**MONITOR ENERGÉTICO**

**INFORME DE DESARROLLO E IMPLEMENTACIÓN**

Descripción del problema.

La compañía Control Academic se encuentra en constante desarrollo y expansión, debido al aumento en la producción y al incremento de uso de recursos económicos para la empresa, este proyecto se hace con la finalidad de cuantificar el consumo energético para la compañía.

Se conoce que no se permite obtener información sobre el monitoreo de potencia, consumo energético, voltaje, corriente, factor de potencia, ni registro del sistema. Como consecuencia de esta situación, la producción de energía que abastece a la empresa es una incógnita sin un sustento cuantificable.

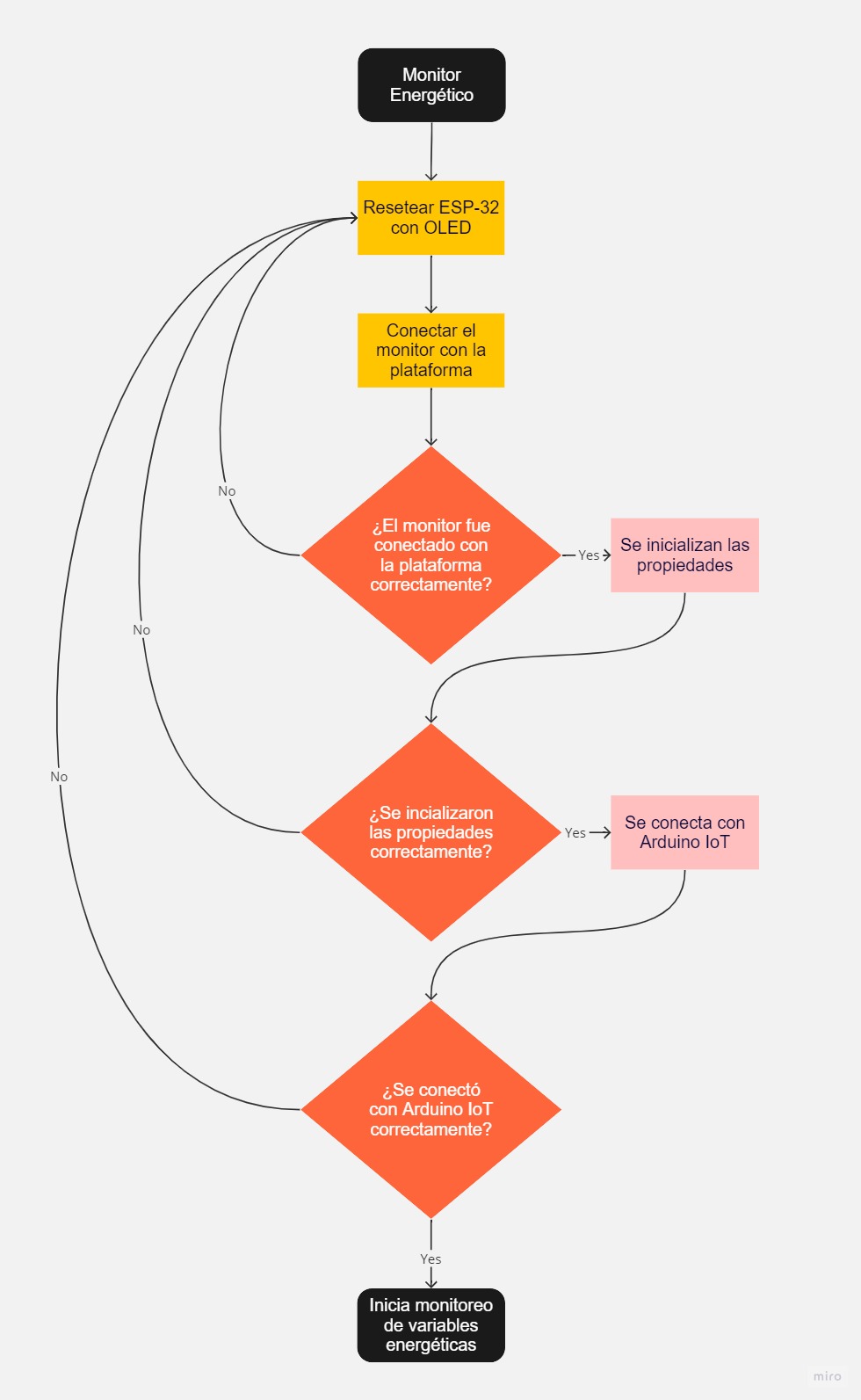
Básicamente el problema principal es que no existe forma de monitorear ni determinar el gasto que representa el consumo energético para la planta.

Pregunta principal

¿De qué manera desarrollar e implementar un sistema de monitoreo energético para la empresa?

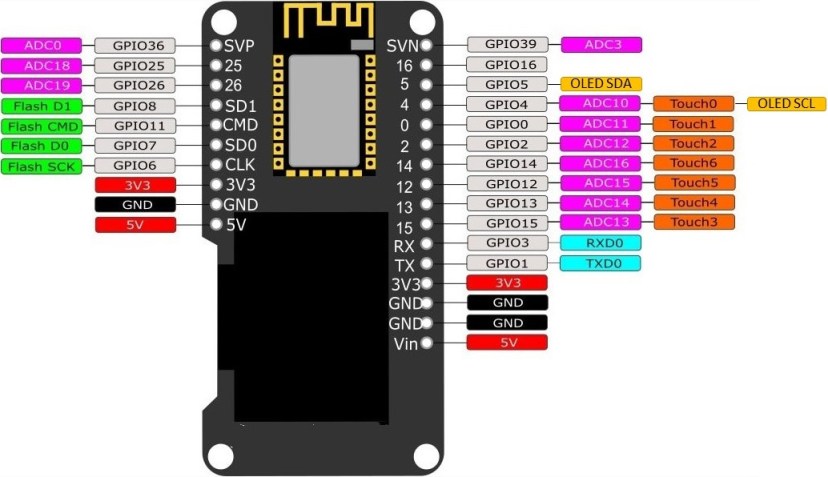
Objetivo general: Diseñar e implementar un sistema de monitoreo energético para la planta energética de la empresa Control Acdemic.

**Diagrama de flujo del funcionamiento principal del monitor energético**



**Fase I**. Descripción de los materiales utilizados

1. ESP-WROOM-32 con librería WEMOS LOLIN32



Puertos utilizados para realizar las conexiones al circuito impreso (Fig. 1 y Fig. 2)

1. Fuente de voltaje de 5V de Hi-Link HLK-PM01



+Vo (Conectado a fuente de 5V)

-Vo (Conectado a GND común del circuito)

Conexión a la línea de AC

Fase – Blanco

Neutro – Rojo

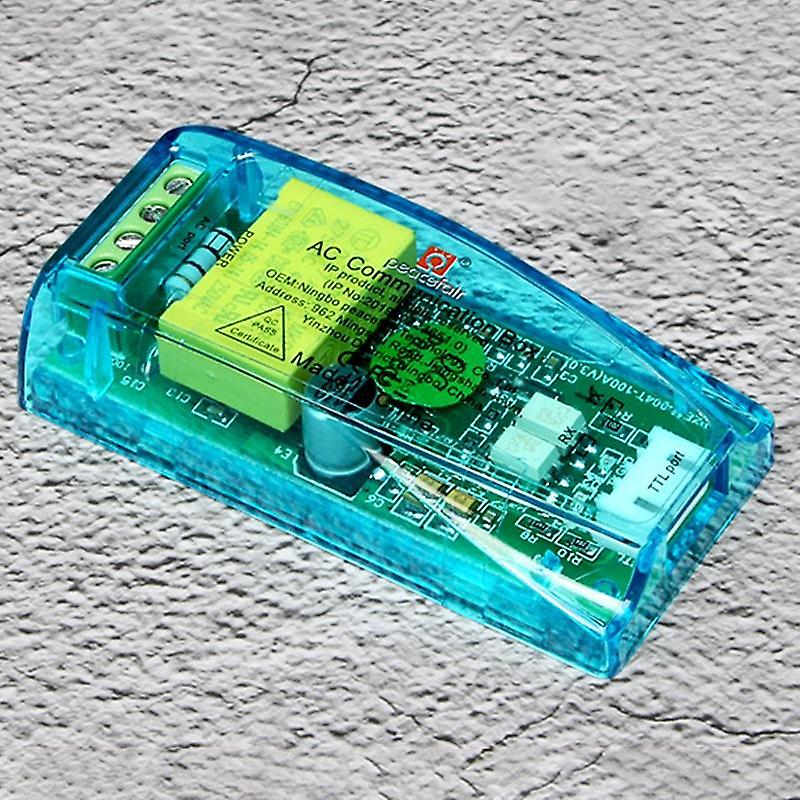
(Fig. 1 y Fig. 2)

Conexiones al circuito impreso (Fig. 1 y Fig. 2)

1. AC Communication Box de la empresa peacefair

Módulo de comunicación de CC PZEM-003/017, el módulo se utiliza principalmente para medir voltaje de CC, corriente, potencia activa y consumo de energía, el módulo no tiene función de visualización, los datos se leen a través de la interfaz UART.

* PZEM-003: Measuring Range 10A (Built-in Shunt)
* PZEM-017: Measuring Range 50A、100A、200A、300A (the current range is dependent on the external shunt specification)



Conexiones al circuito impreso (Fig. 1 y Fig. 2)

N F (sensor corriente)

Puerto AC

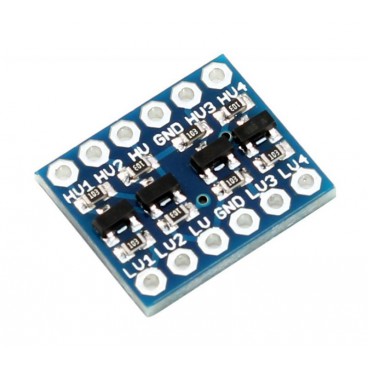
Puerto TTL

1. Sensor de corriente PZCT – 02



Conexiones al circuito impreso (Fig. 1 y Fig. 2)

1. Convertidor de voltaje bidireccional de 4 bits



Conexiones al circuito impreso (Fig. 1 y Fig. 2)

3.3V GND

RX TX 5V GND

1. Materiales no electrónicos

Impresión 3D con programa AutoDesk Inventor Professional

* Circuito impreso
* Caja de material AMG
* Cableado de diferentes calibres
* Bornes
* Equipo de soldadura (estaño)

**Fase II**. Programación en interfaz de Arduino para el correcto funcionamiento del monitor energético. Código realizado con Arudino IoT Cloud y uso de la herramienta de Dashboard.

Primeramente, se incluyen las librerías a utilizar, así como definir los pines 25 y 26 para el pin de recepción RX y transmisión TX respectivamente, eso únicamente si no están definidos en primera instancia.

#include "arduino\_secrets.h"

#include "thingProperties.h"

#include <PZEM004Tv30.h>

#include <Wire.h>

#include "SSD1306Wire.h"

#if !defined(PZEM\_RX\_PIN) && !defined(PZEM\_TX\_PIN)

#define PZEM\_RX\_PIN 25

#define PZEM\_TX\_PIN 26

#endif

Después se declaran los pines a utilizar para el ESP-WROOM-32 y para el OLED que viene incorporado al mismo.

PZEM004Tv30 pzem(Serial1, 25, 26);

SSD1306Wire display(0x3c, 5, 4);  // ADDRESS, SDA, SCL

Después, dentro del void setup se inicializa el Serial, se pone la instrucción de visualizar en el display OLED la secuencia de procedimiento que descarta posibilidades de errores y se realiza conexión con Arduino IoT Cloud.

void setup() {

  Serial.begin(9600);

  delay(1500);

  display.init();

  display.flipScreenVertically();

  display.setFont(ArialMT\_Plain\_10);

  display.clear();

  display.setTextAlignment(TEXT\_ALIGN\_LEFT);

  display.setFont(ArialMT\_Plain\_10);

  display.drawString(0, 20, "Conectando con plataforma");

  display.display();

  delay(1500);

  initProperties();

  display.clear();

  display.setTextAlignment(TEXT\_ALIGN\_LEFT);

  display.setFont(ArialMT\_Plain\_10);

  display.drawString(0, 20, "Inicializando propiedades");

  display.display();

  delay(1500);

  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  display.clear();

  display.setTextAlignment(TEXT\_ALIGN\_LEFT);

  display.setFont(ArialMT\_Plain\_10);

  display.drawString(0, 20, "Conectando con ");

  display.drawString(0, 30, "Arduino IoT Cloud");

  display.display();

  delay(1500);

  setDebugMessageLevel(2);

  ArduinoCloud.printDebugInfo();

}

Dentro del void loop se encontrará la impresión de valores energéticos en el Serial Monitor, también se realiza una actualización en la nube de Arduino.

void loop() {

  ArduinoCloud.update();

  voltaje = pzem.voltage();

  Serial.print("Voltage: ");

  Serial.print(voltaje);

  Serial.println("V");

  corriente = pzem.current();

  Serial.print("Current: ");

  Serial.print(corriente);

  Serial.println("A");

  potencia = pzem.power();

  Serial.print("Power: ");

  Serial.print(potencia);

  Serial.println("W");

  consumoAcumulado = pzem.energy();

  Serial.print("Consumo (kWh): ");

  Serial.print(consumoAcumulado, 3);

  Serial.println("kWh");

  factorPotencia = pzem.pf();

  Serial.print("PF: ");

  Serial.println(factorPotencia);

  display.clear();

  drawFontFaceDemo(voltaje, corriente, potencia, consumoAcumulado, factorPotencia);

  display.display();

  delay(1000);

}

Para mostrar en el display OLED que se encuentra en conjunto con el microcontrolador ESP-WROOM-32, se utiliza la función void drawFontFaceDemo que se manda a llamar dentro del void loop.

void drawFontFaceDemo(float voltaje, float corriente, float potencia, float consumoAcumulado, float factorPotencia) {

  display.setTextAlignment(TEXT\_ALIGN\_LEFT);

  display.setFont(ArialMT\_Plain\_10);

  display.drawString(0, 0, "Potencia (W)");

  display.drawString(80, 0, String(potencia));

  Serial.println(String(potencia));

  display.drawString(0, 12, "Consumo (kWh)");

  display.drawString(80, 12, String(consumoAcumulado));

  Serial.println(String(consumoAcumulado));

  display.drawString(0, 24, "Voltaje (V)");

  display.drawString(80, 24, String(voltaje));

  display.drawString(0, 36, "Corriente (A)");

  display.drawString(80, 36, String(corriente));

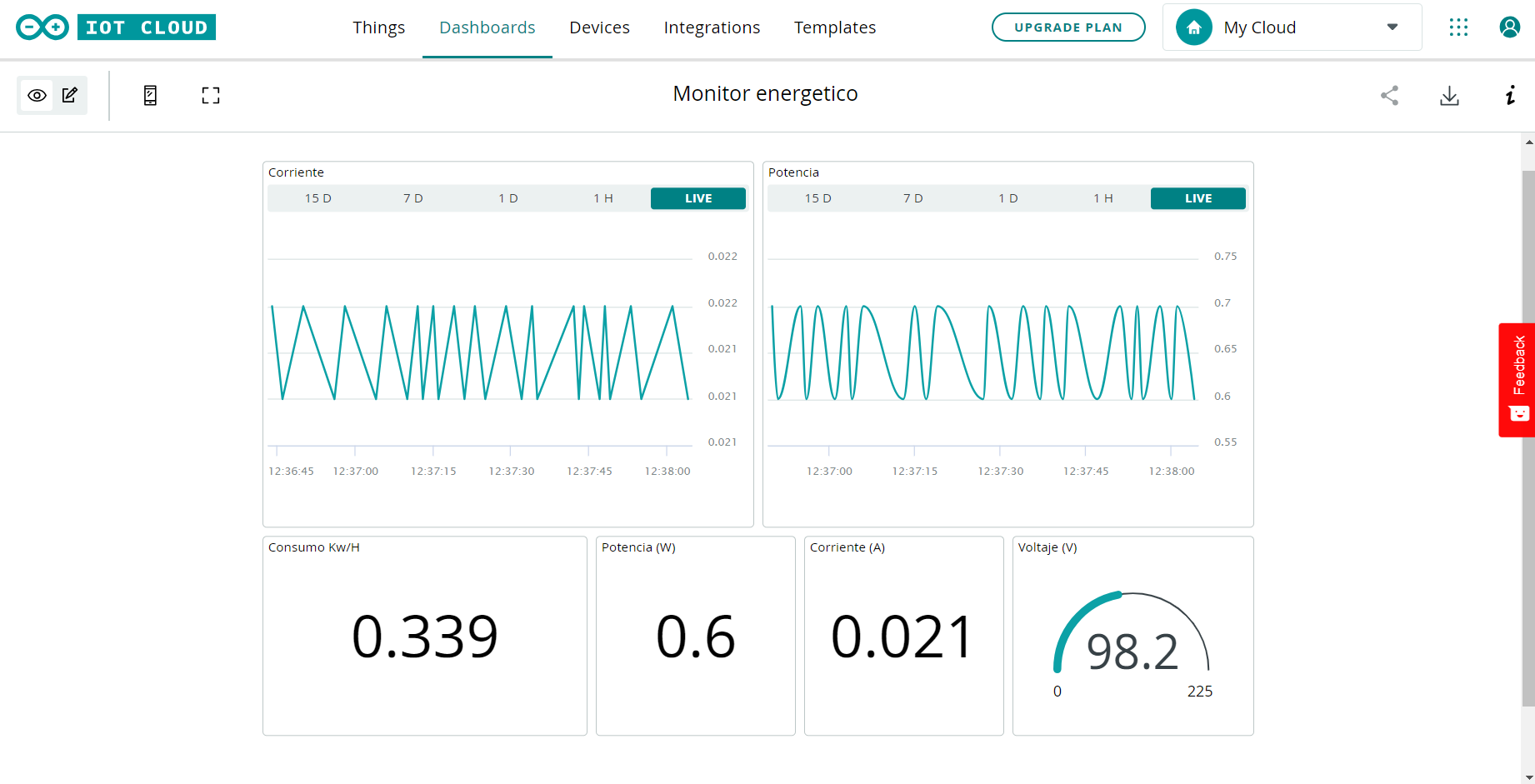
  display.drawString(0, 48, "Factor PF");

  display.drawString(80, 48, String(factorPotencia));

}

Cabe resaltar que, también hay archivos con extensión adoc, .h y .json incluidos dentro del proyecto aparte del archivo .ino del que se explicaron los bloques más importantes anteriormente, estos son de mucha utilidad para el correcto funcionamiento del monitor energético.

Pasando al dashboard de Arduino IoT Cloud, se colocaron widgets correspondientes a cada una de las variables energéticas del monitor. A continuación, se mostrará como se perciben los resultados que arroja dicha plataforma.



Vista del dashboard del monitor energético para PC.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Vista del dashboard del monitor energético para celular.

**Monitor energético final montado**

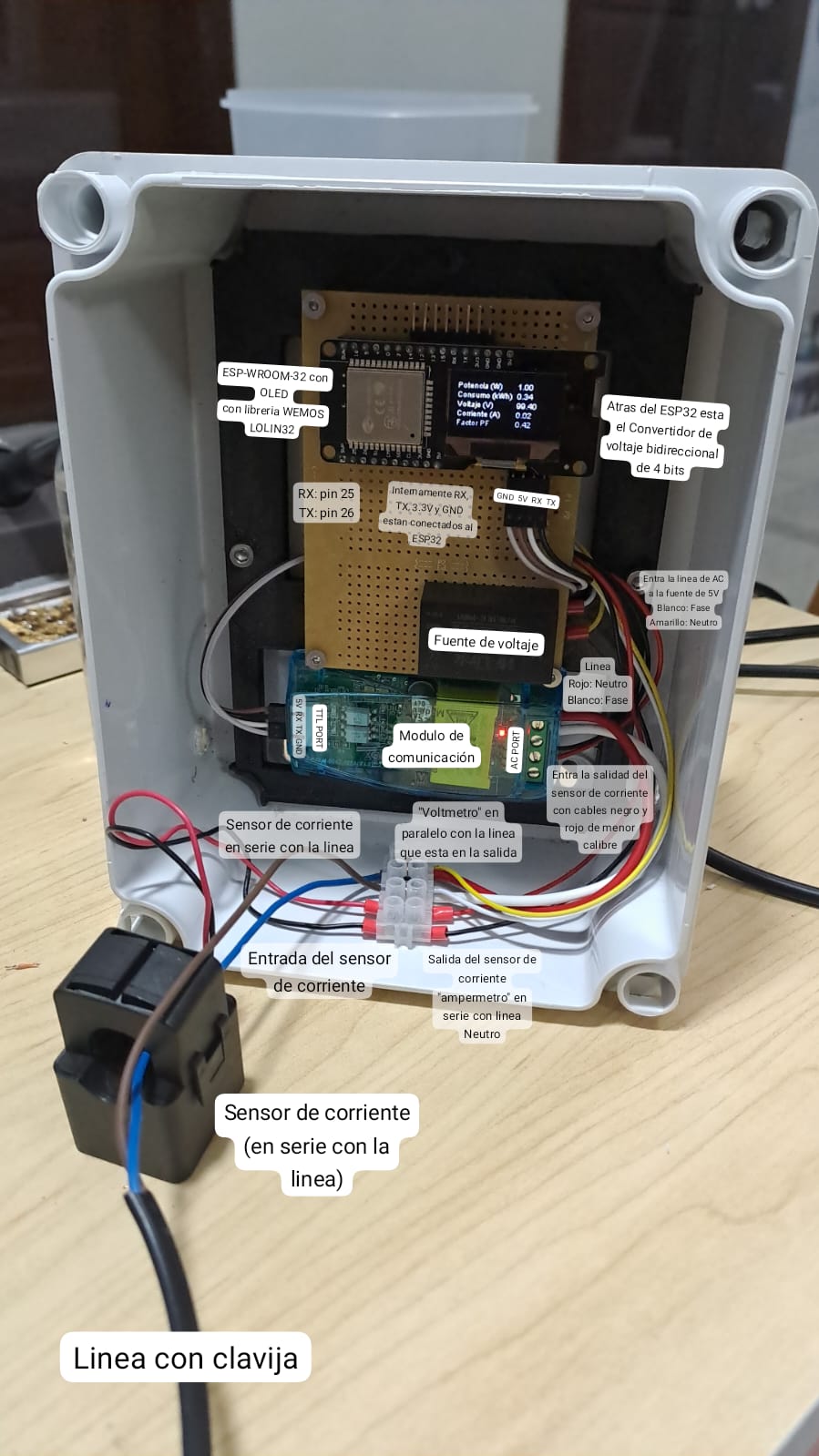


Figura. 1

**Interconexiones del circuito impreso**

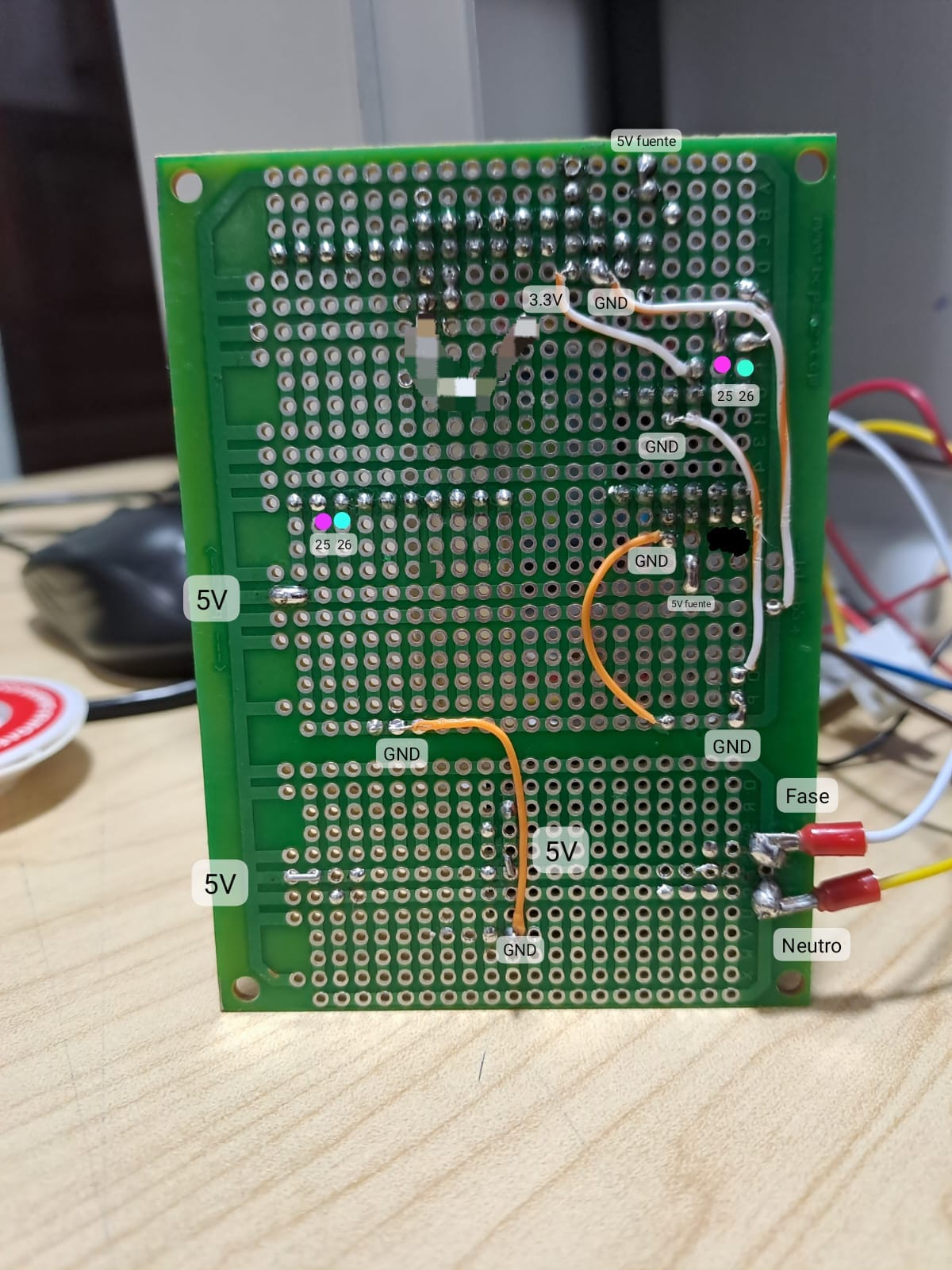


Figura. 2

Fe de erratas: Los puntos de colores rosa y azul corresponden a interconexiones cableadas.