Description du logiciel.

Elle est destinée à utiliser l'excédent de courant électrique comme puissance de chauffage électrique dans le ménage.
Comme environnement de programmation, nous utilisons Visual Studio code avec la plate-forme Addon IO.

Les valeurs peuvent être lues dans 2 modes différents.

- MQTT
- Modbus

Le logiciel utilise 3 sorties.

```
        // #define BUILTIN_LED 2
        #define Alarm 27
        #define SSR_1 26
        // #define SSR_1 13
        #define SSR_2 22
```

A la sortie 27, il est possible de raccorder une LED ou un bip (avertisseur sonore).
Aux sorties 26 et 22, il est possible d'émettre un signal PWM ou de commutation, selon le réglage.

Pour être flexible, il est possible de paramétrer et d'expérimenter via une page web intégrée.
Le site Web est l'adresse IP de l'ESP32.

Le site dispose de 4 groupes d'entrée. qui correspondent aux sorties mentionnées ci-dessus.

//********** SSR-1 **********
//********** SSR-2 **********

//********** activate Boiler, heater General Purpose**********//

Il n'y a rien d'actif. est prévu pour les versions futures.

/********** Values for test **********

Page Web intégrée

Voici une description de ce que signifient les différents champs.

- XXX_1 mode

```
0=digital 1=PWM mapped 2=Pwm  PID
```

0= Digital Ein /Aus
1= PWM mapped
2= PWM PID

PWM mapped signifie ceci.

J'ai par exemple une puissance solaire maximale de 600 watts.
Mais ma tige de chauffage tire 2 kW

La puissance solaire peut donc être comprise entre 0 et 600 watts.
Par exemple, si j'ai défini un PWM 8 bits, les valeurs possibles sont 0-255
Mapped convertit les valeurs 0-600 en valeurs 0-255 linéaires.

Je dois maintenant trouver en débogage quelle valeur correspond à mes 600 watts.

Par exemple, 600 watts sur le compteur de consommation correspondent à la valeur PWM de 100.
Puis je mappe le 0-600 watts à 0-100 PWM

PWM PID signifie que la valeur initiale est calculée par le contrôleur PID.

- XXX_1 input select

```
0 = no input
```

```
1 = power_excess_solar_calc_W
2 = power_excess_solar_l1_calc_W
3 = power_excess_solar_l2_calc_W
4 = power_excess_solar_l3_calc_W
5 = production Surplus L1
6 = production Surplus L2
7 = production Surplus L3
8 = production Surplus L1_L2_L3
9 = production Surplus L1_L2
10 = production Surplus L1_L3
11 = production Surplus L2_L3
12 = test_val_1
13 = test_val_2
```

1 -4 Ici, les valeurs sont lues par le lecteur Smarty via MQTT.
5-11 lit les valeurs directement à partir de l'onduleur via Modbus TCP.
5-7 une seule phase est utilisée pour le calcul.
8-11 utilise la somme de chaque phase ou onduleur.
12 et 13 sont des valeurs de test que vous pouvez entrer et simuler
par exemple test_val_1 250 signifie simuler une puissance photovoltaïque de 250 watts
• Fréquence PWM Fréquence PWM. par exemple 1000HZ
Résolution PWM par exemple 8 pour 8 bits signifie 2 haute 8 ou 0-255

• XXX_1_on valeur d'allumage en watts
• XXX_1_off Valeur d'arrêt en watts

- • Setpoint
  Valeur que le PID doit contenir, par exemple 30 au-dessus du Smartyreader.
  Cela signifie qu'une alimentation de 30 watts est régulée automatiquement.

  Cependant, parce que les valeurs sont lentes à lire. Pendant au moins 10 secondes avec le Smarty Reader, le régulateur PID 2 a été programmé par étapes.

  Ça veut dire.
  on a 2 ensembles de constantes pour la valeur proportionnelle, intégrale et différentielle.
  aggxx sont les valeurs agressives

  la commutation fonctionne par
  Setpoint distance PID reaction speed switch ex. 50
  Le comportement est alors le suivant.

  Par exemple, je veux faire une alimentation à zéro. setpoint = 30

  le Smarty m'indique par exemple 100 watts. maintenant, les valeurs agressives sont utilisées
  Le régulateur PID agressif augmente la puissance et l'alimentation est moindre.

  Si on ne m'injecte que 79 watts (= < 30+50), le régulateur PID passe aux valeurs non agressives.
  On s'approche lentement de l'alimentation 30.

  • SSR_1 PID direction
  donne la direction de la règle
  ```
  0 direct (default) or 1 reverse
  ```

  0 Standard 1 invers

Comment ajuster le code du programme.

une partie est exécutée dans special_settings. h.

Adaptation 1 pour lire le Smarty Reader.

```
// MQTT
const char *MQTT_SERVER = "192.168.1.80";
const char *MQTT_CLIENT_ID = "Smarty_esp_Voltaik";
const char *MQTT_OUT_TOPIC = "Voltaik_Surplus";
const short MQTT_PORT = 1883; // TLS=8883
const char *MQTT_IN_TOPIC = "#";


// Smarty

const char *Subscription_will = "smarty-1/lastwill/onlinestatus";
const char *Subscription_1 = "smarty-1/power_excess_solar_calc_W";
const char *Subscription_2 = "smarty-1/power_excess_solar_l1_calc_W";
const char *Subscription_3 = "smarty-1/power_excess_solar_l2_calc_W";
const char *Subscription_4 = "smarty-1/power_excess_solar_l3_calc_W";
```

ce sont les positions 1-4 dans l'input select.

Personnalisation 2 Modbus TCP

```
// Modbus Server z.b Kostal (R)

IPAddress Modbus_ip = {192, 168, 178, 97}; // IP address of modbus server
uint16_t Modbus_port = 502;                // port of modbus server

#define Modbus_on_off 0 // switch on (1)or  of (0)

const int Modbus_ID_1 = 1; // Kostal Modbus ID
const int Modbus_ID_2 = 2; //  Modbus ID
const int Modbus_ID_3 = 3; //  Modbus ID
const int Modbus_ID_4 = 4; //  Modbus ID

int Adresse_Modbus_Register_1 = 30017;//Kostal Modbus Adresse Watt L1
int Adresse_Modbus_Register_2 = 30021;//Kostal Modbus Adresse Watt L2
int Adresse_Modbus_Register_3 = 30025;//Kostal Modbus Adresse Watt L3
int Adresse_Modbus_Register_4  = 30030;//Kostal Modbus Adresse Watt Total
```

Les adresses des onglets Modbus sont codées en dur.

```
  error = MB.addRequest((uint32_t)lastMillis, Modbus_ID_1, READ_HOLD_REGISTER, 40084, 1);
  error = MB.addRequest((uint32_t)lastMillis + 1, Modbus_ID_2, READ_HOLD_REGISTER, 40084, 1);
  error = MB.addRequest((uint32_t)lastMillis + 2, Modbus_ID_3, READ_HOLD_REGISTER, 40084, 1);
  if (error != SUCCESS)
```

Ils sont compatibles avec tous les onduleurs aver intertface Modbus tcp , par examples les micro-onduleurs d'AP systems.

Les adresses se trouvent dans le registre Modbus Description de l'onduleur.

ID1, ID2 et ID3 correspondent aux onduleurs individuels ou aux phases L1, L2 et L3 des micro-onduleurs 1phasés.

Debug

Pour rendre le débogage plus facile, vous pouvez simplement définir ce qui doit être débogué Les commutateurs sont dans special_settings. h

```
#define DEBUG 0
#define DEBUGf 0
#define MQTT_DEBUG 1
#define SMARTY_DEBUG 1
#define MODBUS_DEBUG 1
#define DAC_DEBUG 1
#define PID_DEBUG 1
#define SSR_DEBUG 0
#define PWM_DEBUG 1
#define I2C_DEBUG 0
#define DFPLAYER_DEBUG 0
```

Comment puis-je spécifier des valeurs connues lorsque je programme l'ESP 32.

C'est très simple.
La page web enregistre l'entrée dans des fichiers xxxx. txt individuels. Entrez simplement la valeur connue dans le champ de texte.
Ces fichiers se trouvent dans le dossier Data.

Les bogues existants.

Le code est un peu plus ancien.
Au moment où il a été écrit, SPIFFS est passé à littleFS.
https://github. com/lorol/LITTLEFS

il y a un problème avec le string processor lignes 434 et 1110

This library is now part of Arduino esp32 core v2
Note, there it is renamed from LITTLEFS to LittleFS, Please post your issues there. This here is kept for Arduino esp32 core 1.x purposes

Espressif (https://www. espressif. com/) a adopté le code

il est possible d'utiliser les deux, mais la nouvelle version intégrée devrait être utilisée. Lignes 39 et 40

```
//#include "LITTLEFS.h" // old version
#include "LittleFS.h"  // now included in ESP32
```

```
/*
https://github.com/lorol/LITTLEFS/issues/43
LITTLEFSimpl::exists() bug with Platformio (ESP32 arduino framework) in Visual Studio Code
Fix by adding third argument
File f = open(path, "r", false);
line 44 in LITTLEFS.cpp

also found
https://github.com/espressif/arduino-esp32/pull/6179
With LittleFS the `fs.exists(path)` returns true also on folders. A `isDirectory()` call is required to set _isFile to false on directories.
This enables serving all files from a folder like : `server->serveStatic("/", LittleFS, "/", cacheHeader.c_str());
     File f = fs.open(path);
     _isFile = (f && (! f.isDirectory()));
line 1222
*/
```

Dans la page Web, les valeurs définies devraient être affichées après une nouvelle consultation de la page Web.
Cela ne fonctionne pas. il faudrait aller au fond de cette erreur.
Le logiciel est opérationnel et fait son service.

Une lecture de température de la mémoire est prévue pour réaliser un arrêt automatique.

```
const char *Subscription_6 = "Warmwater-Mer/upper_temp";
```

des sites web supplémentaires sont prévus pour les futurs élargissements. par exemple enregistrement des courbes PID, etc.

Dans le platformio.ini est fourni la sortie de langue mp3.
dfrobot/DFRobotDFPlayerMini@^1.0.5
il fonctionne via un module DFR0299 text to speech avec MQTT est possible.

Sécurité intégrée MQTT.

Le logiciel Smartyreader de Weigu https://github.com/weigu1/SmartyReader a été ajouté:
Un testament a été incorporé
on s'est assuré que le lecteur Smarty n'affiche pas de valeurs avec l'option retained.
veuillez utiliser uniquement le logiciel SmartyReader personnalisé.

Dans la ligne 752, le testament est lu dans la variable var_Will et dans la ligne 806, les valeurs lues sont obligatoirement mises à 0.

```
    if (var_Will == 0)
    {
      var_Einsp_tot = 0.0;
      var_Einsp_L1 = 0.0;
      var_Einsp_L2 = 0.0;
      var_Einsp_L3 = 0.0;
    }
```

Je veux lire plus de données sur MQTT par ex. température de chaudière, rayonnement solaire, etc.

specialsettings.h et le code de programme main.h doivent être adaptés.
c'est-à-dire que les masques doivent être supprimés.

Les données MQTT à lire sont définies dans specialsettings.h.
Le lien MQTT se situe entre les "....."

```
 // Boiler

//const char *Subscription_6 = "Warmwater-Mer/upper_temp";
/*
const char *Subscription_7 = "Warmwater-Mer/lower_temp";
const char *Subscription_8 = "Warmwater-Mer/max_temp";
const char *Subscription_9 = "Warmwater-Mer/test_1";
const char *Subscription_10 = "Warmwater-Mer/test_2";
const char *Subscription_11 = "Warmwater-Mer/Test_3";
*/
```

Par exemple, si je veux lire la température du chauffe-eau en haut, je supprime // au début de la ligne

```
const char *Subscription_6 = "Warmwater-Mer/upper_temp";
```

sont prévus dans le code de programme 2 chaudières (mémoire) avec 2 points de mesure ainsi que la température maximale.
cette température peut être effectuée en envoyant un message MQTT.
Ceci est nécessaire, par exemple, si j'ai aussi de l'énergie solaire thermique.
Ça ne sert à rien si je chauffe le stockage électriquement et que je dois ensuite éteindre le solaire thermique à cause de la surchauffe.

Masquer dans le code du programme

Ils sont masqués à 2 endroits dans le code du programme.

```
MQTT_Client.subscribe(Subscription_will);
 MQTT_Client.subscribe(Subscription_1);
 MQTT_Client.subscribe(Subscription_2);
 MQTT_Client.subscribe(Subscription_3);
 MQTT_Client.subscribe(Subscription_4);
 MQTT_Client.subscribe(Subscription_5);
 /* MQTT_Client.subscribe(Subscription_6);
  MQTT_Client.subscribe(Subscription_7);
  MQTT_Client.subscribe(Subscription_8);
  MQTT_Client.subscribe(Subscription_9);
  MQTT_Client.subscribe(Subscription_10);
  MQTT_Client.subscribe(Subscription_11);
  */
```

Les abonnements 6 à 11 sont masqués.
Lignes 758-764 du code.
en masquant les groupes /* à */
et aux lignes 908 à 1008

```
 /*
 if (strcmp(topic, Subscription_6) == 0)
 {
  if (msg_upper_1 != 1)
  {
   char buff_p[length];
   for (int i = 0; i < length; i++)
   {
    buff_p[i] = (char)payload[i];
   }
   buff_p[length] = '\0';
   String msg_p = String(buff_p);
   var_upper_temp_1 = msg_p.toFloat(); // to float
   msg_upper_1 = 1;
  }
 }
```

Comment puis-je attribuer de nouveaux noms à msg.

dans l'exemple ci-dessus, j'ai lu la température supérieure de la chaudière et lui ai donné le nom `msg_upper_1 = 1;`

Par exemple, je veux lui donner le nom `msg_test_1 = 1;` .
alors je dois aussi entrer le nom à partir de la ligne 377 dans le code.

```
int msg_test_1 = 0;
```