14 de febrero de 2020

Modelo Piloto Medidores DENMARK

14 de febrero de 2020

Objetivo

- Poder diseñar medidores que tenga la capacidad de enviar información de las lecturas cada 10 segundo, cada 15 minutos de acuerdo con normativa de la SIGET y que a la vez guarde la información en un registro de datos. Los medidores deben tener la capacidad de conectarse a internet vía wifi y enviar la información a un MQTT, en el caso que el internet o el servidor falle el medidor debe tener la capacidad de seguir guardando la información interna y posteriormente cuando se normalice el internet o el servidor poder enviar dicha información.
- Poder realizar los medidores con todas las características mencionada con un bajo presupuesto.

Introducción

El propósito de este documento es describir la funcionalidad, diseño, justificación de los módulos seleccionados, programación e instalación de los medidores DENMARK. Este documento incluirá datos técnicos de cada uno de los módulos y su conexión. Los módulos seleccionados son los siguientes:

- Medidor EASTRON SD120M monofásico/ SD220 trifásico
- ESP8266 NodeMCU
- Modulo Lector MicroSD
- DS3231 RTC
- Convertidor TTL - RS485

Descripción de los componentes:

- Medidor EASTRON SDM120M y SDM220

Representa una solución avanzada de monitoreo de energía. Está equipado con una pantalla LCD para visualización de datos, especialmente indicada para la medición de la energía activa y otros parámetros y para el cálculo de costos.

El dispositivo está diseñado en un gabinete de montaje en riel DIN, con clasificación de protección IP51 y equipado con cubiertas sellables (terminal de fase y neutro). El instrumento también está equipado con una salida de pulso, proporcional a la energía activa medida, y un puerto de comunicación RS485 para monitoreo remoto.

Este medidor no es un Smart meter, pero tiene la capacidad de recolectar información y mandarla a otro dispositivo a otro dispositivo a través del puerto RS485 que se comunica por el protocolo de comunicación MODBUS.

- ESP8266 NodeMCU

El ESP8266 es un chip altamente integrado diseñado para las necesidades de un nuevo mundo conectado. Ofrece una solución completa y autónoma de redes Wi-Fi, lo que le permite alojar la aplicación o servir como puente entre Internet y el microcontrolador.

El ESP8266 tiene dos características importantes por las cuales lo estamos usando la primera es debido a que el chip por si solo tiene la capacidad de conectarse a una red wifi la cual será el medio de transmisión de información y la segunda característica y más importante es debido a su bajo costo.

- Convertidor TTL - RS485

El protocolo de comunicación entre el medidor y el NodeMCU no es el mismo por lo cual es necesario usar un convertidor el cual pueda filtrar la señal para que microcontrolador sea capaz de poder recibir la información correctamente.

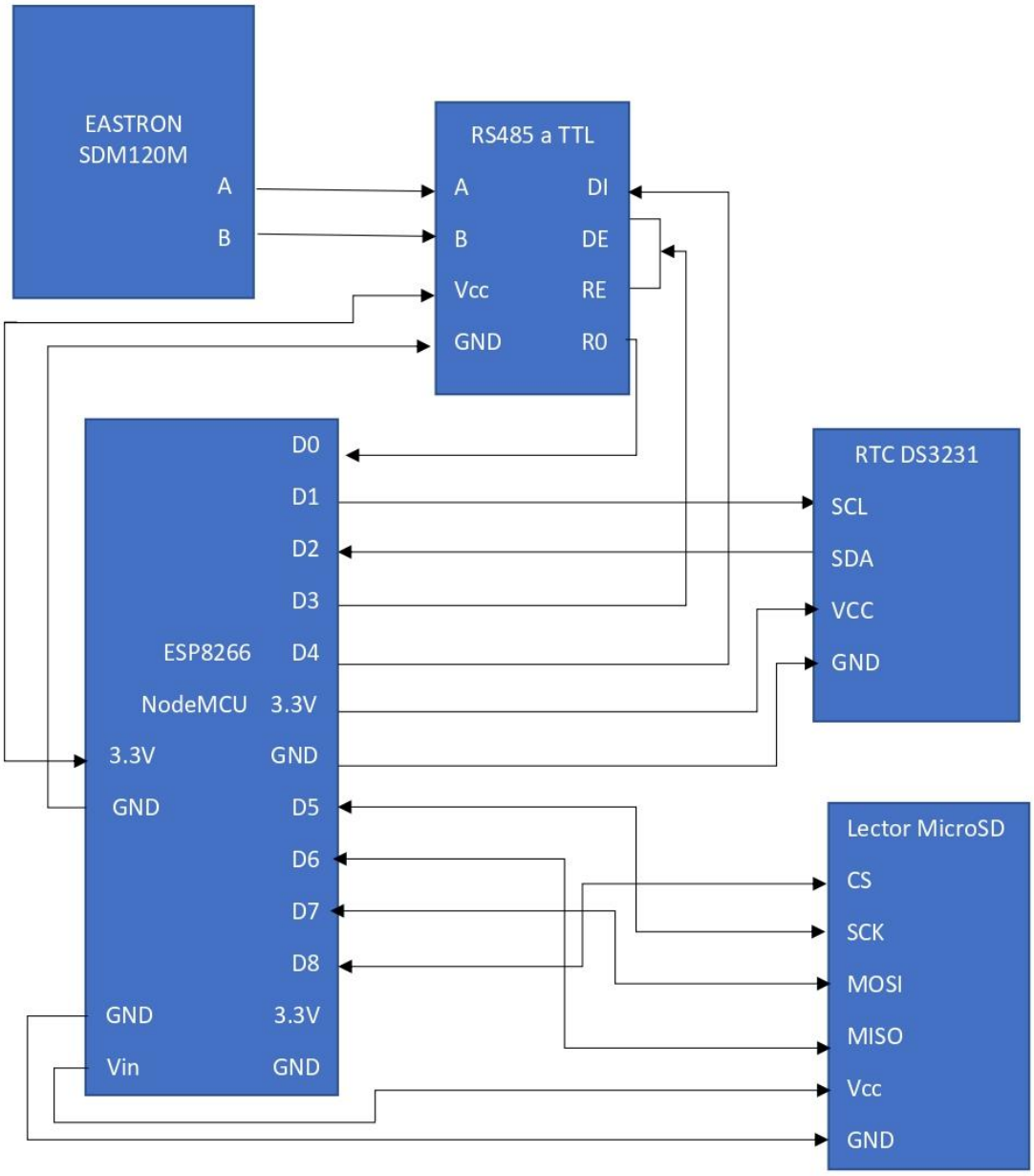
- Modulo Lector MicroSD

Para poder guardar la información de la toma de datos cada 15 minutos se ha decidido usar un lector microSD para Arduino ya que es compatible con ESP8266 y podemos realizar ciclos de escritura y de lectura sin reducir su vida útil.

- DS3231 RTC

Este módulo es el encargado de guardar la variable fecha y hora, aunque todo nuestro equipo tenga ausencia de electricidad gracias a dicho modulo no se perderían estas variables ya que cuenta con una batería de 3V y debido a que posee compensación de la temperatura garantizando una precisión de al menos 2ppm, lo que equivale a un desfase máximo 172ms/día o un segundo cada 6 días.

Diagrama de conexión



Programación

Para la programación del ESP8266 utilizamos varias librerías de Arduino. Dichas librerías tenían la función de facilitar la comunicación, parametrización y conversión de información entre en medidor el ESP8266 para luego modelar la información que se envía al MQTT. Las librerías son las siguientes:

SDM.h

El uso de esta librería nos facilita el trabajo de comunicación entre el medidor y el ESP8266 debido a que todo el protocolo de comunicación ya se encuentra parametrizados por lo que usando funciones de la librería extraemos fácilmente diferentes parámetros del medidor.

WiFiManager.h

Esta librería nos permite guardar una red wifi desde un dispositivo con capacidad de conectarse a una red wifi, si el ESP8266 no encuentra ninguna red conocida o no tiene ninguna guardada crea un punto de acceso en el cual desde una computadora, celular o Tablet puede entrar y conectarse a una red wifi, con uso de una función de la librería se puede agregar parámetros que serán escritos desde la configuración del wifi que se guardaran en el microcontrolador.

RTClib.h

Configura y guarda la variable tiempo y hora en el ds3231 RTC y con el uso de funciones de la librería poder extraer la variable fecha y hora.

PubSubClient.h

Para el uso de MQTT únicamente colocando el usuarios, contraseña, servidor y puerto podemos acceder y mandar información especificando a un topico especifico.

EEPROM.h

Con el uso de esta librería se guardan unas cuantas variables las cuales se alojaran en la memoria EEPROM del ESP8266 esta memoria tiene la característica de ser una memoria no volatil , esto significa que las variables se guardan a pesar de reinicios o ausencia de energía en el sistema.

SD.h

Con el uso de esta librería podemos leer y escribir en una memoria microSD.

ArduinoJson.h

Los datos de que obtenemos de el medidor para ser enviados deben ser enviados en forma de objeto siguiendo una sintaxis, para ello se usa dicha librería con la cual convertimos la información que la obtenemos en formato String en un Json.

La información que es enviada a MQTT tiene el siguiente formato:

Cada 10 segundos:

{"date":"2020-02-10 15:23:45","serialNumber":"01240735","measures":{"Readings: Volts Average":"40.22","Readings: Volts A-N":"120.57","Readings: Volts B-N":"0.00","Readings: Volts C-N":"0.00","Readings: I A":"0.00","Readings: I B":"0.00","Readings: I C":"0.00","Readings: Frequency":"59.96","Readings: Watts Total":"0.00","Readings: Watts A":"0.00","Readings: Watts B":"0.00","Readings: Watts C":"0.00","Scaled Energy: Wh Net":"13555.98","Scaled Energy: Wh Net A":"0.00","Scaled Energy: Wh Net C":"0.00","Scaled Energy: Wh received":"19.00","Scaled Energy: Wh received A":"19.00","Scaled Energy: Wh received B":"0.00","Scaled Energy: Wh received C":"0.00","Scaled Energy: Wh delivered":"0.00","Scaled Energy: Wh delivered A":"0.00","Scaled Energy: Wh delivered B":"0.00","Scaled Energy: Wh delivered C":"0.00","Scaled Energy: Wh Total":19,"Scaled Energy: Wh Total A":"19.00","Scaled Energy: Wh Total B":"0.00","Scaled Energy: Wh Total C":"0.00"}}

Cada 15 minutos:

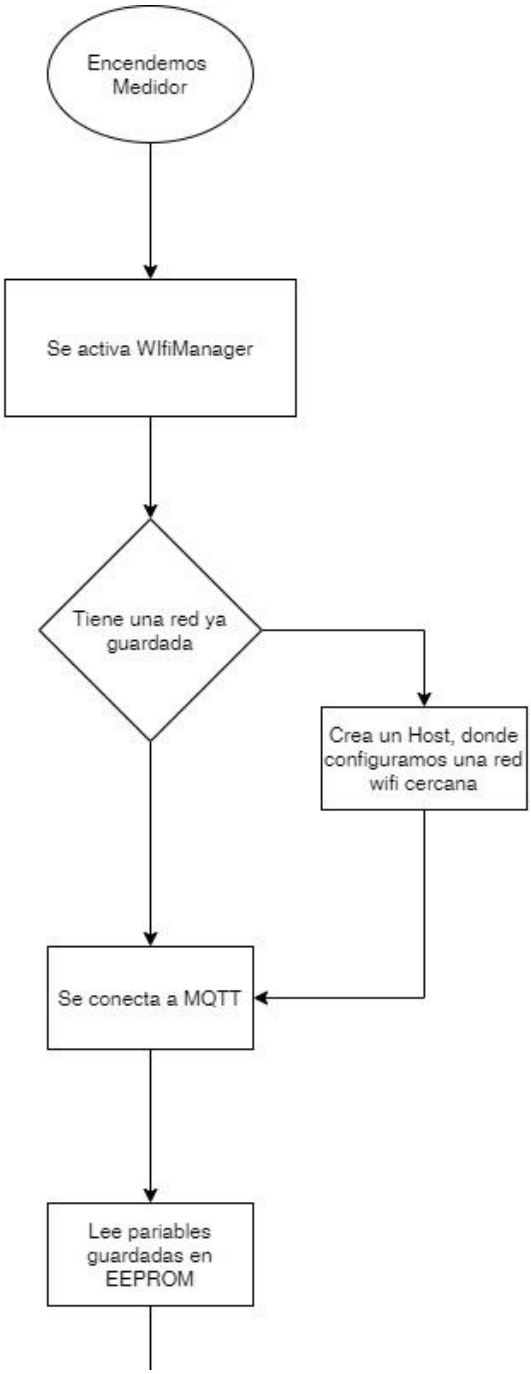
{"date":"2020-02-10 15:30","serialNumber":"01240735","measures":[{"tipo":"Voltaje","fase":"A","lectura":"120.73"},{"tipo":"Voltaje","fase":"B","lectura":"0.00"},{"tipo":"Voltaje","fase":"C","lectura":"0.00"},{"tipo":"Corriente","fase":"A","lectura":"0.00"},{"tipo":"Corriente","fase":"B","lectura":"0.00"},{"tipo":"Corriente","fase":"C","lectura":"0.00"},{"tipo":"Frecuencia","fase":"Total","lectura":"59.93"},{"tipo":"Potencia","fase":"A","lectura":"0.00"},{"tipo":"Potencia","fase":"B","lectura":"0.00"},{"tipo":"Potencia","fase":"C","lectura":"0.00"},{"tipo":"Energy: Wh Total","fase":"A","lectura":"19.00"},{"tipo":"Energy: Wh Total","fase":"B","lectura":"0.00"},{"tipo":"Energy: Wh Total","fase":"C","lectura":"0.00"},{"tipo":"Energy: Wh received","fase":"A","lectura":"19.00"},{"tipo":"Energy: Wh received","fase":"B","lectura":"0.00"},{"tipo":"Energy: Wh received","fase":"C","lectura":"0.00"},{"tipo":"Energy: Wh delivered","fase":"A","lectura":"0.00"},{"tipo":"Energy: Wh delivered","fase":"B","lectura":"0.00"},{"tipo":"Energy: Wh delivered","fase":"C","lectura":"0.00"}]}

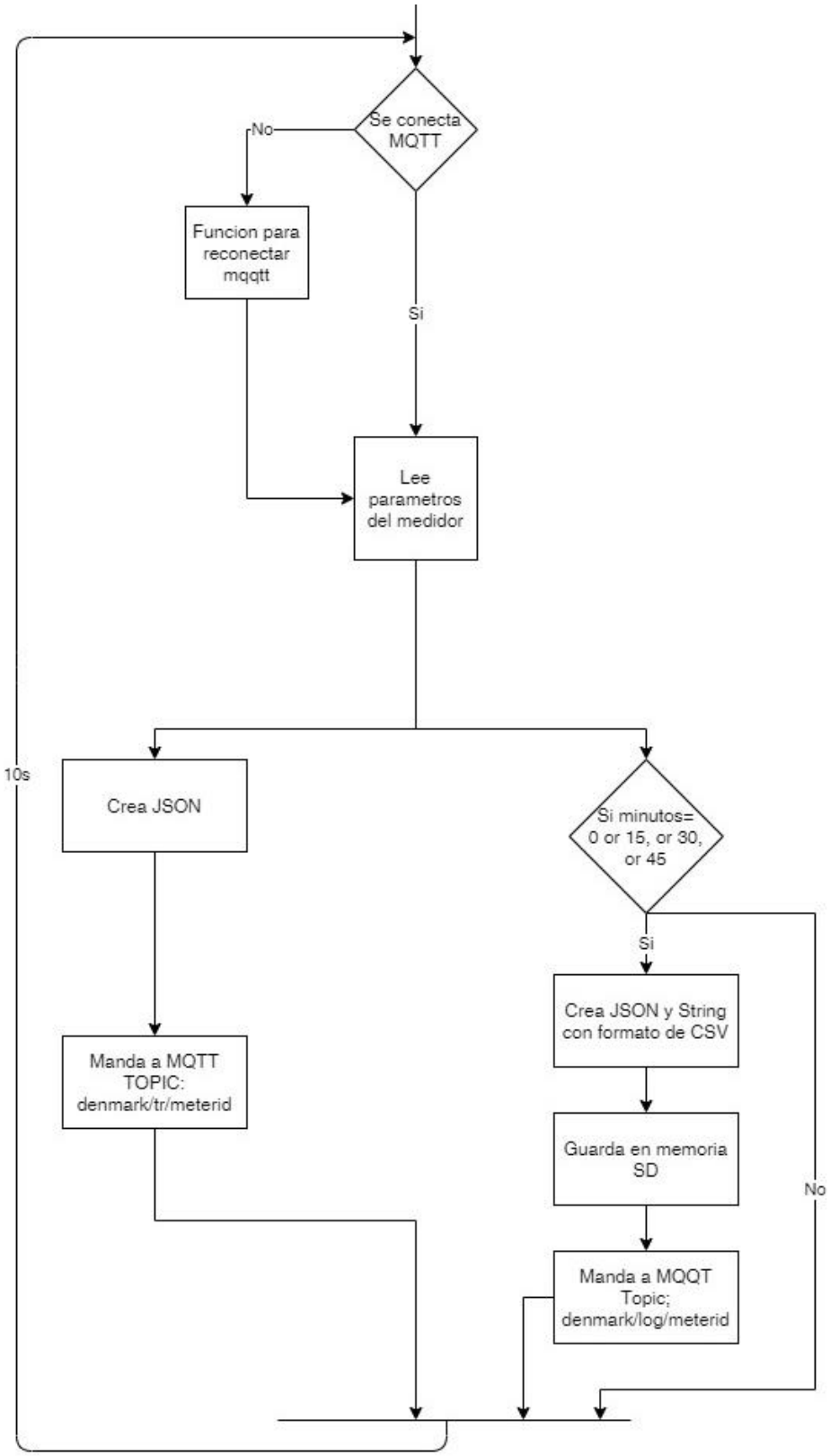
La información que se guarda en la memoria SD sigue la siguiente formato:

CSV Formato:

date,serialNumber,Voltaje promedio(V),Voltale A-N(V),Voltale B-N(V),Voltale C-N(V),Corriente promedio(A),Corriente A(A),Corriente B(A),Corriente C(A),Frecuencia (Hz),Potencia total(W),Potencia A(W),Potencia B(W),Potencia C(W),Energia Neta total(Wh),Energia Neta A(Wh),Energia Neta B(Wh),Energia Neta C(Wh),Energia Importada total(Wh),Energia Importada A(Wh),Energia Importada B(Wh),Energia Importada C(Wh),Energia Exportada total(Wh),Energia Exportada A(Wh),Energia Exportada B(Wh),Energia Exportada C(Wh),Energia Total (Wh),Energia Total A(Wh),Energia Total B(Wh),Energia Total (Wh) C

Diagrama de flujo




Se conecta MQTT
No
Funcion para reconectar mqqtt
Si
Lee parametros del medidor
10s
Crea JSON
Manda a MQTT TOPIC: denmark/tr/meterid
Si minutos= 0 or 15, or 30, or 45
Si
Crea JSON y String con formato de CSV
Guarda en memoria SD
Manda a MQQT Topic; denmark/log/meterid
No

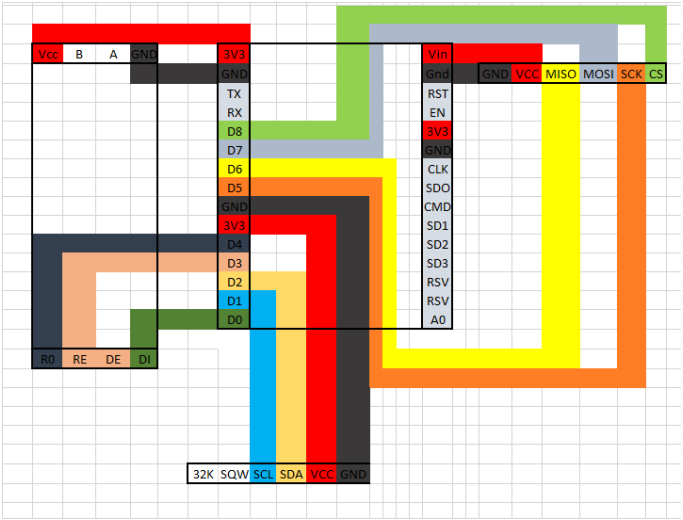
14 de febrero de 2020

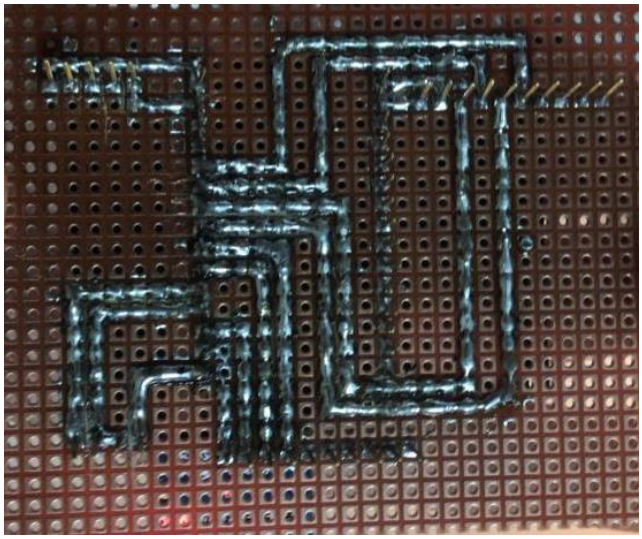
Explicación de la programación:

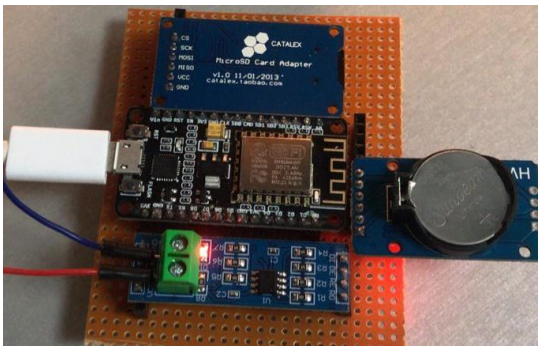
Inicamos con la librería WiFiManager busca las redes cercanas y si tiene configurado alguna de ellas se conecta de no ser así crea una host, desde el cual puedes entrar desde un celular o una computadora, después de conectarse a la red, entra al servidor de MQTT, inicia los demás módulos para asegurarse del buen funcionamiento del módulo RTC y el lector microSD y de ultimo lee las variables alojadas en la memoria EEPROM. Cuando termina todo los procesos de setup entra al void loop, en este realiza la inspección si está conectado al mqtt y corre una función la cual realiza la lectura de los datos cada 10 segundos, de ahí tomamos la información y le un formato .Json manda la información al topico de denmark/tr/meterid y si en el minuto en que hace la lectura es 0, 15, 30 o 45 va mandar la información a denmark/log/meterid y al mismo tiempo se guarda la información en la memoria SD, al correr por primera vez el programa guarda la fecha en la cual se realizó, en esa carpeta se guardara la información de la mediciones de cada 15 minutos y dentro de una semana creara una nueva carpeta.

Modelo piloto:

Ya teniendo el diagrama de conexión y el programa realizamos la instalación del primer medidor en una instalación fotovoltaica residencial. Para colocar todos los componentes se utilizó un PCB preperforado en la cual se soldaron todos los componentes, para instalar todo se ubico dentro de una caja plástica arriba de la caja del sistema fotovoltaico.

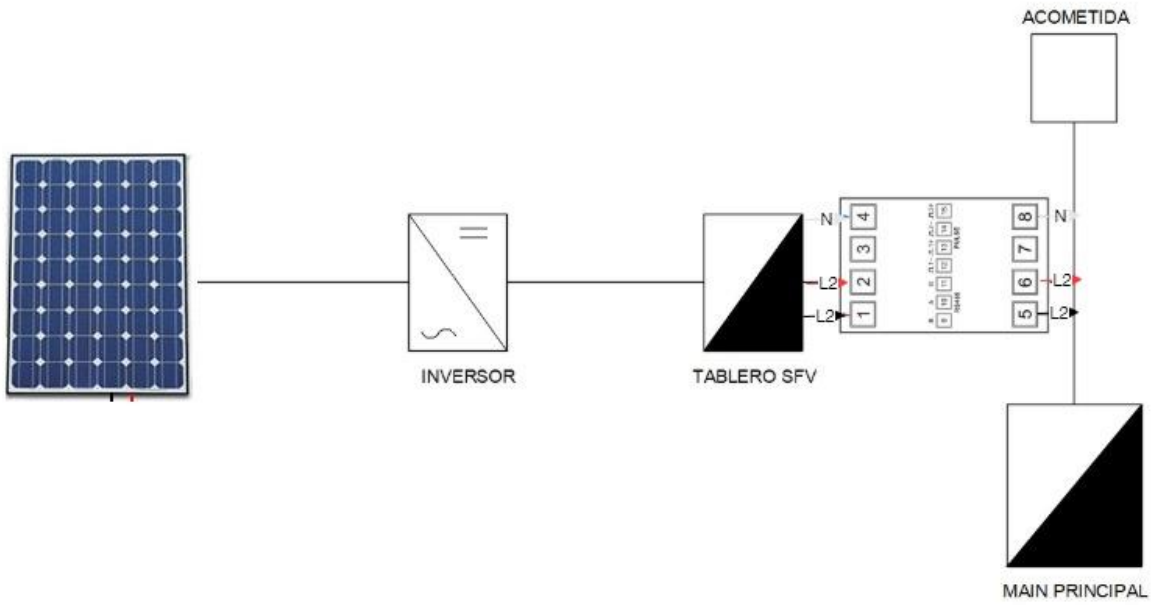






El diagrama de conexión fue el siguiente:

El sistema fotovoltaico estaba conectado a la línea que viene del medidor de la distribuidora antes del tablero principal, el medidor DENMARK se conectó de la siguiente forma:



El resultado de la instalación fue la siguiente:

Codigo:

```
#include <FS.h>
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <PubSubClient.h>
#include <SDM.h>
#include <SoftwareSerial.h>
#include <ArduinoJson.h>
#include <RTClib.h>
#include <Wire.h>
      #include <SPI.h>
      #include <SD.h>
      #include <EEPROM.h>
      #include <WiFiManager.h>
      #include <ESP8266WebServer.h>
      #include <DNSServer.h>

      File myFile;
      #define ESP8266
      int band =1;
      int band_day =1;
      int dia_next;
      int mes_next;
      int dia_nextEEPROM;
```

```
24.       int mes_nextEEPROM;
25.       char t2_EEPROM[15];
26.       int anho_EEPROM;
27.       int dia_EEPROM;
28.       int mes_EEPROM;
29.       int diaEEPROM;
30.       int linea1_EEPROM;
31.       RTC_DS3231 rtc;
32.
33.       char meterID[10];
34.
35.       //Configuracion contador
36.       SoftwareSerial swSerSDM;
37.       SDM sdm(swSerSDM, 9600, 0, SWSERIAL_8N1, 2, 16);
38.
39.
40.       // Servidor MQTT
41.       char   SERVER[50]   = "denmark.inverlec.solar"; //
   denmark.inverlec.solar
42.       int    SERVERPORT   = 31010;
43.
44.
45.       WiFiClient espClient;
46.       PubSubClient client(espClient);
47.
48.       // Callback al recibir mensaje por MQTT
49.       void callback(char* topic,
   byte* payload, unsigned int length) {
50.         Serial.print("Message arrived [");
51.         Serial.print(topic);
52.         Serial.print("] ");
53.         for (int i = 0; i < length; i++) {
54.           Serial.print((char)payload[i]);
55.         }
56.         Serial.println();
57.       }
58.
59.       //funcion para mandar a mqqt la lectura cada 10s, 15 min y
   guardar en memoria SD
60.       int denmark() {
61.
62.         char bufout[10];
63.         //Variable JSON que se mandara a mqtt aproximadamente cada
   10 segundos
64.         char t1[15];
65.         char t2 [15];
66.         char t3[15];
67.         DateTime now = rtc.now();
68.         sprintf(t1, "%02d-%02d-%02d %02d:%02d:%02d",  now.year(),
   now.month(), now.day(),   now.hour(), now.minute(),
   now.second());  //Extrae del rtc hora y fecha
69.         sprintf(t2, "%02d-%02d-%02d ",  now.year(), now.month(),
   now.day());  //Extrae del rtc hora y fecha
70.         sprintf(t3, "%02d:%02d:%02d", now.hour(), now.minute(),
   now.second());
```

```
71.
72.         DynamicJsonDocument JSONencoder(1024);
73.         JSONencoder["date"] = t1;//Manda al JSON la variable t
74.         JSONencoder["serialNumber"] = meterID;//numero serial de el
    medidor
75.         delay(50);
76.         JsonObject
    measures = JSONencoder.createNestedObject("measures");//Anidamos al
    JSON la variable measures
77.         float voltage = sdm.readVal(SDM630_VOLTAGE_AVERAGE);
78.         measures["Readings: Volts Average"] = String(voltage);
79.         delay(50);
80.         float voltage_a = sdm.readVal(SDM630_VOLTAGE1);
81.         measures["Readings: Volts A-N"] = String(voltage_a);
82.         delay(50);
83.         float voltage_b = sdm.readVal(SDM630_VOLTAGE2);
84.         measures["Readings: Volts B-N"] = String(voltage_b);
85.         delay(50);
86.         float voltage_c = sdm.readVal(SDM630_VOLTAGE3);
87.         measures["Readings: Volts C-N"] = String(voltage_c);
88.         delay(50);
89.         float current = sdm.readVal(SDM630_CURRENT_AVERAGE);
90.         measures["Readings: I A"] = String(current);
91.         delay(50);
92.         float current_a = sdm.readVal(SDM630_CURRENT1);
93.         measures["Readings: I A"] = String(current_a);
94.         delay(50);
95.         float current_b = sdm.readVal(SDM630_CURRENT2);
96.         measures["Readings: I B"] = String(current_b);
97.         delay(50);
98.         float current_c = sdm.readVal(SDM630_CURRENT3);
99.         measures["Readings: I C"] = String(current_c);
100.        delay(50);
101.        float frequency = sdm.readVal(SDM630_FREQUENCY);
102.        measures["Readings: Frequency"] = String(frequency);
103.        delay(50);
104.        float power = sdm.readVal(SDM630_POWERTOTAL);
105.        measures["Readings: Watts Total"] = String(power);
106.        delay(50);
107.        float power_a = sdm.readVal(SDM630_POWER1);
108.        measures["Readings: Watts A"] = String(power_a);
109.        delay(50);
110.        float power_b = sdm.readVal(SDM630_POWER2);
111.        measures["Readings: Watts B"] = String(power_b);
112.        delay(50);
113.        float power_c = sdm.readVal(SDM630_POWER3);
114.        measures["Readings: Watts C"] = String(power_c);
115.        delay(50);
116.        float received = sdm.readVal(SDM630_IMPORT_ACTIVE_ENERGY)*1
    000;
117.        measures["Scaled Energy: Wh received"] = String(received);
118.        delay(50);
119.        float received_a = sdm.readVal(SDM630_IMPORT1)*1000;
120.        measures["Scaled Energy: Wh received
    A"] = String(received_a);
```

```
121.        delay(50);
122.        float received_b = sdm.readVal(SDM630_IMPORT2)*1000;
123.        measures["Scaled Energy: Wh received
   B"] = String(received_b);
124.        delay(50);
125.        float received_c = sdm.readVal(SDM630_IMPORT3)*1000;
126.        measures["Scaled Energy: Wh received
   C"] = String(received_c);
127.        delay(50);
128.        float delivered = sdm.readVal(SDM630_EXPORT_ACTIVE_ENERGY)*
   1000;
129.        measures["Scaled Energy: Wh
   delivered"] = String(delivered);
130.        delay(50);
131.        float delivered_a = sdm.readVal(SDM630_EXPORT1)*1000;
132.        measures["Scaled Energy: Wh delivered
   A"] = String(delivered_a);
133.        delay(50);
134.        float delivered_b = sdm.readVal(SDM630_EXPORT2)*1000;
135.        measures["Scaled Energy: Wh delivered
   B"] = String(delivered_b);
136.        delay(50);
137.        float delivered_c = sdm.readVal(SDM630_EXPORT3)*1000;
138.        measures["Scaled Energy: Wh delivered
   C"] = String(delivered_c);
139.        delay(50);
140.        float wh_net = (received-delivered);
141.        float wh_net_a = (received_a - delivered_a);
142.        float wh_net_b = (received_b - delivered_b);
143.        float wh_net_c = (received_c - delivered_c);
144.        measures["Scaled Energy: Wh Total"] = String(wh_net) ;
145.        measures["Scaled Energy: Wh Total A"] = String(wh_net_a);
146.        measures["Scaled Energy: Wh Total B"] = String(wh_net_b);
147.        measures["Scaled Energy: Wh Total C"] = String(wh_net_c);
148.        delay(50);
149.        float wh_total = (received + delivered);
150.        measures["Scaled Energy: Wh Net"] = String(wh_total);
151.        delay(50);
152.        float wh_total_a = sdm.readVal(SDM630_TOTAL_ENERGY1)*1000;
153.        measures["Scaled Energy: Wh Net A"] = String(wh_total_a);
154.        delay(50);
155.        float wh_total_b = sdm.readVal(SDM630_TOTAL_ENERGY2)*1000;
156.        measures["Scaled Energy: Wh Net A"] = String(wh_total_b);
157.        delay(50);
158.        float wh_total_c = sdm.readVal(SDM630_TOTAL_ENERGY3)*1000;
159.        measures["Scaled Energy: Wh Net C"] = String(wh_total_c);
160.        delay(50);
161.        char JSONmessageBuffer[1000];
162.        serializeJson(JSONencoder, JSONmessageBuffer);
163.      if (( now.day() == dia_nextEEPROM || diaEEPROM ==0 ||(now.day
   ()>dia_nextEEPROM & mes_nextEEPROM == now.month())) & band_day==1)
164.        {
165.          if (diaEEPROM == 0) {
166.             EEPROM.write(2,2);
167.             diaEEPROM =2;
```

```
168.             EEPROM.commit();
169.           }
170.
171.         dia_next= now.day() + 7;
172.         mes_next = now.month();
173.
174.       //Con este if en el caso de que el dia siguiente sea de otro
   mes hace el arreglo del dia dependiendo del mes actual
175.         if ((now.month()==1 || now.month()==3 || now.month()==5 ||
   now.month()==7 || now.month()==8 || now.month()==10 || now.month()=
   =12) & (dia_next>31)){
176.           dia_next= dia_next -31;
177.           mes_next = now.month() + 1;
178.         }
179.         else   if ((dia_next>30)&(now.month()==4 || now.month()==6
   || now.month()==9 || now.month()==11)){
180.           dia_next= dia_next -30;
181.           mes_next = now.month() + 1;
182.         }
183.         else   if ((dia_next>29)&(now.month()==2 & ((now.year()==20
   20 || now.year()==2024 || now.year()==2028 || now.year()==2032 || n
   ow.year()==2036)))){
184.           dia_next= dia_next -29;
185.           mes_next = now.month() + 1;
186.          }
187.         else if ((dia_next>28)&(now.month()==2)){
188.           dia_next= dia_next -28;
189.           mes_next = now.month() + 1;
190.           }
191.
192.         //Guardando variables EEPROM
193.         EEPROM.write(6,dia_next);
194.         EEPROM.write(0,mes_next);
195.         EEPROM.write(3,now.day());
196.         EEPROM.write(4,now.year()-2000);//si toma mas de tres
   dijitos me escribe otra cosa
197.         EEPROM.write(5,now.month());
198.         EEPROM.write(7,0);//Variable linea 1
199.         dia_EEPROM = now.day();
200.         anho_EEPROM = now.year()-2000;
201.         mes_EEPROM = now.month();
202.         dia_nextEEPROM = dia_next;
203.         mes_nextEEPROM = mes_next;
204.         linea1_EEPROM = 0;
205.         EEPROM.commit();
206.         band_day=2;
207.         }
208.         else   if (( now.day() != dia_nextEEPROM & diaEEPROM !=0) &
    band_day==2)
209.         {
210.         band_day=1;
211.         }
212.
213.
```

```
214.       if ((now.minute()==00 || now.minute()==15 || now.minute()==
   30 || now.minute()==45) & band==1) {
215.     //if ((now.minute()==00 || now.minute()==05 ||
   now.minute()==10 || now.minute()==15 || now.minute()==20 ||
   now.minute()==25 || now.minute()==30 || now.minute()==35 ||
   now.minute()==40 || now.minute()==45 || now.minute()==50 ||
   now.minute()==55) & band==1) {
216.        //Variables que seran guardadas en el logging en formato
   csv
217.       String linea1=("date,serialNumber,Voltaje
   promedio(V),Voltale A-N(V),Voltale B-N(V),Voltale C-N(V),Corriente
   promedio(A),Corriente A(A),Corriente B(A),Corriente C(A),Frecuencia
   (Hz),Potencia total(W),Potencia A(W),Potencia B(W),Potencia
   C(W),Energia Neta total(Wh),Energia Neta A(Wh),Energia Neta
   B(Wh),Energia Neta C(Wh),Energia Importada total(Wh),Energia
   Importada A(Wh),Energia Importada B(Wh),Energia Importada
   C(Wh),Energia Exportada total(Wh),Energia Exportada A(Wh),Energia
   Exportada B(Wh),Energia Exportada C(Wh),Energia Total (Wh),Energia
   Total A(Wh),Energia Total B(Wh),Energia Total (Wh) C");
218.       delay(50);
219.       String
   measurescsv1 = (String(t1) + "," + String(meterID) + ","+ String(vo
   ltage) + "," +  String(voltage_a) + ",");
220.       delay(50);
221.       String
   measurescsv2 = (String(voltage_b) + "," + String(voltage_c) + "," +
    String(current) + "," + String(current_a) + ",");
222.       delay(50);
223.       String
   measurescsv3 = (String(current_b) + "," + String(current_c) + "," +
     String(frequency) + "," +  String(power) + ",");
224.       delay(50);
225.       String
   measurescsv4 = (String(power_a) + "," + String(power_b) + "," + Str
   ing(power_c) + "," +  String(wh_total) + ",");
226.       delay(50);
227.       String
   measurescsv5 = (String(wh_total_a) + "," + String(wh_total_b) + ","
    + String(wh_total_c) + "," + String(received) + ",");
228.       delay(50);
229.       String
   measurescsv6 = (String(received_a) + "," + String(received_b) +  ",
   " + String(received_c) + "," +  String(delivered) + ",");
230.       delay(50);
231.       String
   measurescsv7 = (String(delivered_a) + "," +  String(delivered_b) +
   "," +  String(delivered_c) + "," + String(wh_net) + ",");
232.       delay(50);
233.       String
   measurescsv8 = (String(wh_net_a) + "," + String(wh_net_b) + "," + S
   tring(wh_net_c));
234.       delay(50);
235.       String
   measurescsv = (measurescsv1 + measurescsv2 + measurescsv3 + measure
   scsv4 + measurescsv5 + measurescsv6 + measurescsv7 + measurescsv8);
```

```
236.        Serial.println(measurescsv);
237.        sprintf(t2_EEPROM, "%02d-%02d-
   20%02d.csv",dia_EEPROM,mes_EEPROM,anho_EEPROM);
238.
239.        myFile = SD.open(t2_EEPROM, FILE_WRITE);
240.        // if the file opened okay, write to it:
241.        if (myFile) {
242.          if (linea1_EEPROM ==0) {
243.             EEPROM.write(7,1);     //Solo inicialmente valdra cero
   o cuando inicie una nueva medicion cada 7 dias
244.             linea1_EEPROM = 1;
245.             myFile.println(linea1); //Imprime en el csv la linea 1
246.             EEPROM.commit();
247.          }
248.        myFile.println(measurescsv); //imprime en el csv los
   parametros medidos
249.        // close the file:
250.        myFile.close();
251.        Serial.println("done.");
252.        } else {
253.        // if the file didn't open, print an error:
254.        Serial.println("error opening .csv");
255.        }
256.
257.      //Variable JSON la cual se mandara al mqtt cada 15 minuntos
258.        DynamicJsonDocument var_login(1500);
259.        var_login["date"] = (String(t2) + String(t3));
260.        var_login["serialNumber"] = String(meterID);
261.        JsonArray feeds = var_login.createNestedArray("measures");
262.       /* JsonObject feed1=feeds.createNestedObject();
263.        feed1["tipo"] = "Voltaje";
264.        feed1["fase"] = "Total";
265.        feed1["lectura"] = voltage;
266.        feed1["unidad"] = "V";*/
267.        JsonObject feed1_a=feeds.createNestedObject();
268.        feed1_a["tipo"] = "Voltaje";
269.        feed1_a["fase"] = "A";
270.        feed1_a["lectura"] = String(voltage_a);
271.        //feed1_a["unidad"] = "V";
272.        JsonObject feed1_b=feeds.createNestedObject();
273.        feed1_b["tipo"] = "Voltaje";
274.        feed1_b["fase"] = "B";
275.        feed1_b["lectura"] = String(voltage_b);
276.        //feed1_b["unidad"] = "V";
277.        JsonObject feed1_c=feeds.createNestedObject();
278.        feed1_c["tipo"] = "Voltaje";
279.        feed1_c["fase"] = "C";
280.        feed1_c["lectura"] = String(voltage_c);
281.        //feed1_c["unidad"] = "V";
282.       /* JsonObject feed2=feeds.createNestedObject();
283.        feed2["tipo"] = "Corriente";
284.        feed2["fase"] = "Total";
285.        feed2["lectura"] = current;
286.        feed2["unidad"] = "A";*/
287.        JsonObject feed2_a=feeds.createNestedObject();
```

```
288.        feed2_a["tipo"] = "Corriente";
289.        feed2_a["fase"] = "A";
290.        feed2_a["lectura"] = String(current_a);
291.        //feed2_a["unidad"] = "A";
292.        JsonObject feed2_b=feeds.createNestedObject();
293.        feed2_b["tipo"] = "Corriente";
294.        feed2_b["fase"] = "B";
295.        feed2_b["lectura"] = String(current_b);
296.        //feed2_b["unidad"] = "A";
297.        JsonObject feed2_c=feeds.createNestedObject();
298.        feed2_c["tipo"] = "Corriente";
299.        feed2_c["fase"] = "C";
300.        feed2_c["lectura"] = String(current_c);
301.        //feed2_c["unidad"] = "A";
302.        JsonObject feed3=feeds.createNestedObject();
303.        feed3["tipo"] = "Frecuencia";
304.        feed3["fase"] = "Total";
305.        feed3["lectura"] = String(frequency);
306.        //feed3["unidad"] = "Hz";
307.       /* JsonObject feed4=feeds.createNestedObject();
308.        feed4["tipo"] = "Potencia";
309.        feed4["fase"] = "Total";
310.        feed4["lectura"] = power;
311.        feed4["unidad"] = "W";*/
312.        JsonObject feed4_a=feeds.createNestedObject();
313.        feed4_a["tipo"] = "Potencia";
314.        feed4_a["fase"] = "A";
315.        feed4_a["lectura"] = String(power_a);
316.        //feed4_a["unidad"] = "W";
317.        JsonObject feed4_b=feeds.createNestedObject();
318.        feed4_b["tipo"] = "Potencia";
319.        feed4_b["fase"] = "B";
320.        feed4_b["lectura"] = String(power_b);
321.        //feed4_b["unidad"] = "W";
322.        JsonObject feed4_c=feeds.createNestedObject();
323.        feed4_c["tipo"] = "Potencia";
324.        feed4_c["fase"] = "C";
325.        feed4_c["lectura"] = String(power_c);
326.        //feed4_c["unidad"] = "W";
327.        JsonObject feed5=feeds.createNestedObject();
328.        feed5["tipo"] = "Energy: Wh Total";
329.        feed5["fase"] = "Total";
330.        feed5["lectura"] = String(wh_total);
331.        //feed5["unidad"] = "Wh";
332.        JsonObject feed5_a=feeds.createNestedObject();
333.        feed5_a["tipo"] = "Energy: Wh Total";
334.        feed5_a["fase"] = "A";
335.        feed5_a["lectura"] = String(wh_total_a);
336.        //feed5_a["unidad"] = "Wh";
337.        JsonObject feed5_b=feeds.createNestedObject();
338.        feed5_b["tipo"] = "Energy: Wh Total";
339.        feed5_b["fase"] = "B";
340.        feed5_b["lectura"] = String(wh_total_b);
341.        //feed5_b["unidad"] = "Wh";
342.        JsonObject feed5_c=feeds.createNestedObject();
```

```
343.        feed5_c["tipo"] = "Energy: Wh Total";
344.        feed5_c["fase"] = "C";
345.        feed5_c["lectura"] = String(wh_total_c);
346.        //feed5_c["unidad"] = "Wh";
347.        /*JsonObject feed6=feeds.createNestedObject();
348.        feed6["tipo"] = "Energy: Wh received";
349.        feed6["fase"] = "Total";
350.        feed6["lectura"] = received;
351.        feed6["unidad"] = "Wh";*/
352.        JsonObject feed6_a=feeds.createNestedObject();
353.        feed6_a["tipo"] = "Energy: Wh received";
354.        feed6_a["fase"] = "A";
355.        feed6_a["lectura"] = String(received_a);
356.        //feed6_a["unidad"] = "Wh";
357.        JsonObject feed6_b=feeds.createNestedObject();
358.        feed6_b["tipo"] = "Energy: Wh received";
359.        feed6_b["fase"] = "B";
360.        feed6_b["lectura"] = String(received_b);
361.        //feed6_b["unidad"] = "Wh";
362.        JsonObject feed6_c=feeds.createNestedObject();
363.        feed6_c["tipo"] = "Energy: Wh received";
364.        feed6_c["fase"] = "C";
365.        feed6_c["lectura"] = String(received_c);
366.        //feed6_c["unidad"] = "Wh";
367.        /*JsonObject feed7=feeds.createNestedObject();
368.        feed7["tipo"] = "Energy: Wh delivered";
369.        feed7["fase"] = "Total";
370.        feed7["lectura"] = delivered;
371.        feed7["unidad"] = "Wh";*/
372.        JsonObject feed7_a=feeds.createNestedObject();
373.        feed7_a["tipo"] = "Energy: Wh delivered";
374.        feed7_a["fase"] = "A";
375.        feed7_a["lectura"] = String(delivered_a);
376.        //feed7_a["unidad"] = "Wh";
377.        JsonObject feed7_b=feeds.createNestedObject();
378.        feed7_b["tipo"] = "Energy: Wh delivered";
379.        feed7_b["fase"] = "B";
380.        feed7_b["lectura"] = String(delivered_b);
381.        //feed7_b["unidad"] = "Wh";
382.        JsonObject feed7_c=feeds.createNestedObject();
383.        feed7_c["tipo"] = "Energy: Wh delivered";
384.        feed7_c["fase"] = "C";
385.        feed7_c["lectura"] = String(delivered_c);
386.        //feed7_c["unidad"] = "Wh";
387.
388.        char JSONmessageBuffer2[1300];
389.        serializeJson(var_login, JSONmessageBuffer2);
390.        Serial.println(JSONmessageBuffer2);
391.        char topic2[20];
392.        sprintf(topic2,"denmark/log/%s",meterID);
393.        Serial.println(topic2);
394.        if (client.publish(topic2,
    JSONmessageBuffer2) == true) { //update when server is changed
395.
396.        Serial.println("Success sending message");
```

```
397.        } else {
398.        Serial.println("Error sending message");
399.        }
400.        band = 2;//Me limita que solo la primera medicion tomada en
   los minutos 0, 15, 30, 45 sean mandados al mqtt y al logging
401.
402.      }
403.        else if ((now.minute()!=00 & now.minute()!=15 & now.minute(
   )!=30 & now.minute()!=45) & band==2)
404.        //else   if ((now.minute()!=00 & now.minute()!=05 &
   now.minute()!=10 & now.minute()!=15 & now.minute()!=20 &
   now.minute()!=25 & now.minute()!=30 & now.minute()!=35 &
   now.minute()!=40 & now.minute()!=45 & now.minute()!=50 &
   now.minute()!=55) & band==2)
405.        {
406.        band=1;
407.        }
408.        char topic1[10];
409.        sprintf(topic1,"denmark/tr/%s",meterID);
410.        Serial.println(topic1);
411.        if (client.publish(topic1,
   JSONmessageBuffer) == true) { //update when server is changed
412.        Serial.println("Success sending message");
413.        } else {
414.        Serial.println("Error sending message");
415.        }
416.        Serial.print("JSON: ");
417.        Serial.println(JSONmessageBuffer);
418.
419.      }
420.
421.
422.      //Reconeccion al servidor de mqtt
423.      void reconnect() {
424.        // Loop until we're reconnected
425.       //  while (!client.connected()) { //Creo que si comentareo
   este while en lugar de quedarse probando que conecte solo lo haria
   una vez
426.          Serial.print("Attempting MQTT connection...");
427.          // Create a random client ID
428.          String clientId = "ESP8266Client";
429.
430.          // Attempt to connect
431.          if (client.connect(clientId.c_str())) {
432.            Serial.println("connected");// Mostrar mensaje de exito
   a MQTT
433.            // Once connected, publish an announcement...
434.            StaticJsonDocument<300> JSONEncoder;
435.            JSONEncoder['status'] = "true";
436.            char JSONmessageBuffer[100];
437.            serializeJson(JSONEncoder, JSONmessageBuffer);
438.            //client.publish("testTopic", JSONmessageBuffer);
439.            //client.publish("testTopic2", JSONmessageBuffer);
440.            // ... and resubscribe
441.            //client.subscribe("inTopic");
```

```
442.         } else {
443.           Serial.print("Conexión MQTT Falló, rc=");// Mostrar
   mensaje de fallo a MQTT
444.           Serial.print(client.state());
445.           Serial.println(" try again in 5 seconds");
446.           // Wait 5 seconds before retrying
447.           delay(5000);
448.         }
449.       //}
450.     }
451.
452.
453.     void setup() {
454.       // put your setup code here, to run once:
455.
456.       Serial.begin(115200);
457.       //initialize serial
458.       EEPROM.begin(512);
459.       Wire.begin();
460.       rtc.begin();
461.       if (! rtc.begin()) {
462.         Serial.println("Couldn't find RTC");
463.         while (1);
464.       }
465.
466.       if (rtc.lostPower()) {
467.         Serial.println("RTC lost power, lets set the time!");
468.         // following line sets the RTC to the date & time this
   sketch was compiled
469.         rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
470.       }
471.
472.       sdm.begin();
473.       delay(10);
474.
475.       WiFiManagerParameter custom_output("Meter ID", "Meter ID",
   meterID, 10);
476.
477.       //WiFiManager
478.       //Local intialization. Once its business is done, there is
   no need to keep it around
479.       WiFiManager wifiManager;
480.
481.       //reset settings - for testing
482.       //wifiManager.resetSettings();
483.
484.     /*
485.       //start-block2
486.       IPAddress _ip = IPAddress(192, 168, 98, 164);
487.       IPAddress _gw = IPAddress(192, 168, 98, 254);
488.       IPAddress _sn = IPAddress(255, 255, 255, 0);
489.       //end-block2
490.
491.       wifiManager.setSTAStaticIPConfig(_ip, _gw, _sn);
492.     */
```

```
493.      wifiManager.addParameter(&custom_output);
494.       //tries to connect to last known settings
495.       //if it does not connect it starts an access point with the
   specified name
496.       //here  "AutoConnectAP" with password "password"
497.       //and goes into a blocking loop awaiting configuration
498.
499.
500.       if (!wifiManager.autoConnect("DENMARK", "123456789")) {
501.         Serial.println("failed to connect, we should reset as see
   if it connects");
502.         delay(3000);
503.         ESP.reset();
504.         delay(5000);
505.       }
506.
507.       //if you get here you have connected to the WiFi
508.       Serial.println("connected :)");
509.
510.     //Guarda la variable IDMeter
511.       if (EEPROM.read(8) == 0) {
512.         strcpy(meterID, custom_output.getValue());//extrae el
   valor ingresado en la app
513.         EEPROM.write(8,1); //band para que no cambie el valor de
   IDMeter
514.         EEPROM.put(9,meterID); //guarda el valor de IDMeter y no
   se puede cambiar a menos que se borren las variable eeprom
515.         EEPROM.commit();
516.       }
517.
518.       Serial.println("local ip");
519.       Serial.println(WiFi.localIP());
520.
521.       delay(1000);
522.       Serial.println("Creando cliente MQTT...");
523.       client.setServer("denmark.inverlec.solar", 31010);
524.       client.setCallback(callback);
525.
526.       if (!SD.begin(10)) {
527.         Serial.println("initialization SD card failed!");
528.         return;
529.       }
530.       else {
531.       Serial.println("initialization  SD card done.");
532.       }
533.
534.     ///Set variables en EEPROM
535.       EEPROM.get(9,meterID);//
536.       diaEEPROM = EEPROM.read(2);// define que ya fue corrida al
   menos una vez el programa
537.       dia_EEPROM = EEPROM.read(3);//variable de dia de la carpeta
   actual csv
538.       anho_EEPROM = EEPROM.read(4);//variable de año de la
   carpeta actual csv
```

```
539.        mes_EEPROM = EEPROM.read(5);  //variable de mes de la
   carpeta actual csv
540.        dia_nextEEPROM = EEPROM.read(6);//variable la cual contiene
   el siguiente dia en cual se creara la carpeta csv
541.        linea1_EEPROM = EEPROM.read(7);//Esta variable es para
   limitar que el encabezado se mande solo una vez al crar el csv
542.        mes_nextEEPROM = EEPROM.read(0);//variable la cual contiene
   el siguiente mes en cual se creara la carpeta csv
543.        /*Serial.println(mes_nextEEPROM);
544.        Serial.println(meterID);
545.        Serial.println(dia_nextEEPROM);
546.        Serial.println(anho_EEPROM);
547.        Serial.println(mes_EEPROM);
548.      */
549.      }
550.
551.      void loop() {
552.
553.        //Loop reconexion a Mqtt
554.        {
555.        if (!client.connected()) {
556.        reconnect();
557.        }
558.        client.loop();
559.        }
560.
561.        denmark();
562.        delay(7920);
563.      }
```