp:52,90.
- mqttEventQueue usa xQueueReceive con timeout de 100ms en MqttTask, combinando recepción de eventos con mantenimiento de conexión MQTT. mqtt.cpp:65.

```
ISR (Interrupt Service Routine)
static void IRAM_ATTR sensor_isr() {
    BaseType_t higherTaskWoken = pdFALSE;
    xSemaphoreGiveFromISR(sensorSemaphore, &higherTaskWoken);
    if (higherTaskWoken) {
        portYIELD_FROM_ISR();
    }
}
```

- Adjunta al pin del sensor magnético en flanco CHANGE mediante attachInterrupt() (tasks.cpp:148).
- No usa gpio_install_isr_service() de ESP-IDF para evitar conflicto con attachInterrupt de Arduino y el driver de cámara.

- La ISR es mínima: solo libera un semáforo. Todo el procesamiento (debounce, lectura de pin, encolado) se delega a SensorTask.

- Incluye portYIELD_FROM_ISR() condicional para que el planificador ejecute inmediatamente SensorTask si tiene mayor prioridad que la tarea interrumpida.

Medición de Corriente y Tensión (ADS1115)

Hardware

- Se utiliza un ADS1115 (ADC de 16 bits, I2C) conectado en 0x48 por I2C (pines 15-SDA, 14-SCL, clock 400kHz). Mide dos canales:

Canal	Señal	Acondicionamiento
A0	Tensión de red	Divisor resistivo, factor = 513.69
A1	Corriente	Sensor de efecto Hall (ACS712 o similar), factor = 0.0916

Cálculo RMS (ventana deslizante de 20ms)

La función calcular_RMS() en ads.cpp:25-48 implementa un algoritmo RMS de dos pasadas sobre una ventana fija de 20ms para capturar exactamente un ciclo de 50Hz:

Primera pasada — recorre ~20ms leyendo muestras a 860 SPS (la tasa máxima del ADS1115), acumulando valores de tensión para calcular el zeroPoint (offset DC promedio).

Segunda pasada — recorre otros ~20ms restando el offset DC a cada muestra, elevando al cuadrado, sumando, y finalmente calculando sqrt(sum_sq / count).

```
float zeroPoint = Vsum / conteo;
```

```
// ...
double sum_sq = 0.0;
while (micros() - t0 < 20000) {
    float v = ads.computeVolts(ads.readADC_SingleEnded(canal)) -
zeroPoint;
    sum_sq += (double)(v * v);
    conteo++;
}
return sqrtf((float)(sum_sq / conteo));
```

Filtrado EMA (Exponential Moving Average)

En adsTask() (ads.cpp:50-73), cada 600ms se aplica un filtro EMA suave para eliminar ruido transitorio:

```
voltaje_filt   = 0.6f * voltaje_filt   + 0.4f * tension_instantanea;
corriente_filt = 0.6f * corriente_filt + 0.4f * corriente_instantanea;
if (corriente_filt < 0.015f) corriente_filt = 0.0f;  // umbral de
histéresis
```

Los resultados se publican por MQTT en los tópicos tension y corriente con 1 decimal para tensión y 3 para corriente.

Consideraciones sobre el Tiempo Real

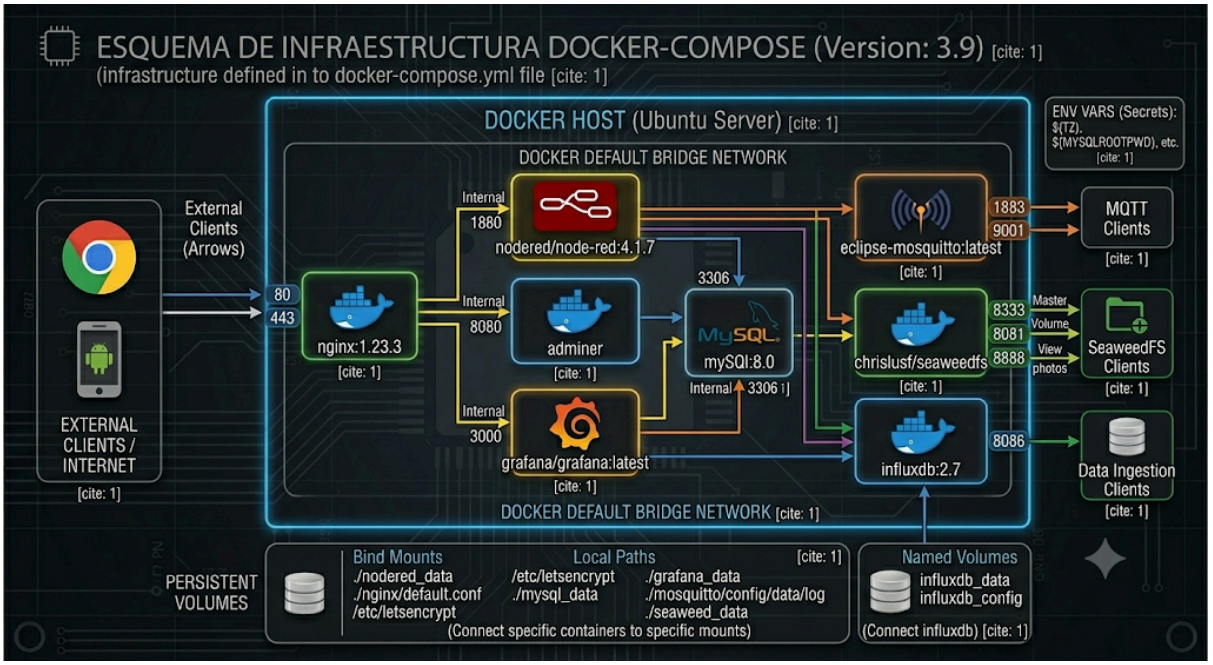
- Latencia ISR → SensorTask: mínima: la ISR libera el semáforo y hace portYIELD. Como SensorTask tiene la máxima prioridad (5) en el Core 0, se ejecuta inmediatamente después de la ISR.
- Debounce de 200ms: evita falsos positivos por rebote mecánico del sensor. Esto establece un límite inferior en la separación mínima entre eventos detectables (~200ms).

- Timeout de captura: esp_camera_fb_get() se invoca con polling de 20 iteraciones × 100ms = 2s. Si no hay fotograma disponible (ej. la cámara ya está siendo usada por el stream HTTP), se descarta el evento.

- Colas acotadas: captureQueue (10) y sensorEventQueue (10) definen el backlog máximo. Si el servidor HTTP está caído, se pueden acumular hasta 10 frames sin pérdida.

9. Software: Infraestructura del server de la nube

Toda esta información es enviada al servidor, que es el encargado luego de graficar y realizar alertas personalizadas con estos datos.
En el siguiente grafico podemos ver un poco de como es la infraestructura

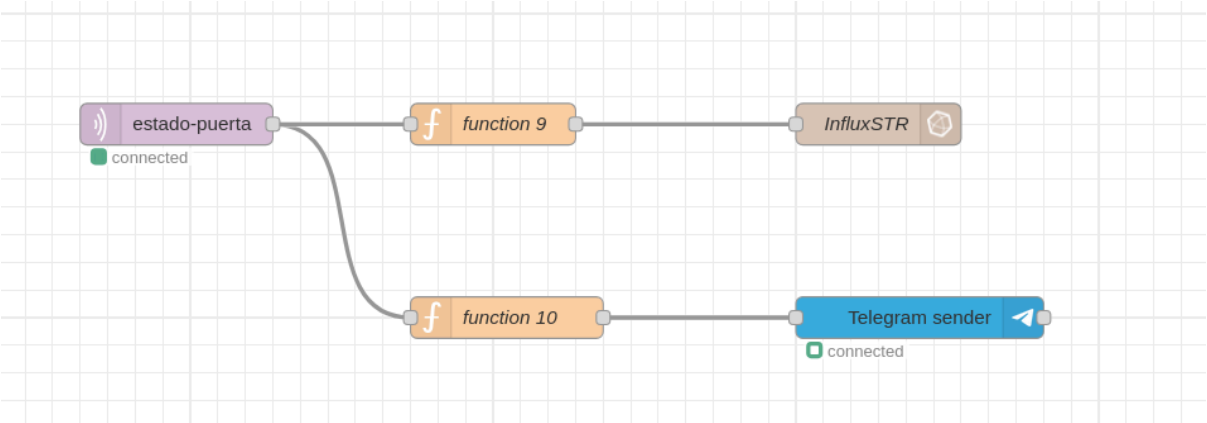


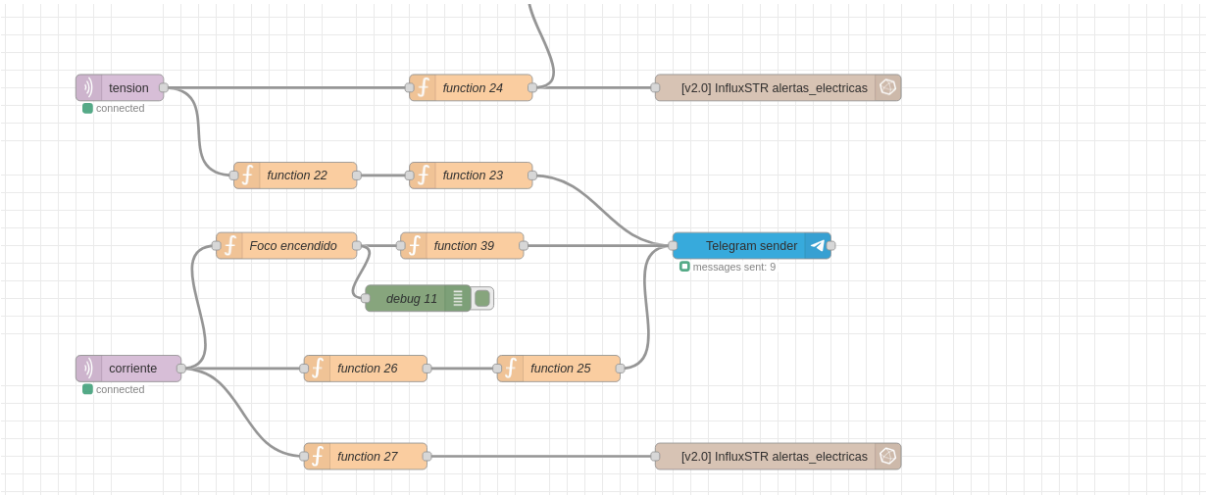
- Se utiliza **mqtt** para el envio constante del sensado de corriente y tension, asi como el ultimo estado de la puerta.
- Se utiliza **http** para el envio de la foto cuando se detecta una apertura

Envío de datos mediante mqtt

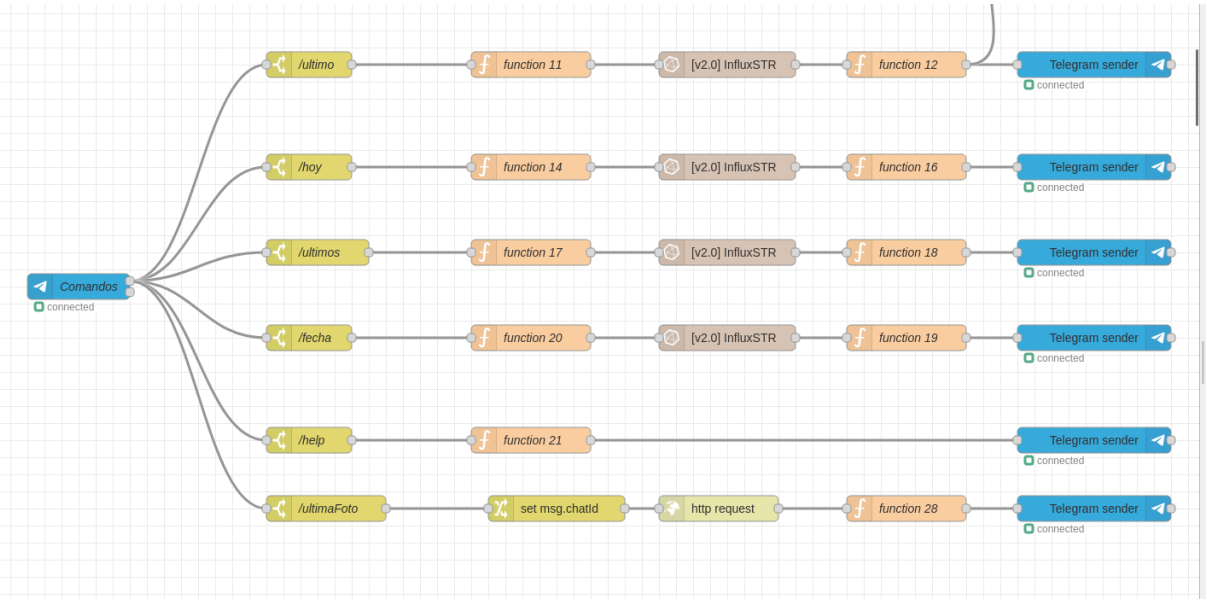
Los datos mqtt son recibidos por un server mosquitto de la vm, el cual mediante un flujo de node red se encarga de guardarlos en influxdb para ser persistentes en el tiempo.
A su vez tambien este flujo está configurado para enviar alertas a telegram cuando se detecta que sobrepasa ciertos umbrales

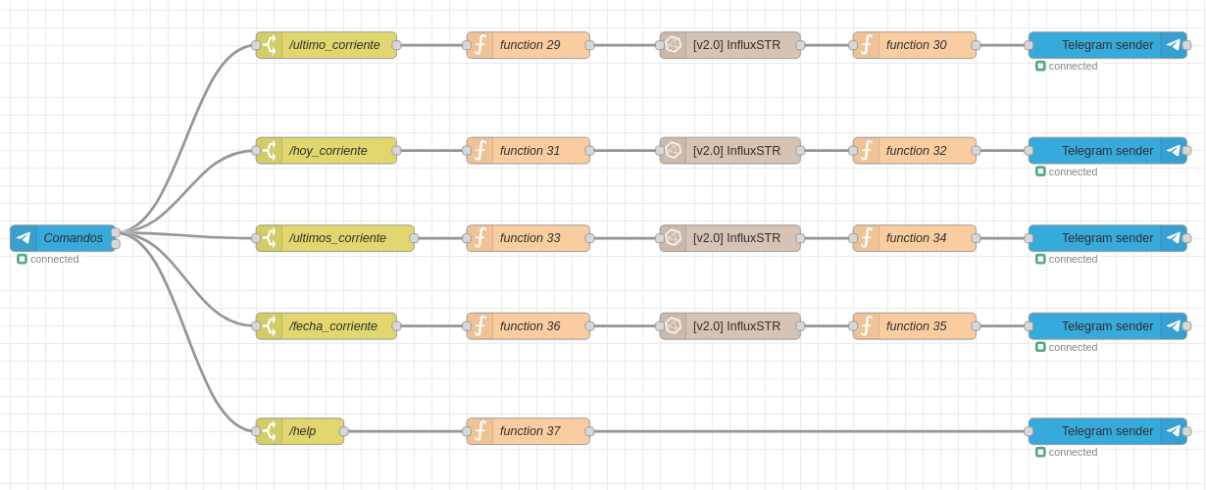
Nodered





Logramos a configurar los siguientes comandos en telegram





Para hacer pruebas consideramos configurar tambien la conexión a un foco y poder encenderlo mediante mqtt para ver la fluctuación de corriente. Para esto se ejecuta el siguiente comando
mosquitto_pub -h cernikiw3.chickenkiller.com -p 1883 -t foco/control -m 1

Y para apagar el foco:

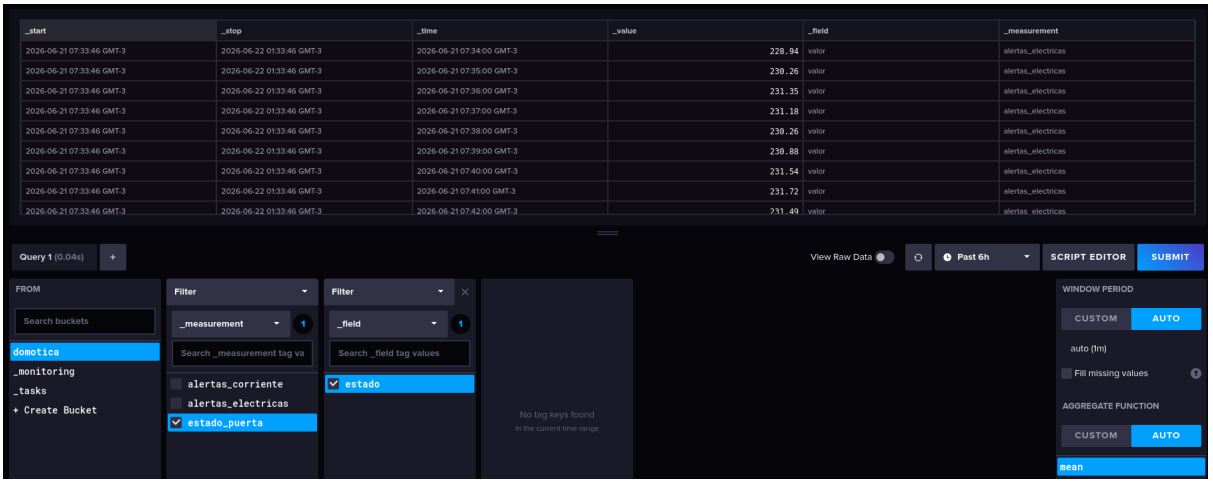
mosquitto_pub -h cernikiw3.chickenkiller.com -p 1883 -t foco/control -m 0

Influxdb

Elegimos la siguiente estructura de base de datos con influxdb
Sensado de corriente:

Sensado de tension

Apertura de puertas



Grafana

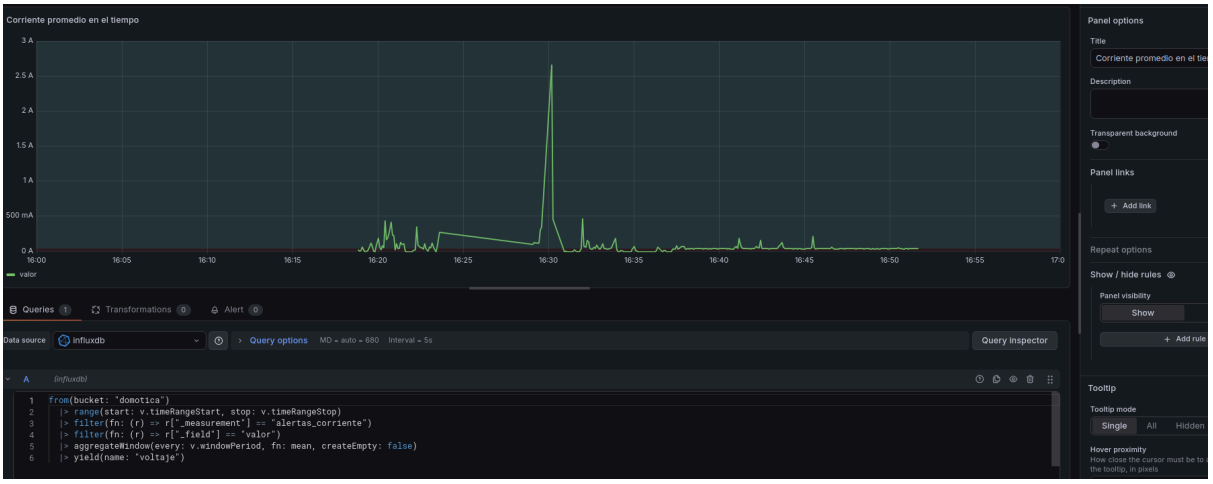
Todos estos datos son desplegados luego en grafana

- **Ultimo voltaje**

- **Voltaje promedio en el tiempo**

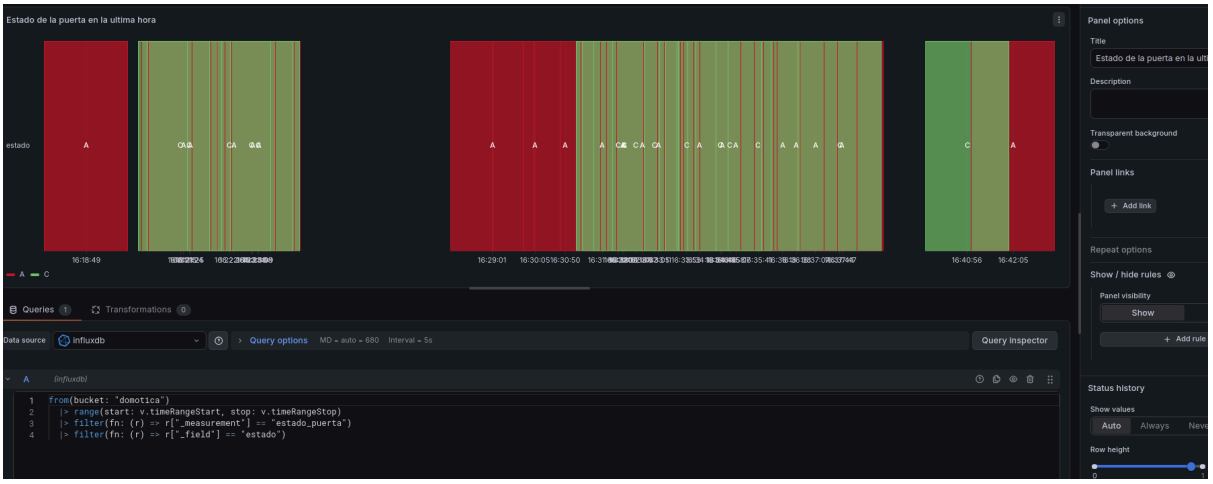
- **Ultima corriente**

- **Corriente promedio**

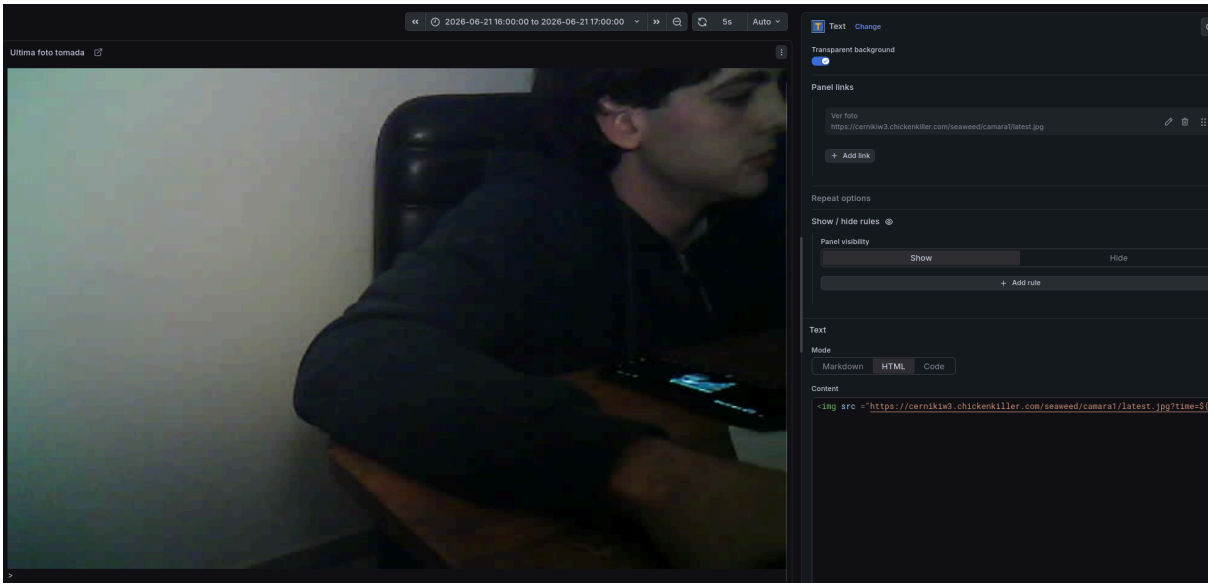


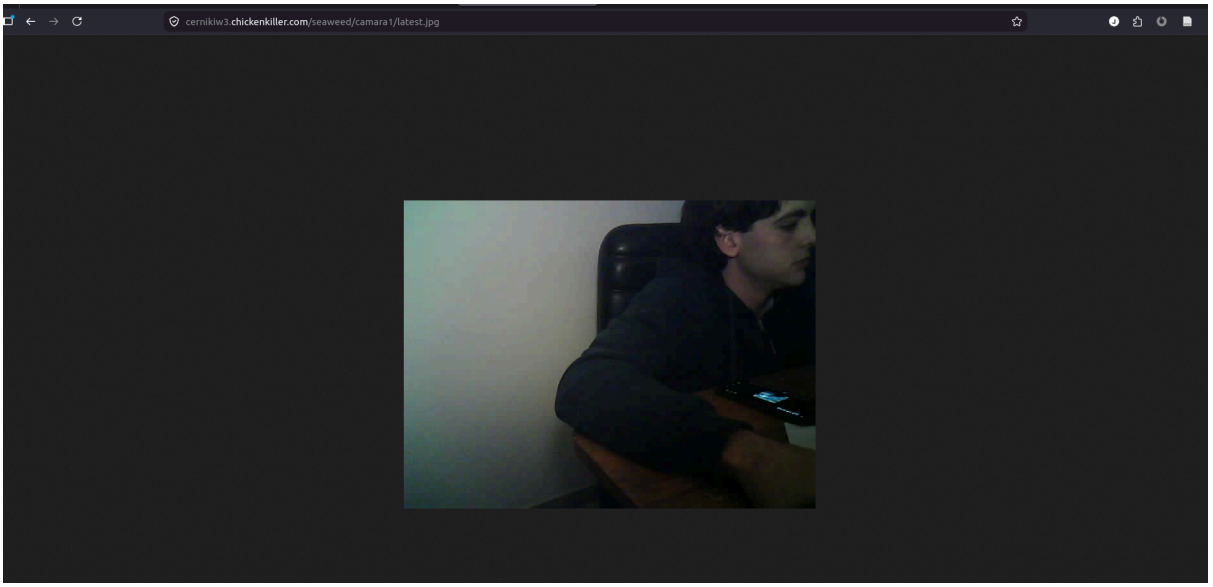
- **Estado actual de puerta**

- **Estado de puerta en la ultima hora**



Para mostrar la ultima foto tomada se hace un get a la utl del seaweed, que es la base de datos donde se persisten las imagenes tomadas desde la esp





Vista general

10. Conclusión

Fue un excelente trabajo donde nos pudimos curtir con todo el mundo electrónico y nos gusto mucho ya que no hemos tenido experiencias anteriores de este estilo en la carrera. Fue un gran desafío ya que fuimos aprendiendo en el camino y revirtiendo errores causados por la inexperiencia. En particular, trabajar con una placa tan limitada pero potente como una esp32cam saco lo mejor de nosotros y estamos muy conformes con el resultado. También creemos que fue un broche de oro que termina de conectar tanto lo aprendido en el teórico y en el práctico de la materia de sistemas en tiempo real, ya que asociamos sensores, I2C, tasks, colas, interrupciones, prioridades, asincronía, y demas, todo en un mismo trabajo

Se adjuntan los repositorios

https://github.com/joaquincernik/str-final

https://github.com/joaquincernik/str-final-infraestructura