UNIVERSIDAD TECNOLÓGICA
EMILIANO ZAPATA DEL ESTADO DE MORELOS

MEDIDOR DE CONSUMO DE ENERGÍA ELÉCTRICA

Basado en ESP32 con conexión a Thingsboard

Manual de fabricación

INDÍCE

CAPÍTULO 1. RECURSOS

1.1 Software

1.1.1 Thingsboard

Es una plataforma de software de código abierto para el monitoreo y administración de dispositivos de IoT (Internet de las cosas) y para el análisis de sensores. Permite el seguimiento de dispositivos conectados a la red y gracias a su interfaz de usuario intuitiva, los usuarios pueden crear paneles personalizados para visualizar datos en tiempo real, lo que facilita el seguimiento y gestión de dispositivos.
Thingsboard también ofrece una amplia gama de integraciones con diferentes sistemas y servicios en la nube, lo que permite una fácil integración en caso de requerirse.

Figura 1 Logo de la plataforma

1.1.2 Arduino

Arduino IDE (Integrated Development Environment), es un entorno de desarrollo integrado de código abierto que se utiliza para programar microcontroladores basados en Arduino para la creación y el desarrollo de prototipos de proyectos. Los programadores pueden escribir programas en distintos lenguajes de programación como C o C++ y cargarlos en el microcontrolador.
El software está diseñado para programadores tanto principiantes como avanzados y es compatible con diferentes sistemas operativos como Windows, MacOS y Linux.

Figura 1.1 Logo de Arduino

1.2 Hardware

1.2.1 Módulo PZEM004T v3

Es un dispositivo electrónico que se utiliza para la medición y monitoreo de la energía eléctrica en un circuito. El dispositivo es muy preciso en las mediciones que realiza, por lo que lo hace ideal para controlar el consumo energético de distintos equipos. Además, su instalación es sencilla y es compatible con varias plataformas de control de hardware, incluyendo Arduino y Raspberry Pi, lo que permite ser utilizado en proyectos de automatización y monitoreo de energía.

Algunas de sus características son:
Rango de prueba de voltaje AC 80-260V, voltaje de funcionamiento 80-260V, la corriente máxima puede alcanzar 100A. Función de medición de parámetros eléctricos que incluye medición de voltaje, corriente, potencia activa y energía.

Diseñado para probar equipos de alta potencia y puede observar el proceso acumulativo de forma intuitiva para la prueba de carga de baja potencia (máximo 100 W), Además cuenta con un puerto de comunicación RS485 y un protocolo de comunicación MODBUS para enviar los datos obtenidos a otros dispositivos.

Figura 1.2 Módulo PZEM-004T

1.2.2 ESP32

Un ESP32 es un microcontrolador de bajo costo y bajo consumo de energía que tiene la accesibilidad de conectarse por medio de Wi-Fi. Se utiliza normalmente para crear dispositivos IoT (Internet de las cosas) y productos que cuenten con conectividad Wi-Fi, por ejemplo, sensores, termostatos, cámaras y otros dispositivos más conectados a la red. La idea de esta placa es favorecer a los proyectos pequeños y sencillos, sin embargo, pueden tener un alcance mayor si así se desea.

Además, es compatible con diferentes entornos de programación como Arduino IDE, Python, LUA, entre otros.

Figura 1.3 Placa ESP32

1.2.3 ESP32 Base v1

Esta placa base funciona como ampliación de pines para el microcontrolador ESP32. Se puede acceder fácilmente a todos los puertos de E/S a través del cabezal de pines de para facilitar la creación de prototipos. También tiene un regulador de 5V integrado y acepta una entrada de alimentación de 5 a 12 V.

Figura 1.4 Base para placa ESP32

1.2.4 Jumpers

Los cables jumper se utilizan en electrónica para realizar conexiones temporales entre componentes electrónicos, especialmente en circuitos de prototipado y en placas de ensayo. Estos cables pueden contener conectores macho o hembra en los extremos, lo que permite una conexión fácil y rápida entre pines y conectores. Además, suelen tener una longitud de pocos centímetros y cuentan con colores variados para proporcionar una mejor organización y facilidad de seguimiento en proyectos más complejos.

Estos cables son una herramienta versátil y económica para la construcción de prototipos electrónicos y para la creación rápida de interconexiones temporales en el proceso de desarrollo tecnológico.

Figura 1.5 Representación cables jumpers

1.2.5 Cable 22 AWG

Es un tipo de cable eléctrico hecho de cobre y recubrimiento por una capa de aislamiento de plástico. El término “AWG” significa “American Wire Gauge” que es el estándar utilizado en los Estados Unidos para medir y clasificar el diámetro del cable.

El AWG 22 se refiere al diámetro del cable, que es 0.0253 pulgadas o 0.644 milímetros. Este tamaño de cable se utiliza generalmente para aplicaciones de baja

corriente y voltaje, como en electrónica, en circuitos de baja potencia o para la conexión de dispositivos como sensores, actuadores, interruptores, entre otros.

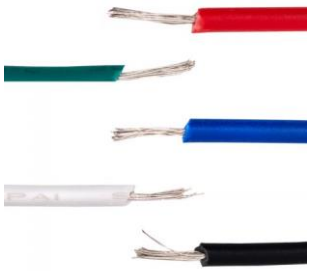

Figura 1.6 Representación cable 22 AWG

1.2.6 Diodo LED

Los diodos LED (Ligth Emitting Diode) son dispositivos de estado solido que emiten luz cuando se polarizar correctamente. Los LED funcionan a través de un fenómeno de emisión de luz producido por la recombinación de electrones en el interior del material semiconductor que compone el diodo. Estos diodos se utilizan comúnmente en en dispositivos electrónicos, como pantallas, iluminación domestica y automotriz, señalización, indicadores de estado, rótulos luminosos, entre otros. Los LED son muy populares debido a su bajo consumo de energía, su larga vida útil y su alta eficiencia luminosa.

Están disponibles en diferentes colores como rojo, verde, azul, amarillo y blanco.

Figura 1.7 Representación diodo LED

CAPÍTULO 2. DIAGRAMAS

2.1 Conexiones

Para comenzar debemos de conocer los pines con los que cuenta nuestro microcontrolador para saber dónde conectar nuestros sensores y si no existirá algún conflicto con el pin seleccionado, para ello nos guiaremos en la siguiente figura.

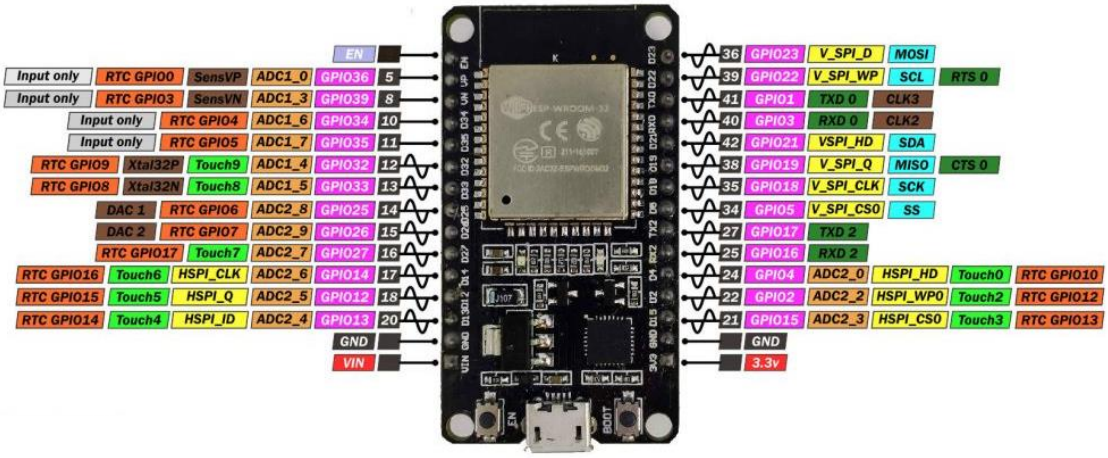

Figura 2 Pinout ESP32

Ahora se presentará la siguiente figura que contiene el datasheet de nuestro módulo PZEM004T para que conozcamos donde debemos de realizar nuestra conexión con nuestro microcontrolador y con el circuito a medir.

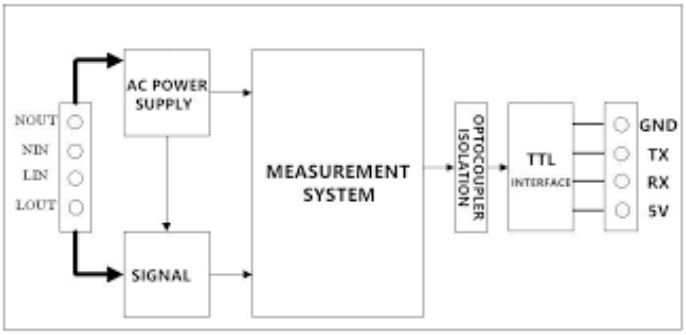

Figura 2.1 Datasheet Módulo PZEM-004T

Ahora en la siguiente figura se muestra el diagrama de conexiones de todos los componentes necesarios para la elaboración del dispositivo.

Se debe realizar la conexión de los pines de transferencia de los módulos a los pines digitales de el ESP32, los tres módulos estarán conectados a los mismos pines de comunicación RX2 y TX2 y por medio de programación serán configurados para su funcionamiento. Las líneas de voltaje de los módulos se conectan en paralelo a el pin de 3v y los pines GND de igual manera en paralelo a el pin GND de nuestra ESP32.

El led de encendido se conecta directamente a los pines de 3v y GND de nuestra placa para que cuando esta se energice el led encienda automáticamente. El que funcionara como indicador de conexión Wifi se conectara la terminal positiva a el pin digital D2 para que por medio de programación encienda cuando la placa se conecte correctamente a internet. Por último, el LED que indicara él envió de datos se conecta a el pin D4 o puede ser cambiado a uno que sea GPIO y este disponible.

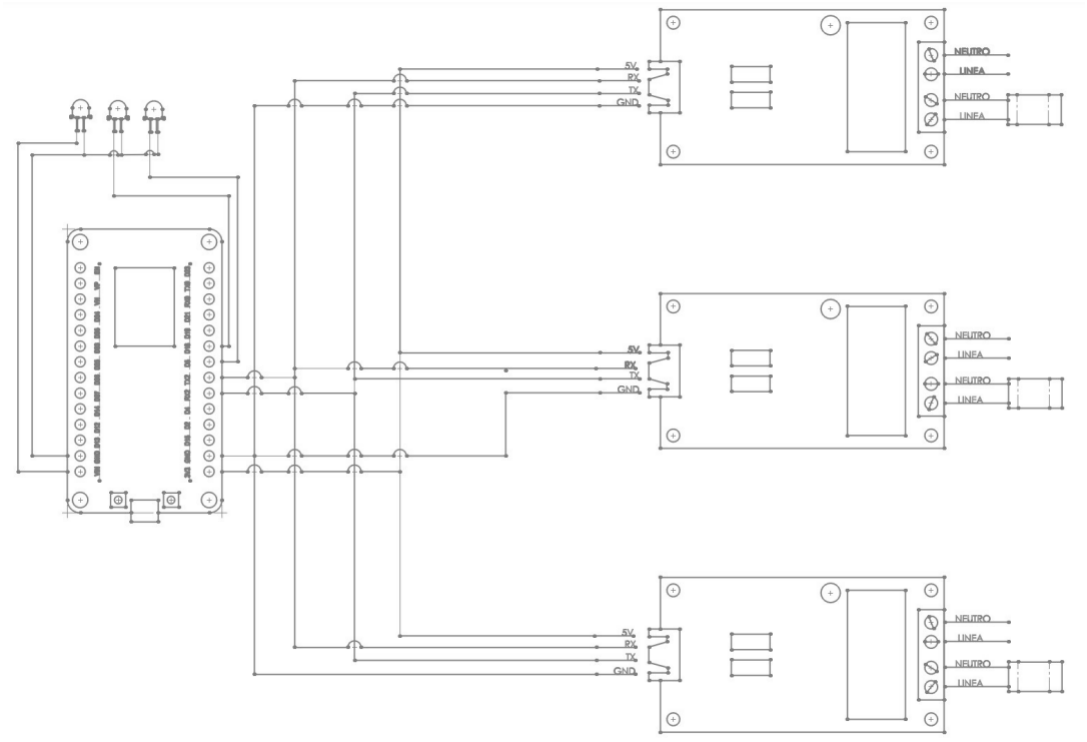

Figura 2.2 Diagrama de conexiones

3.1 Configuración Arduino

Comenzamos configurando lo que es nuestro IDE de Arduino, como sabemos vamos a utilizar una placa ESP32 y desde el gestor no se nos permite descargar la librería sin antes realizar el siguiente paso, para esto nos iremos a preferencias.

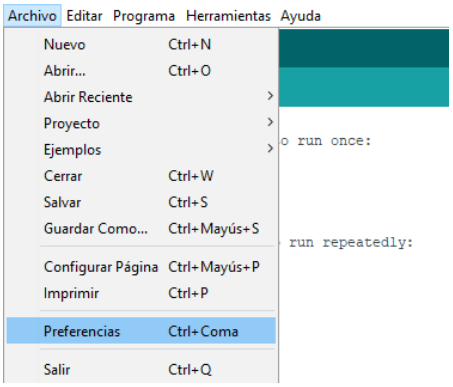

Figura 3 Dirección a preferencias

Se nos desplegara una ventana en la cual debemos de agregar el siguiente comando en la parte resaltada con azul, esto para que el software pueda realizar la búsqueda necesaria para poder descargar la librería de nuestra placa ESP32.

https://dl.espressif.com/dl/package_esp32_index.json

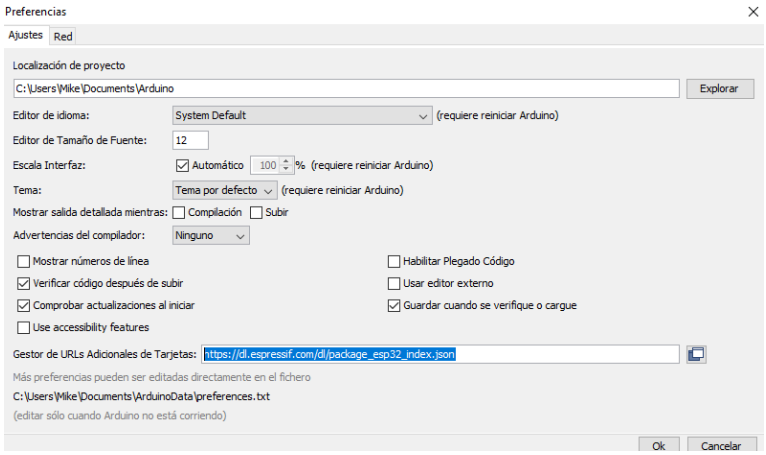

Figura 3.1 Configuración requerida

Una vez hayamos configurado y cerrado la ventana anterior nos dirigimos a la opción de Herramientas – Placa – Gestor de placa y buscamos la opción que dice ESP32 Boards.

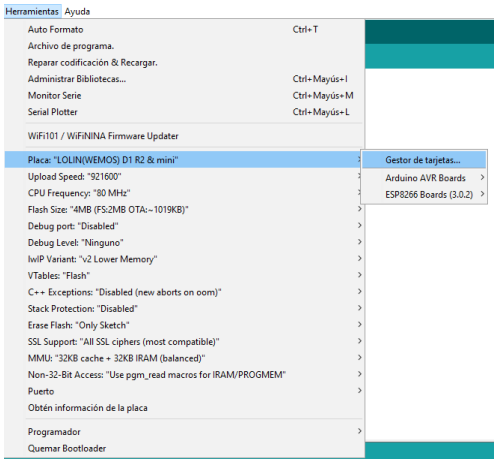

Figura 3.2 Dirección para descargar placa

Una vez la seleccionamos nos desplegara una ventana en la cual podremos gestionar nuestras tarjetas que tengamos instaladas o que requerimos instalar, en este caso en el buscador escribiremos “ESP” e instalamos el que tiene como nombre ESP32 de Espressif Systems, para este proyecto se utiliza la versión 1.0.6 debido a la compatibilidad con las librerías que se instalaran a continuación.

Figura 3.3 Gestor de tarjetas

Y ahora si nos aparecerán las tarjetas disponibles para seleccionar, configuramos nuestra tarjeta, así como el tiempo de carga y el puerto al que está conectado.
Ten en cuenta que si la configuración no es la correcta puede generar error al momento de cargar el programa a la placa.

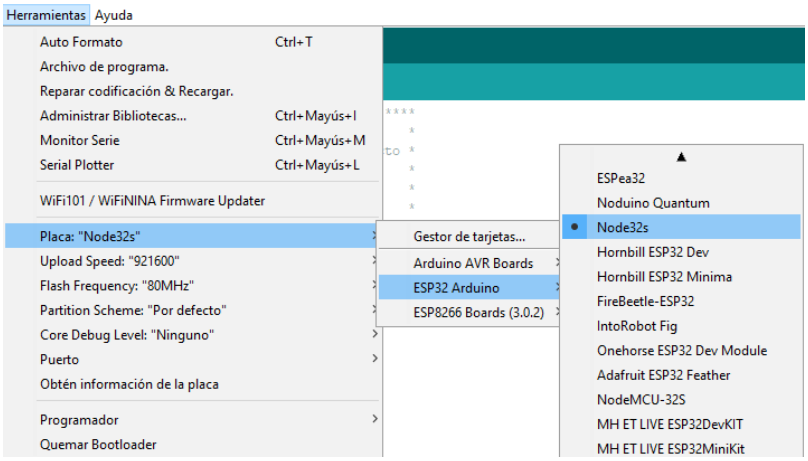

Figura 3.4 Selección de tarjeta

Una vez tengamos esta parte lo que sigue es agregar nuestras librerías que utilizaremos para el correcto funcionamiento del programa, esto se puede hacer desde el mismo software o descargando las librerías de páginas como Github y agregándolas en la siguiente dirección:
C:\Users\Admin\Documents\Arduino\libraries

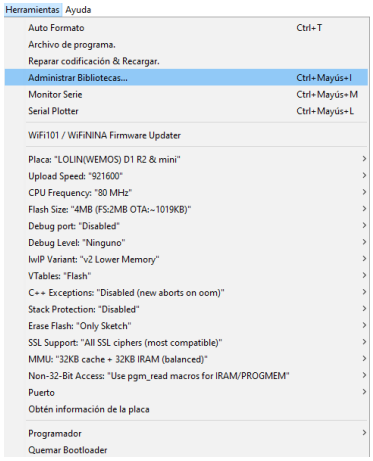

Figura 3.5 Administrador de biblioteca

Para la elaboración del proyecto se requieren una serie de librerías tanto para el funcionamiento de nuestro módulo PZEM, así como para la vinculación con la plataforma de Thingsboard y para el uso Wifi de nuestro ESP32 utilizamos las siguientes librerías con su respectiva versión.

En caso de utilizar una versión distinta asegurarse de que no existan problemas de incompatibilidad o que la versión no sea muy desactualizada porque pueden existir problemas como por ejemplo que el software serial no sea compatible.

- ArduioHttpClient - 0.4.0 - Arduino
- ArduinoJSON v6.10.1 - Benoit Blanchon
- PubSubClient v2.6.0 - Nick O'Leary
- PZEM004Tv30 v1.1.1 - Jakub Mandula
- TBPubSubClient v2.9.1 - ThingsBoard Team
- ThingsBoard v0.4.0 - Thingsoard Team

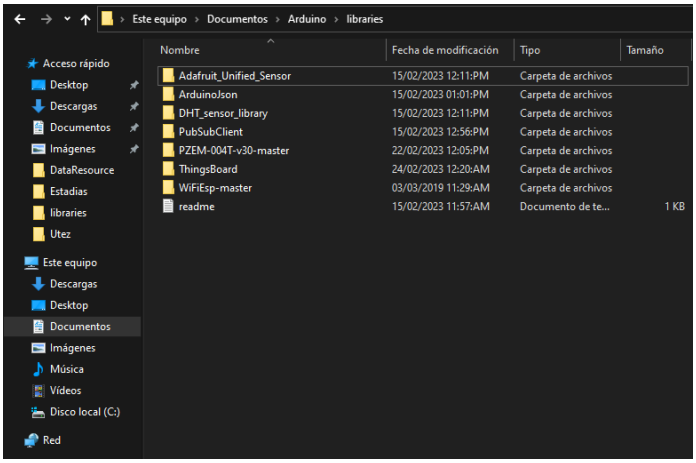

Figura 3.6 Dirección para bibliotecas

Una vez hallamos configurado todo esto procedemos con la configuración en la plataforma de Thingsboard para que los datos obtenidos sean enviados allí.

3.2 Configuración Thingsboard

Como primer paso accedemos a la plataforma de Thingsboard, en este caso en su versión demo para la realización de pruebas. La dirección es la siguiente https://demo.thingsboard.io/login

Una vez aquí debemos de crear una cuenta o acceder con alguna de las opciones disponibles.

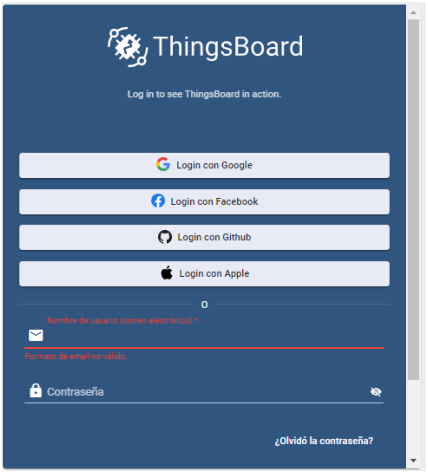

Figura 3.7 Inicio de sesión Thingsboard

Una vez accedamos nos dirigimos a el apartado “Dispositivos” el cual se encuentra en la lista de opciones del lado izquierdo.

Figura 3.8 Página de inicio Thingsboard

Nos dirigirá a una página en la cual podremos visualizar los dispositivos existentes para gestionarlos o podremos agregar nuevos dispositivos, para esto oprimimos el símbolo “+” que se encuentra del lado derecho y seleccionamos “Agregar nuevo dispositivo”.

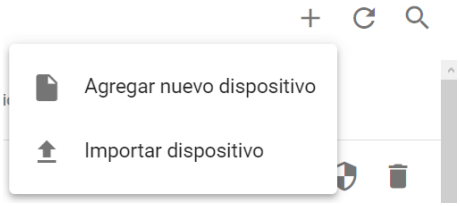

Figura 3.9 Apartado dispositivos

Se desplegará una ventana en la cual debemos de establecer el nombre de nuestro dispositivo y de manera opcional podemos agregarle una etiqueta y una descripción. Una vez terminado damos en aceptar.

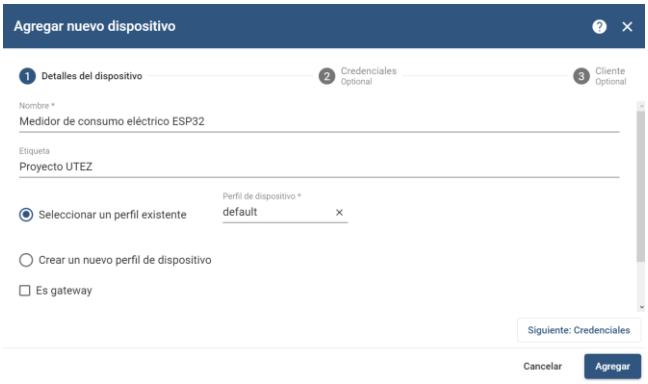

Figura 3.10 Configuración nuevo dispositivo

Nos aparecerá nuestro dispositivo en la pantalla principal, nos dirigimos al recuadro en la fila público y pulsamos, nos desplegará un mensaje de confirmación y seleccionamos sí. Con esto volveremos nuestro dispositivo público.

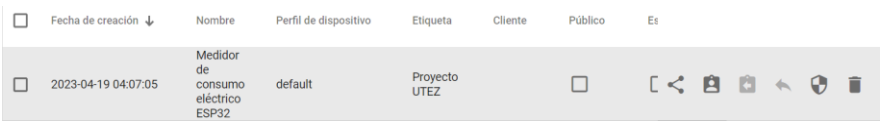

Figura 3.11 Hacer dispositivo publico

Ahora si pulsamos en el nombre de nuestro dispositivo nos desplegará una ventana en la cual veremos los detalles de nuestro dispositivo, lo que nos importa de aquí es donde dice gestionar credenciales.

Figura 3.12 Ventana de detalles

Pulsamos en dicha opción y nos desplegara una ventana con el token el cual utilizaremos en nuestra programación más adelante para poder realizar el envió de datos a este dispositivo.

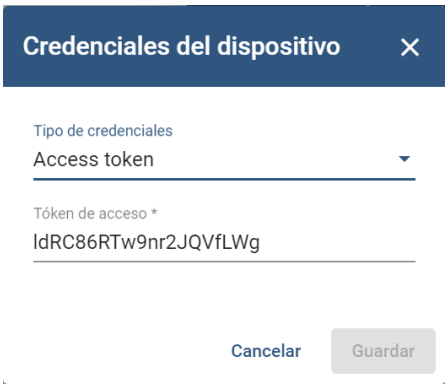

Figura 3.13 Token del dispositivo

3.3 Programación

Comenzamos modificando la dirección que contienen nuestros módulos PZEM para que puedan ser conectados por medio de MODBUS a una sola terminal RX y TX para utilizaremos el siguiente código.

La única librería que utilizamos para esto es la librería del PZEM004Tv30.

```
//Libreria a utilizar
#include <PZEM004Tv30.h>
```

Figura 3.14 Llamado de librerías

Definimos los pines a los cuales conectaremos nuestro sensor, en este caso serán el 16 y 17. Además establecemos el serial por el cual se realizará la comunicación.

```
//Definimos los pines y el serial
#define PZEM_RX_PIN 16
#define PZEM_TX_PIN 17
#define PZEM_SERIAL Serial2
```

Figura 3.15 Definición de pines

Inicializamos lo que es nuestro serial y los pines RX y TX.

```
//Inicializa del pzem con PZEM_SERIAL y los pines RX y Tx
PZEM004Tv30 pzem(PZEM_SERIAL, PZEM_RX_PIN, PZEM_TX_PIN);
```

Figura 3.16 Inicialización requerida

Y establecemos la dirección que utilizaremos, recordar que para cada sensor debe de ser una dirección distinta. Estas direcciones pueden ser desde 0x01 hasta 0xF3.

```
//Se define la direccion para el PZEM
#define SET_ADDRESS 0x03
```

Figura 3.17 Definición de dirección

Nuestro void setup solamente contendrá el tiempo al cual trabajará nuestra placa ESP32, el cual es 115200 pero este puede variar dependiendo él modelo.

```
void setup() {
  //Velocidad a la que trabaja la placa
  Serial.begin(115200);
}
```

Figura 3.18 Definición de dirección

Ahora en nuestro void loop ingresaremos la instrucción para que se establezca nuestra dirección que definimos previamente.

```
void loop() {
    //Se establece la direccion
    static uint8_t addr = SET_ADDRESS;
```

Figura 3.19 Instrucción para dirección

Y mandaremos a imprimir en el monitor serial algunos parámetros como la dirección que contenía anteriormente el PZEM.

```
//Imprime la direccion personalizada actual
Serial.print("Direccion anterior:   0x");
Serial.println(pzem.readAddress(), HEX);
```

Figura 3.20 Impresión de dirección antigua

Y también indicaremos que realice la impresión en nuestro monitor serial de la nueva dirección la cual nosotros establecimos.

```
//Imprime la direccion personalizada
Serial.print("Establecer direccion a: 0x");
Serial.println(addr, HEX);
```

Figura 3.21 Impresión de dirección nueva

Esta parte es para avisar en caso de que la dirección no haya sido cambiada por algún motivo, si aparece es importante revisar que el PZEM esté conectado correctamente.

```
//Fallo en la configuracion
if(!pzem.setAddress(addr))
{
  //Fallo, probablemente ningun PZEM conectado
  Serial.println("Error al establecer direccion");
```

Figura 3.22 Aviso que existe un fallo

De caso contrario imprimirá sin ningún problema la dirección actual la cual debe ser ya la actualizada. Es decir, se visualizará dos veces la misma dirección.

```
} else {
  //Imprime la nueva direccion personalizada
  Serial.print("Direccion actual:    0x");
  Serial.println(pzem.readAddress(), HEX);
  Serial.println();
}
```

Figura 3.23 Impresión de dirección actual

Colocamos un retardo entre cada ciclo, este puede ser a consideración de cada programador. Y cerramos lo que es nuestro void loop.

```
    //Reatardo entre ciclos
    delay(5000);
}
```

Figura 3.24 Cierre del programa

Una vez hayamos creado nuestro dispositivo en Thingsboard, hayamos modificado la dirección de cada uno de los módulos PZEM y tengamos hecho las conexiones requeridas procedemos a generar el código con el cual la placa ESP adquirirá los valores obtenidos por el PZEM-004T y a su vez enviará los datos para ser visualizados en la plataforma IoT.

Las librerías utilizadas son tanto para la vinculación con Thingsboard, el funcionamiento de nuestro módulo PZEM y la conexión wifia de la placa ESP32.

```
//Librerias
#include <Arduino.h>
#include "ThingsBoard.h"
#include <WiFi.h>
#include <PZEM004Tv30.h>
```

Figura 3.25 Llamado de librerías

Ahora definimos los pines en los cuales serán conectados nuestros pines RX y TX de cada uno de los PZEM, en este caso serán los pines 16 y 17. Además, establecemos el serial que ocuparemos para la comunicación.

```
//Definicion de pines y serial utilizado
#define PZEM_RX_PIN 16
#define PZEM_TX_PIN 17
#define PZEM_SERIAL Serial2
```

Figura 3.26 Asignación de puertos para el PZEM

Creamos unas instancias con las cuales diferenciaremos cada uno de los PZEM a la hora de adquirir los datos, en este caso siendo señalados por la línea de voltaje a la que ira dirigido.

```
//Se crea la instancia pzems
PZEM004Tv30 pzemL1;
PZEM004Tv30 pzemL2;
PZEM004Tv30 pzemL3;
```

Figura 3.27 Asignación de instancias

Definimos y configuramos las siguientes líneas de forma que ingresaremos el nombre de la red wifi a la cual se conectará la placa y su respectiva contraseña.

```
//Configuracion del nombre y contraseña Wifi
#define WIFI_AP             "Utez"
#define WIFI_PASSWORD       "TerritorioCalidad"
```

Figura 3.28 Configuración de red wifi

También realizamos la configuración del token y la dirección de la plataforma a la cual serán mandados los datos que recolectemos, esto se configura con las siguientes instrucciones.

```
//Congiguracion para Thingsboard
#define TOKEN              "ldRC86RTw9nr2JQVfLWg"
#define THINGSBOARD_SERVER  "demo.thingsboard.io"
```

Figura 3.29 Configuración de plataforma

Y también configuraremos la velocidad a la cual estará trabajando nuestra placa, nuevamente esta puede variar dependiendo el modelo.

```
//Velocidad a la que trabajara nuestra placa
#define SERIAL_DEBUG_BAUD   115200
```

Figura 3.30 Configuración de velocidad

Necesitamos inicializar el cliente para la conexión con Thingsboard y conocer el estado del wifi, para ello utilizaremos las siguientes instrucciones.

```
//Inicializa el cliente ThingsBoard
WiFiClient espClient;
ThingsBoard tb(espClient);

//Declaración del estatus del Wifi
int status = WL_IDLE_STATUS;
```

Figura 3.31 Inicialización wifi

Y para conocer el estado de una manera más sencilla visualmente utilizaremos dos LEDs los cuales servirán para conocer si el wifi conectó exitosamente y el momento en el que se envían datos, para ello debemos crear las siguientes variables y asignarles el pin correspondiente a el que están conectados, nos podemos guiar con la figura de pinouts del ESP32.

Utilizaremos los pines D2 y D4 para realizar la conexión, pero estos pueden ser modificados por cualquier pin GPIO disponible.

```
//Configuracion de variables para Leds
int conexion = 2; //PIN D2 - GPIO2
int enviar = 4; //PIN D4 - GPIO4
```

Figura 3.32 Asignación de variables LEDS

En nuestro void setup establecemos nuestros pines como salidas para el encendido de los leds, tanto para el led que indicara que la que está conectado a wifi, como para el led que indicara el momento en el que se enviaran datos a la plataforma. Además, realizamos la inicialización del wifi, así como del serial begin.

```
void setup() {
  //Configuración de pines como salida
  pinMode(conexion, OUTPUT);
  pinMode(envio, OUTPUT);

  // Inicializa el serial y el wifi
  Serial.begin(SERIAL_DEBUG_BAUD);
  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  InitWiFi();
}
```

Figura 3.33 Instrucciones en el void setup

Y también inicializaremos lo que es nuestros pines RX y TX, así como nuestro serial. Además, estableceremos las direcciones de nuestros PZEM las cuales configuramos previamente.

```
  //Inicializa el arreglo de PZEM
  pzemL1 = PZEM004Tv30(PZEM_SERIAL, PZEM_RX_PIN, PZEM_TX_PIN, 0x02);
  pzemL2 = PZEM004Tv30(PZEM_SERIAL, PZEM_RX_PIN, PZEM_TX_PIN, 0x03);
  pzemL3 = PZEM004Tv30(PZEM_SERIAL, PZEM_RX_PIN, PZEM_TX_PIN, 0x04);
}
```

Figura 3.34 Inicializaciones requeridas

Pasando con nuestro void loop, primero establecemos el tiempo con el cual se estarán repitiendo los ciclos, en este caso cada 5 segundos, pero esto se puede modificar para tener un tiempo ya sea menor o mayor, dependiendo lo que se requiera. También colocamos una instrucción que se utilizara en caso de que el estato del wifi este desconectado este se intente conectar automáticamente.

```
void loop() {
  //Tiempo entre cada ciclo
  delay(5000);

  //Instruccion para reconectar el wifi
  if (WiFi.status() != WL_CONNECTED) {
    reconnect();
  }
```

Figura 3.35 Tiempo de ciclo y reconexión wifi

Las siguientes instrucciones son para realizar la conexión con Thingsboard, se visualizará en el monitor serial una serie de leyendas en las cuales si conecta de manera correcta procederá a realizar la obtención de datos, en caso de no ser así mostrará un mensaje en el cual dirá “Fallo la conexión”.
En caso de que no se realice la conexión automáticamente iniciara la secuencia de conexión hasta que esta se logre realizar.

```
//Conexion con Thingsboard
if (!tb.connected()) {

  Serial.print("Conectando a: ");
  Serial.print(THINGSBOARD_SERVER);
  Serial.print(" Con el TOKEN ");
  Serial.println(TOKEN);

  if (!tb.connect(THINGSBOARD_SERVER, TOKEN)) {
    Serial.println("Fallo la conexion");
    return;
  }
}
```

Figura 3.36 Conexión con Thingsboard

En esta parte generamos las variables para las mediciones que realiza el PZEM, creamos tres conjuntos de instrucciones, pero con la distintiva de que deben de ser

para nuestras tres líneas. Con estas instrucciones obtendremos voltaje, corriente, potencia, energía, frecuencia y factor de potencia.

```
//Mediciones del PZEM
Serial.println("");
Serial.println("Midiendo parametros...");

//Instrucciones para las variables L1, L2 y L3
float voltsL1 = pzemL1.voltage();
float ampsL1 = pzemL1.current();
float wattsL1 = pzemL1.power();
float kiloWattsL1 = pzemL1.energy();
float hertzL1 = pzemL1.frequency();
float factorL1 = pzemL1.pf();

float voltsL2 = pzemL2.voltage();
float ampsL2 = pzemL2.current();
float wattsL2 = pzemL2.power();
float kiloWattsL2 = pzemL2.energy();
float hertzL2 = pzemL2.frequency();
float factorL2 = pzemL2.pf();

float voltsL3 = pzemL3.voltage();
float ampsL3 = pzemL3.current();
float wattsL3 = pzemL3.power();
float kiloWattsL3 = pzemL3.energy();
float hertzL3 = pzemL3.frequency();
float factorL3 = pzemL3.pf();
```

Figura 3.37 Variables para obtener datos

Colocaremos una instrucción con la cual se visualizará lo que es el PZEM que esta tomando lectura y la dirección que le pertenece.

```
//Imprime la direccion del PZEM L1
Serial.print("PZEM L1");
Serial.print(" - Direccion:");
Serial.println(pzemL1.getAddress(), HEX);
```

Figura 3.38 Variables para obtener datos

Ahora imprimiremos los datos obtenidos para visualizarlos en el monitor serial, para esto requerimos de las siguientes instrucciones en las cuales se mostrará una frase que indicará que se está midiendo, por ejemplo “Voltaje L1: “, la variable a la cual corresponde, que en este caso sería “voltsL1”, la cantidad de decimales que queramos visualizar “3” y un carácter que lo define “V”.

Así con todas las variables de cada una de las tres líneas que sean de interés visualizar.

```
//Imprimir mediciones del PZEM
Serial.println("");
Serial.println("Valores obtenidos:");

Serial.print("Voltaje L1: "); Serial.print(voltsL1,3); Serial.println("V");
Serial.print("Corriente L1: "); Serial.print(ampsL1,3); Serial.println("A");
Serial.print("Potencia L1: "); Serial.print(wattsL1,3); Serial.println("W");
Serial.print("Energia L1: "); Serial.print(kiloWattsL1,3); Serial.println("kWh");
Serial.print("Frequencia L1: "); Serial.print(hertzL1,2); Serial.println("Hz");
Serial.print("Factor de potencia L1: "); Serial.println(factorL1,3);
Serial.println("");

Serial.print("Voltaje L2: "); Serial.print(voltsL2,3); Serial.println("V");
Serial.print("Corriente L2: "); Serial.print(ampsL2,3); Serial.println("A");
Serial.print("Potencia L2: "); Serial.print(wattsL2,3); Serial.println("W");
Serial.print("Energia L2: "); Serial.print(kiloWattsL2,3); Serial.println("kWh");
Serial.print("Frequencia L2: "); Serial.print(hertzL2,2); Serial.println("Hz");
Serial.print("Factor de potencia L2: "); Serial.println(factorL2,3);
Serial.println("");


Serial.print("Voltaje L3: "); Serial.print(voltsL3,3); Serial.println("V");
Serial.print("Corriente L3: "); Serial.print(ampsL3,3); Serial.println("A");
Serial.print("Potencia L3: "); Serial.print(wattsL3,3); Serial.println("W");
Serial.print("Energia L3: "); Serial.print(kiloWattsL3,3); Serial.println("kWh");
Serial.print("Frequencia L3: "); Serial.print(hertzL3,2); Serial.println("Hz");
Serial.print("Factor de potencia L3: "); Serial.println(factorL3,3);
Serial.println("");
```

Figura 3.39 Instrucciones para visualizar datos

Con esto podemos visualizar que los tres modulo estén trabajando de manera correcta para proceder con el envió de datos el cual se explica a continuación.

Las primeras líneas son instrucciones para imprimir un mensaje el cual avise que los datos están siendo enviados. Las siguientes líneas son instrucciones que contiene la librería de Thingsboard para el envío de datos.

Primero debemos de seleccionar el tipo de dato que enviamos, este puede ser entero o flotante, en nuestro caso será “Float”, la leyenda con la cual se visualizara en la plataforma “Voltaje L1” y nuestra variable que contiene los valores “voltsL1”.

```
//Mensaje de envio de datos
Serial.println("Enviando a la plataforma...");
Serial.println("");

//Envio de datos a Thingsboard
tb.sendTelemetryFloat("Voltaje L1", voltsL1);
tb.sendTelemetryFloat("Corriente L1", ampsL1);
tb.sendTelemetryFloat("Potencia L1", wattsL1);
tb.sendTelemetryFloat("Energia L1", kiloWattsL1);
tb.sendTelemetryFloat("Frecuencia L1", hertzL1);
tb.sendTelemetryFloat("FP L1", factorL1);

tb.sendTelemetryFloat("Voltaje L2", voltsL2);
tb.sendTelemetryFloat("Corriente L2", ampsL2);
tb.sendTelemetryFloat("Potencia L2", wattsL2);
tb.sendTelemetryFloat("Energia L2", kiloWattsL2);
tb.sendTelemetryFloat("Frecuencia L2", hertzL2);
tb.sendTelemetryFloat("FP L2", factorL2);

tb.sendTelemetryFloat("Voltaje L3", voltsL3);
tb.sendTelemetryFloat("Corriente L3", ampsL3);
tb.sendTelemetryFloat("Potencia L3", wattsL3);
tb.sendTelemetryFloat("Energia L3", kiloWattsL3);
tb.sendTelemetryFloat("Frecuencia L3", hertzL3);
tb.sendTelemetryFloat("FP L3", factorL3);
```

Figura 3.40 Instrucciones para envío de datos

También está la instrucción FOR para que en cada envío de datos se encienda y apague un Led un tal de 5 veces. Y cerramos nuestro void loop.

```
  //Indicador led de envio de datos
  for (int conteo=0; conteo<5; conteo++) {
    digitalWrite(enviar, HIGH);
    delay(50);
    digitalWrite(enviar, LOW);
    delay(50);
  }

  tb.loop();
}
```

Figura 3.41 Instrucciones para envío de datos

La siguiente instrucción es la encargada de realizar la conexión con la red wifi en nuestra placa ESP32, comprobando el nombre de la red y la contraseña procederá a realizar la conexión, durante el proceso se visualizará un tiempo de carga y cuando la conexión sea exitosa arrojará el mensaje “Conectado a AP” y el Led de conexión encenderá.

```
//Conexión Wifi
void InitWiFi()
{
  Serial.println("Conectando a AP...");

  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Conectado a AP");
  digitalWrite(conexion, HIGH);
}
```

Figura 3.42 Instrucción conexión wifi

Esta instrucción es para el caso en que el wifi se desconecte este realice la tarea de volver a conectarse de manera automática para continuar con el proceso de obtención de datos.

```
//Reconexion Wifi
void reconnect() {

  status = WiFi.status();
  if ( status != WL_CONNECTED) {
    WiFi.begin(WIFI_AP, WIFI_PASSWORD);
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("Conectado a AP");
    digitalWrite(conexion, HIGH);
  }
}
```

Figura 3.43 Instrucción reconexión wifi

3.4 Visualización de datos

Una vez tengamos la programación realizada podemos cargar nuestro código a nuestra placa y comenzar a obtener datos, los cuales desde el monitor serie se visualizarían de la siguiente forma.

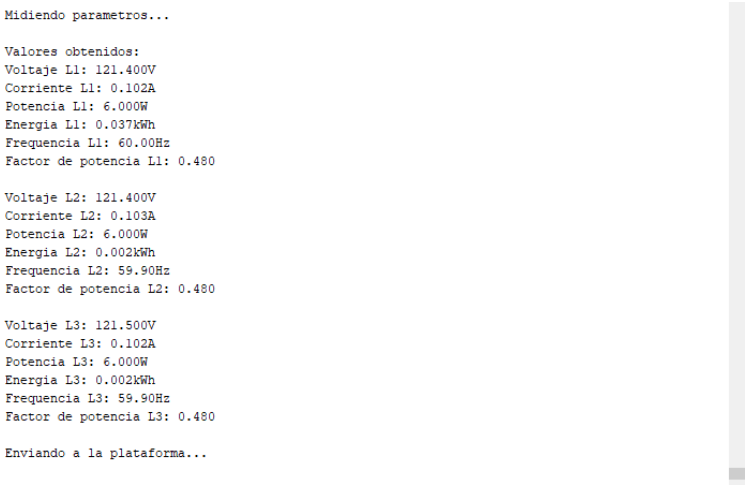

Figura 3.44 Visualización en el monitor serial

Y en la plataforma desde la ventana que se abre cuando seleccionamos nuestro dispositivo esta la opción que lleva por nombre telemetría, en donde podremos visualizar los datos recibidos. En la siguiente imagen la demostración.

Figura 3.45 Telemetría en Thingsboard

3.5 Creación de widgets

Para crear los widgets que nos ayudaran a visualizar de una mejor manera los datos que vayamos recolectando es necesario primero crear un panel, para esto nos dirigimos a el apartado “Paneles” que se encuentra en las opciones del lado izquierdo.

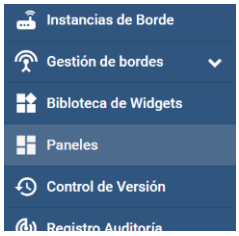

Figura 3.46 Creación del panel

Abrirá una ventana en la cual nos mostrará los paneles existentes, nos dirigiremos a el lado derecho y buscaremos el símbolo “+”, lo pulsamos y seleccionamos “Crear nuevo panel”

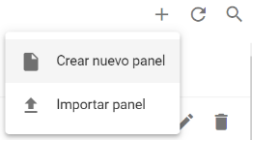

Figura 3.47 Creación del panel

Nos desplegara la siguiente ventana en la cual debemos de asignar un nombre para nuestro panel y de manera opcional un icono y una descripción.

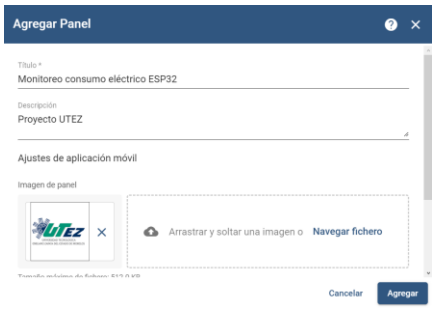

Figura 3.48 Configuración panel

Cuando se haya creado deberemos de realizar el proceso de volver el panel público, para ello nos dirigimos al icono que se encuentra del lado derecho y lo pulsamos.

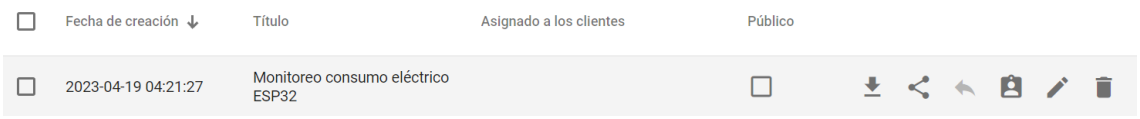

Figura 3.49 Volver público el panel

Nos desplegara una ventana en la cual debemos de confirmar el proceso, también ahí nos mostrara un link con el cual podremos visualizar nuestro panel desde cualquier dispositivo. En este caso es el siguiente.

https://demo.thingsboard.io/dashboard/f1d61630-de9b-11ed-877a-2b8500ce9bd7?publicId=01631200-b332-11ed-b62c-7d8052ad39cf

Ahora seleccionamos nuestro panel y nos dirigirá a una pagina en la cual tendremos nuestro espacio de trabajo para agregar los widgets que requerimos para una buena visualización, solo debemos de dirigimos al botón naranja con un lápiz y seleccionamos donde dice “Crear nuevo widget”

Nos desplegara la siguiente ventana en la cual debemos de ingresar nuestro parámetro que queremos visualizar “Creamos un nuevo alias para cada parámetro” y el dato que mostrara, en este caso tendrán el nombre de las variables que programamos anteriormente.

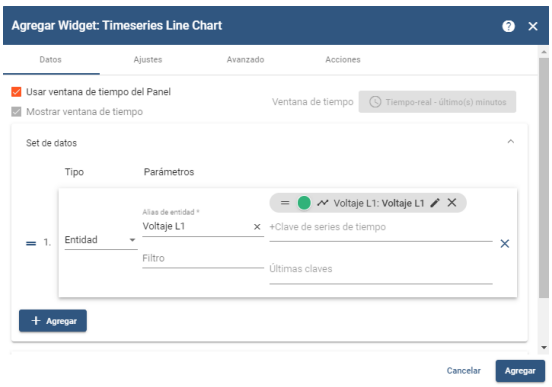

Figura 3.50 Ingresar parámetro

Podemos configurar el color con el cual se visualizará distintos aspectos, como el fondo del widget o la línea que mostrará la gráfica. La paleta de colores utilizados en nuestro caso son los siguientes y con ellos configuraremos la gran mayoría de los objetos en nuestro panel.

Figura 3.51 Paleta de colores

Ahora solo queda agregar todos datos a visualizar, por ejemplo, un panel de grafica en la cual se visualice el voltaje de las tres líneas, para hacerlo es el mismo proceso, creamos un alias con el nombre de dato que deseemos, seleccionamos nuestro dispositivo y pulsamos en agregar.

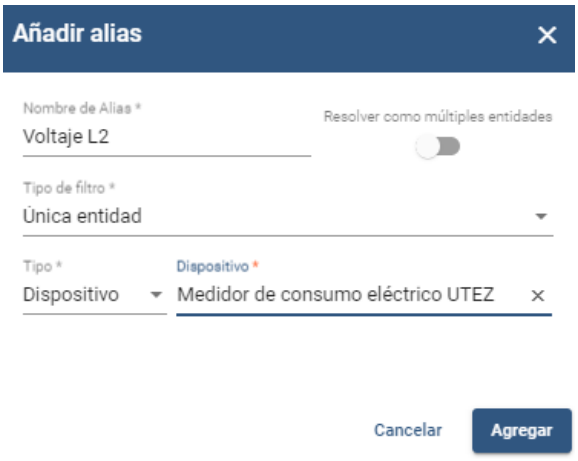

Figura 3.52 Creación del alias

Ahora creamos nuestros tres alias para nuestras tres líneas y configuramos la variable que se mostrara la cual previamente visualizamos en la telemetría.

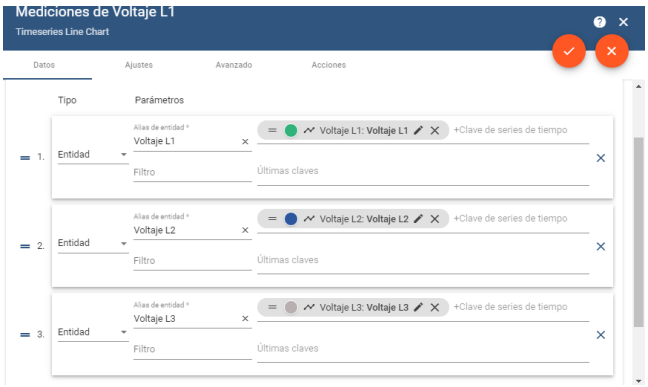

Figura 3.53 Vista de las tres variables

En otros ajustes podemos configurar aspectos generales como, sombras, color o títulos que pueden ser mostrados.

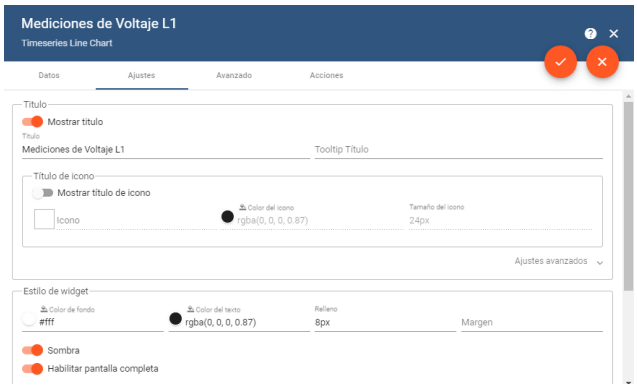

Figura 3.54 Configuraciones extras

Sin el dispositivo enviando datos debe de verse de la siguiente forma.

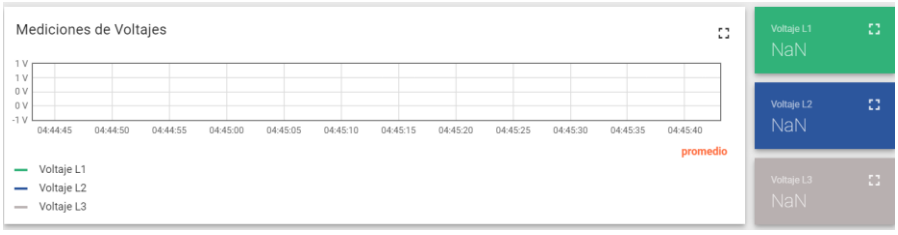

Figura 3.55 Panel mediciones voltaje

Una vez hayamos creado y configurado todos nuestros paneles y pongamos en funcionamiento nuestro dispositivo se visualizarán de la siguiente manera.

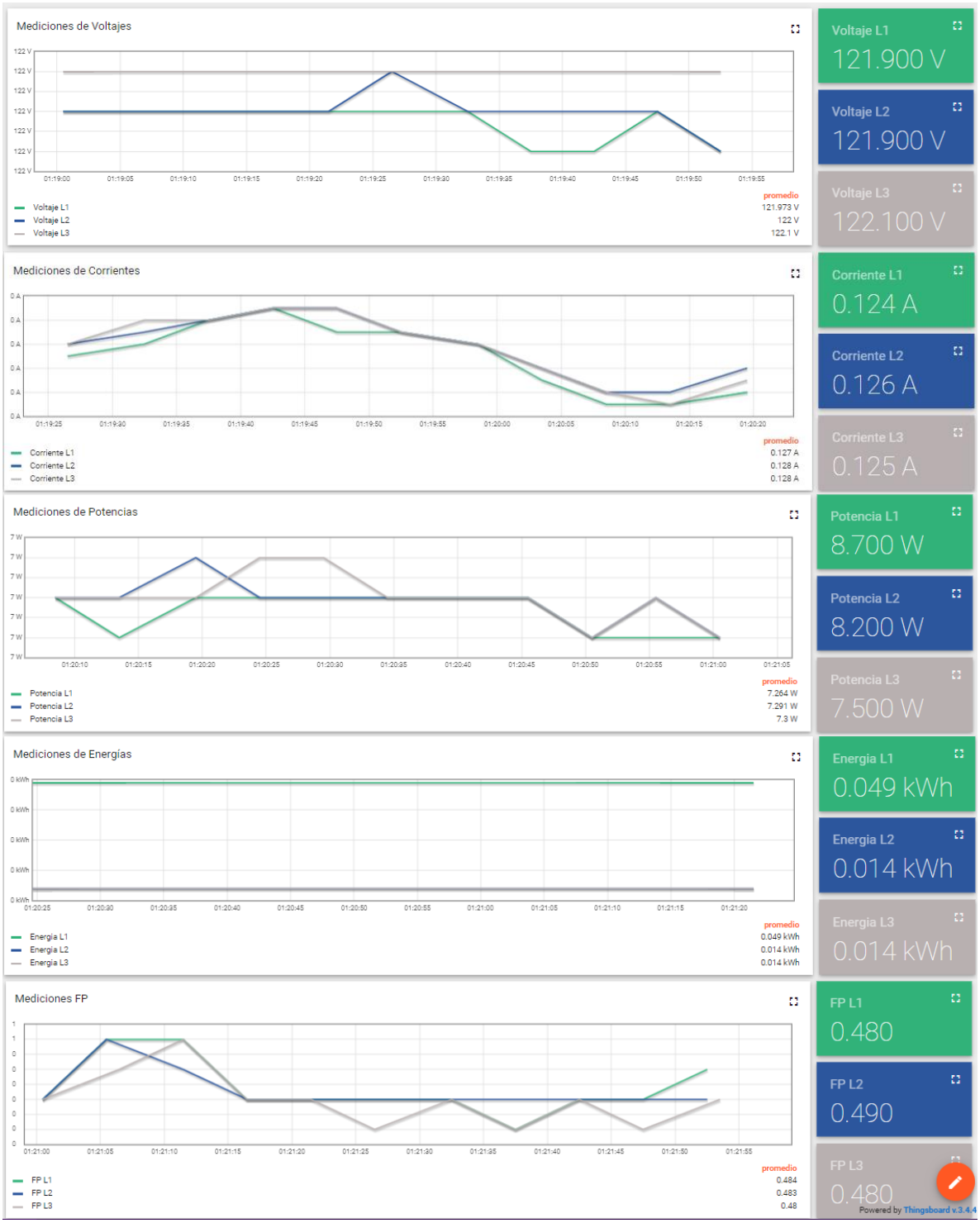

Figura 3.56 Visualización de widgets

Y a su vez por medio del link que se nos otorgó previamente, podemos visualizarlos directamente desde un dispositivo móvil como se muestra a continuación.

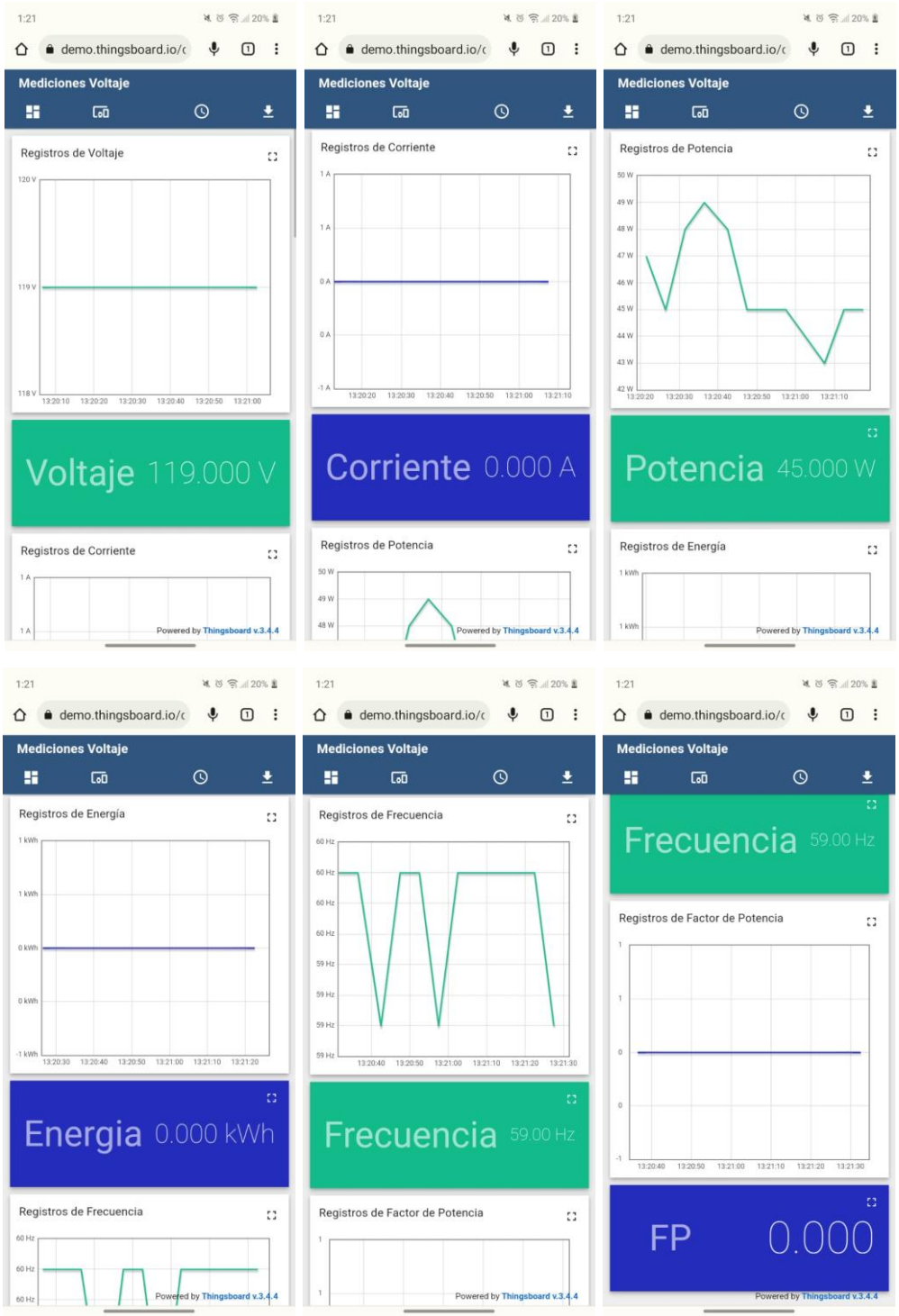

Figura 3.57 Visualización de widgets en otro dispositivo