mise à jour du projet #4 : compteur d'énergie ESP32

Surveiller l'énergie avec MQTT

Saad Imtiaz (Elektor)

Précédemment, nous nous sommes concentrés sur la configuration du compteur d'énergie basé sur ESP32 et sur son intégration dans Home Assistant. Nous avons aussi évoqué les possibilités offertes par la puce ESP32-S3 pour des fonctionnalités d'IA et de ML qui permettraient de prédire les modèles de consommation d'énergie et d'identifier les appareils. Dans cette mise à jour, nous franchissons une nouvelle étape en déployant un micrologiciel qui permet la surveillance de l'énergie en temps réel via MQTT, jetant les bases pour des fonctionnalités plus avancées.

Nous avons déjà discuté en détail du parcours complet du compteur d'énergie ESP32, depuis l'assemblage initial jusqu'à son installation finale dans le boîtier de disjoncteurs. Ce processus a inclus la configuration du compteur d'énergie avec ESPHome et Home Assistant, sa calibration et les tests, et enfin l'installation dans le boîtier de disjoncteurs. Dans la dernière mise à jour [1], nous avons entrepris d'intégrer pleinement ce système dans Home Assistant. Notre objectif ambitieux est désormais de développer des fonctionnalités avancées d'intelligence artificielle (IA) et d'apprentissage automatique (ML) afin de prévoir les tendances de consommation énergétique et d'identifier les appareils électriques à partir de leur signature énergétique

Bien que l'intégration AI/ML soit toujours en phase de développement, en raison de la préparation approfondie des données requises, cette mise à jour intermédiaire met en lumière une avancée significative : la mise en place de la surveillance énergétique en temps réel via MQTT. MQTT est un protocole de messagerie léger qui est spécialement conçu pour faciliter une communication rapide et efficace. Pour plus de détails sur MQTT, nous vous invitons à consulter l'encadré **« Qu'est-ce que MQTT ? »**.

Micrologiciel personnalisé et MQTT

Dans cet article, nous aborderons l'étape suivante de notre projet - exploiter MQTT et l'EDI Arduino pour permettre la gestion de l'énergie en temps réel. Nous détaillerons le développement du micrologiciel qui permet à l'ESP32 de communiquer avec un courtier MQTT, envoyant des données énergétiques à un serveur Home Assistant ou à toute autre plateforme compatible MQTT. L'utilisation de MQTT avec un micrologiciel individuel

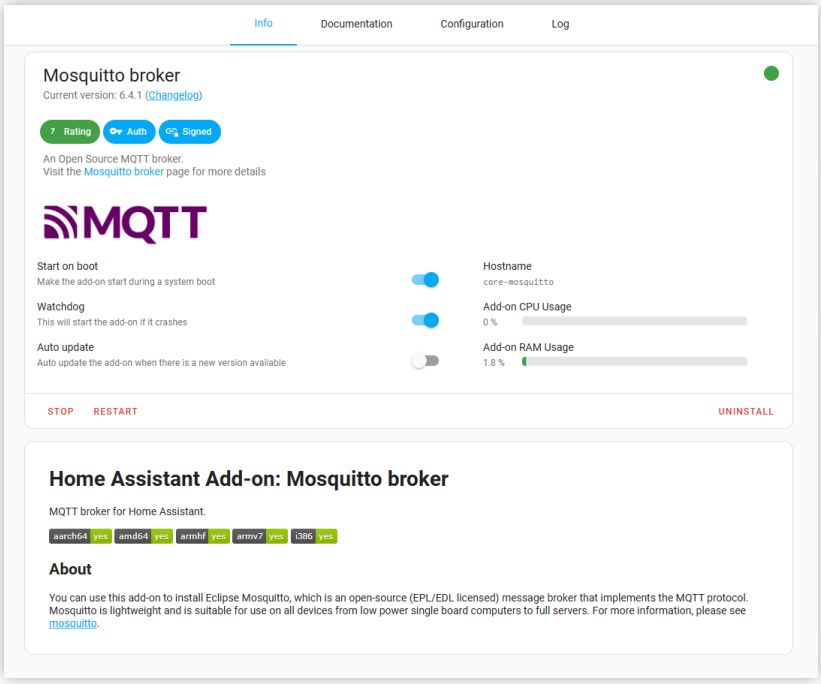
Qu'est-ce que MQTT ?

MQTT est un protocole de messagerie léger conçu pour une communication efficace entre les appareils, en particulier dans les environnements IdO. Ce système repose essentiellement sur un courtier MQTT, un serveur qui agit comme un hub pour l'échange de messages. Le courtier reçoit les messages des appareils, appelés éditeurs, et les achemine vers les destinataires appropriés, appelés abonnés, sur la base d'un système de sujets. Dans MQTT, un sujet (topic) est une chaîne de caractères qui catégorise les messages, servant de canal pour la publication d'informations, tandis qu'un abonné est un appareil ou une application qui suit des sujets spécifiques pour recevoir ces messages. Par exemple, dans une maison intelligente, un sujet tel que `home/energy/voltage` pourrait être utilisé pour diffuser des relevés de tension que recevrait et afficherait un tableau de bord, permettant une surveillance en temps réel.

Le rôle du courtier est de s'assurer que les messages sont transmis de manière efficace et sécurisée, même sur des réseaux peu fiables. Dans les applications IdO, le courtier MQTT est essentiel pour gérer l'échange de données entre les capteurs, les appareils et les systèmes (qui sont les clients MQTT), permettant ainsi la surveillance, le contrôle et l'automatisation en temps réel.

Listage 1. Micrologiciel (extrait).

```
#include <WiFi.h>
#include <SPI.h>
#include <ATM90E32.h>
#include <MQTTPubSubClient.h>
#include <config.ino> // Include configuration file for WiFi and MQTT details

// WiFi Credentials
const char* ssid = WIFISSID;       // Your WiFi SSID
const char* pass = WIFIPASSWORD;   // Your WiFi Password

WiFiClient client;
MQTTPubSubClient mqtt;

ATM90E32 energymeter{};

void setup() {

...

  /* Initialize the ATM90E32 energy meter with the specified parameters */
  energymeter.begin(CS_PIN, LINEFREQ, PGA_GAIN, VOLTAGE_GAIN, GAIN_CT_A, GAIN_CT_B, GAIN_CT_C);

...

  /* Begin the WiFi connection using the provided SSID and password */
  WiFi.begin(ssid, pass);

  /* Initialize the MQTT client */
  mqtt.begin(client);

  /* Connect to WiFi, MQTT broker, and Home Assistant */
  connect();

...
 }

void loop() {
  /* Keep the MQTT client updated */
  mqtt.update();

  /* Reconnect to the MQTT broker if the connection is lost */
  if (!mqtt.isConnected()) {
    connect();
  }

  /* Check and send energy data to Home Assistant every 3 seconds */
  static uint32_t prev_ms = millis();
  if (millis() > prev_ms + 3000) {
    prev_ms = millis();
    getEnergyData(); // Retrieve energy data and send via MQTT
  }
}

void getEnergyData() {

  // Retrieve system status from the ATM90E32
  unsigned short sys0 = energymeter.GetSysStatus0();   //EMMState0
  unsigned short sys1 = energymeter.GetSysStatus1();   //EMMState1
  unsigned short en0 = energymeter.GetMeterStatus0();  //EMMIntState0
  unsigned short en1 = energymeter.GetMeterStatus1();  //EMMIntState1

  // Print system and meter status for debugging
  Serial.println("Sys Status: S0:0x" + String(sys0, HEX) + " S1:0x" + String(sys1, HEX));
  Serial.println("Meter Status: E0:0x" + String(en0, HEX) + " E1:0x" + String(en1, HEX));
  delay(10);

  // Check if the MCU is not receiving data from the energy meter
  if (sys0 == 65535 || sys0 == 0) Serial.println("Error: Not receiving data
                                                  from energy meter - check your connections");

  // Retrieve all parameters from the ATM90E32
  float lineVoltageA = energymeter.GetLineVoltageA();
  float lineVoltageB = energymeter.GetLineVoltageB();
  float lineVoltageC = energymeter.GetLineVoltageC();
       ...

  // Send all the collected energy data via MQTT to Home Assistant
  mqtt.publish("esp32energymeter/lineCurrentA", String(lineCurrentA).c_str());
  mqtt.publish("esp32energymeter/lineCurrentB", String(lineCurrentB).c_str());
  mqtt.publish("esp32energymeter/lineCurrentC", String(lineCurrentC).c_str());
  mqtt.publish("esp32energymeter/totalCurrent", String(totalCurrent).c_str());

...
}
```



▲

Figure 1. Options de configuration du courtier MQTT dans Home Assistant, y compris l'option ***Start on boot*** *pour assurer un démarrage automatique.*

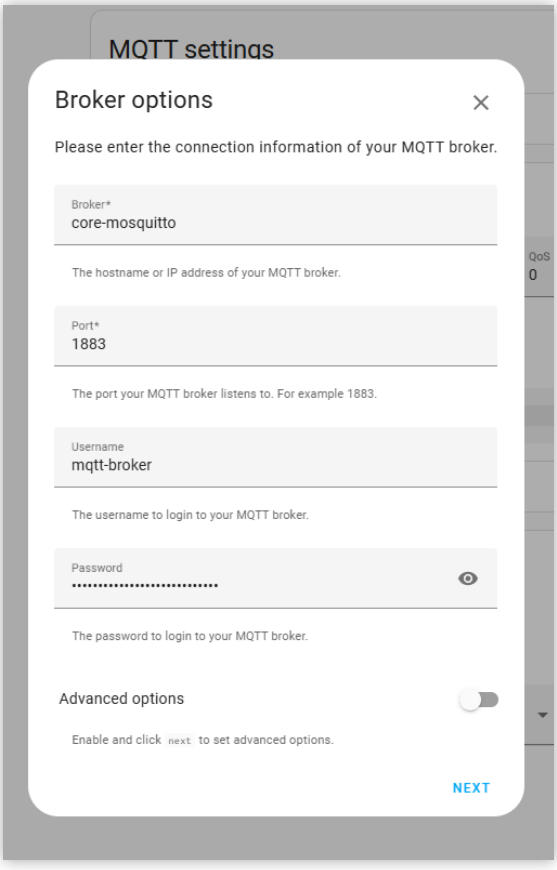


Figure 2. Configuration de l'intégration MQTT dans Home Assistant avec l'adresse IP, le port et les informations d'identification de l'utilisateur.

▶

plutôt qu'avec ESPHome offre beaucoup plus de flexibilité en termes d'intégration et de personnalisation. Avec un micrologiciel personnalisé, vous avez un contrôle complet sur la collecte, le traitement et la transmission des données, ce qui vous permet d'adapter le système aux besoins spécifiques du projet. Ce niveau de personnalisation est particulièrement bénéfique pour les applications complexes nécessitant une optimisation des performances, l'intégration de matériel non standard, ou l'implémentation de fonctionnalités avancées telles que l'IA et les algorithmes d'apprentissage automatique. De plus, les micrologiciels personnalisés facilitent l'intégration avec diverses plateformes au-delà de Home Assistant, telles que les services basés sur le cloud, les tableaux de bord personnalisés et d'autres systèmes IdO. Vous pouvez également mettre en œuvre des mesures de sécurité plus renforcées, telles que des protocoles de cryptage spécifiques ou des mécanismes d'authentification avancés, assurant ainsi une protection accrue des données à travers le réseau. En outre, il est possible d'optimiser ces micrologiciels pour des cas d'utilisation spécifiques, réduisant les frais généraux et améliorant l'efficacité du système, un atout majeur dans les environnements aux ressources limitées.

Logiciel

Le micrologiciel est conçu pour permettre de connecter l'ESP32 à un réseau Wifi et d'utiliser MQTT pour la communication, facilitant l'envoi des données énergétiques à un serveur Home Assistant à des fins de surveillance et d'automatisation. Le **listage 1**, présente un extrait du code. Pour accéder au code complet et aux fichiers associés, consultez le dépôt GitHub du projet [2].

Le projet repose sur deux bibliothèques fondamentales : la bibliothèque *ATM90E32*, développée par CircuitSetup [3], qui gère la communication avec le circuit intégré de mesure de l'énergie, et la bibliothèque *MQTTPubSubClient* [4], qui gère la communication en tant que client MQTT. La bibliothèque *ATM90E32* est essentielle pour la collecte des données provenant de la puce du compteur d'énergie, incluant des paramètres tels que la tension, le courant et la puissance. Lors du développement, l'intégration de MQTT avec une authentification par nom d'utilisateur et mot de passe a représenté un défi majeur. Bien que de nombreuses bibliothèques MQTT assurent les fonctions de base, *MQTTPubSubClient* s'est révélée être l'une des rares à offrir une compatibilité avec l'ESP32 tout en supportant les fonctions d'authentification nécessaires. Le logiciel commence en incluant les bibliothèques essentielles pour la connectivité réseau, la communication SPI, l'interface avec le circuit intégré du compteur d'énergie et à la gestion de la communication MQTT. Les configurations du Wifi et de MQTT sont sauvegardées dans un fichier séparé. La fonction `setup()` initialise le port série pour le débogage, configure le compteur d'énergie ATM90E32 avec les paramètres spécifiés, et

Définir les données MQTT comme des capteurs dans Home Assistant

Pour contrôler les données envoyées par votre compteur d'énergie ESP32 via MQTT, vous devez définir ces points de données comme des capteurs dans Home Assistant. Suivez les étapes suivantes :

Accédez au fichier de configuration :
Ouvrez votre fichier de configuration de Home Assistant (configuration.yaml) avec l'éditeur de fichiers ou tout autre éditeur de texte.

Définissez les capteurs MQTT :
Dans le fichier configuration.yaml, ajoutez la configuration suivante pour définir vos capteurs MQTT :

```
mqtt:
  sensor:
    - name: Line Voltage A
      unique_id: esp32_voltage_a
      state_topic: "esp32energymeter/lineVoltageA"
      unit_of_measurement: "V"

    - name: Line Current A
      unique_id: esp32_current_a
      state_topic: "esp32energymeter/lineCurrentA"
      unit_of_measurement: "A"
```

Personnalisez vos capteurs :
Remplacez `Line Voltage A`, `Line Current A`, etc., par des noms qui répondent à vos besoins.
Assurez-vous que le `state_topic` correspond au topic utilisé dans votre firmware ESP32 pour publier les données. L'identifiant unique `unique_id` doit être un identifiant unique pour chaque capteur, permettant à Home Assistant de les suivre et de les gérer correctement.

Sauvegardez et redémarrez Home Assistant :
Après avoir ajouté les définitions des capteurs, sauvegardez le fichier configuration.yaml et redémarrez Home Assistant pour appliquer les changements.

Visualisez vos capteurs :
Une fois Home Assistant redémarré, vos capteurs MQTT devraient apparaître dans le tableau de bord, vous permettant de suivre les données énergétiques en temps réel.

établit les connexions au réseau Wifi et au courtier MQTT. Cette configuration initiale assure que l'ESP32 est prêt à communiquer avec Home Assistant et d'autres plateformes compatibles MQTT.
Dans la boucle `main()`, le client MQTT est mis à jour en continu pour maintenir la connexion avec le courtier. Si la connexion est interrompue, le micrologiciel tente automatiquement de se reconnecter. Par ailleurs, le logiciel est conçu pour récupérer les données énergétiques de la puce ATM90E32 et les transmettre au courtier MQTT. Cette configuration permet à Home Assistant de suivre la consommation d'énergie en temps quasi réel, fournissant ainsi des données essentielles pour la domotique. La fonction `getEnergyData()` joue un rôle crucial dans le logiciel en collectant diverses mesures énergétiques depuis la puce ATM90E32. Ces mesures incluent la tension de ligne, le courant, la puissance (active, réactive et apparente), le facteur de puissance, l'angle de phase, la fréquence et la température. Les données collectées sont ensuite publiées sur des sujets MQTT spécifiques, les rendant disponibles pour la surveillance et l'analyse dans Home Assistant. Pour les développeurs et les utilisateurs qui utilisent le mode débogage, les données énergétiques sont également affichées sur le moniteur série, ce qui facilite le dépannage et la validation des données. Pour garantir que l'ESP32 maintient une connexion stable au réseau et au courtier MQTT, nous avons implémenté la fonction `connect()`. Cette fonction gère le processus de reconnexion en cas de perte de connexion Wifi ou MQTT. Des messages de débogage fournissent un retour en temps réel sur l'état de la connexion, et une LED s'illumine pour indiquer visuellement quand les connexions sont établies avec succès.

Configuration d'un courtier MQTT dans Home Assistant

Pour permettre au compteur d'énergie ESP32 d'utiliser le protocole MQTT pour envoyer le relevé du compteur, vous devez d'abord configurer un courtier MQTT. Un broker MQTT peut être installé sur presque n'importe quel ordinateur connecté à votre réseau domestique. Il peut être configuré sur votre PC à l'aide d'un conteneur Docker, installé directement sur un Raspberry Pi, ou même hébergé sur un serveur cloud pour un accès à distance. Cependant, pour que les choses restent simples et s'intègrent parfaitement à votre installation domestique intelligente, nous allons le configurer avec Home Assistant. Pour installer le module complémentaire MQTT dans

Figure 3. Test de la connexion MQTT en s'abonnant à tous les sujets pour s'assurer que les messages sont bien reçus par Home Assistant.

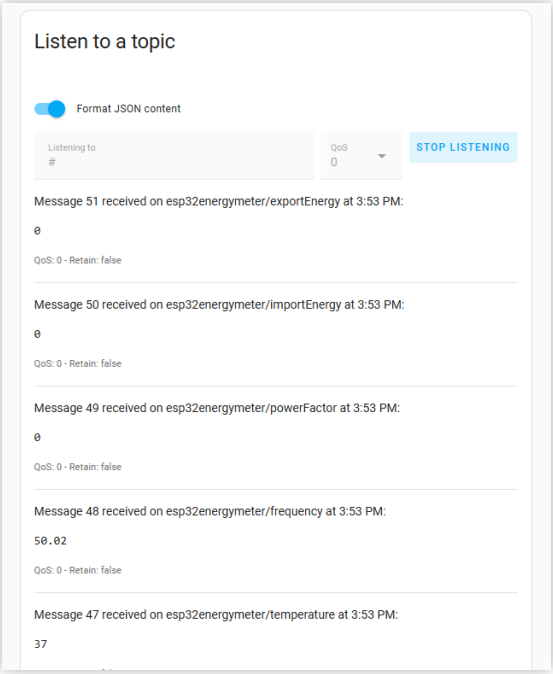
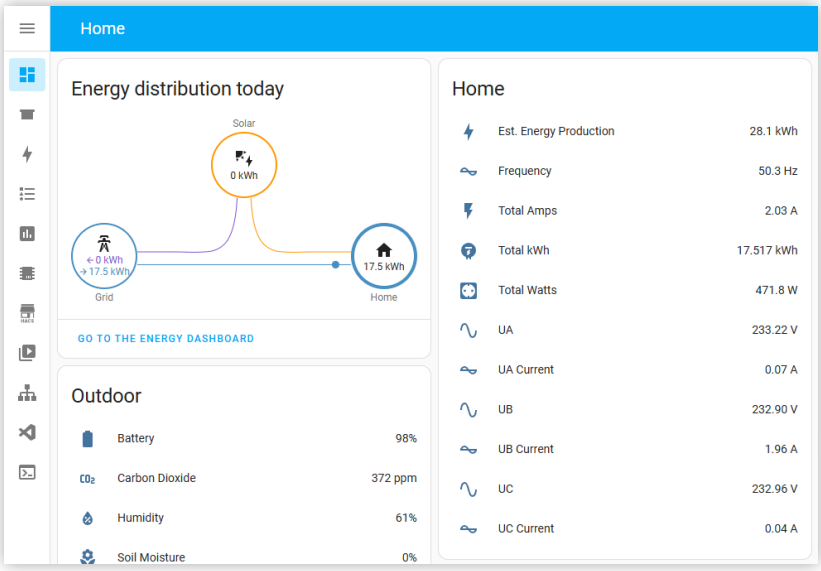


Figure 4. Tableau de bord personnalisé dans Home Assistant, montrant les données énergétiques en temps réel du compteur d'énergie ESP32.



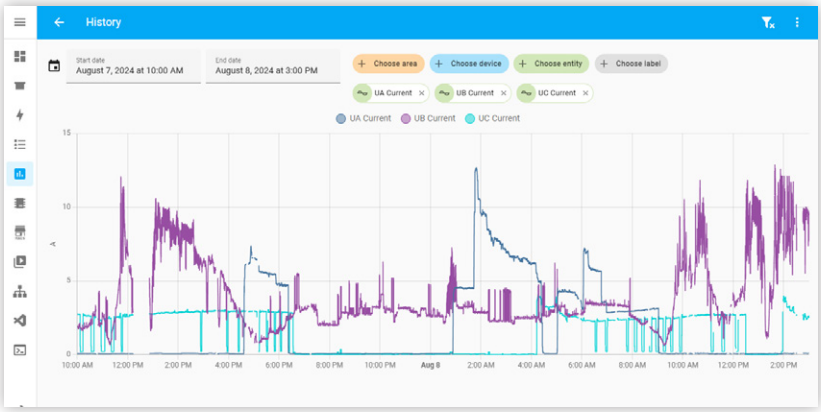


Figure 5. Exemple de graphiques créés dans Home Assistant pour surveiller la consommation énergétique dans le tableau de bord de l'historique.

Home Assistant, accédez d'abord à votre tableau de bord Home Assistant en vous rendant sur l'URL où est exécutée votre instance. Dans la barre latérale, sélectionnez *Settings*, puis *Add-ons*, et ouvrez le *Add-on Store*. Cherchez *MQTT*, et vous devriez trouver le courtier officiel *Mosquitto* parmi les résultats. Cliquez dessus, puis sélectionnez *Install* ; l'installation peut prendre un peu de temps. Une fois installé, vous pouvez configurer le courtier MQTT en modifiant les options de configuration – les paramètres par défaut conviennent généralement, mais il est recommandé de définir un nom d'utilisateur et un mot de passe. Après configuration, cliquez sur le bouton *Start* pour activer le courtier MQTT, et activez l'*option Start on boot* pour assurer son démarrage automatique à chaque redémarrage de Home Assistant (voir **figure 1)**.

Ensuite, configurez l'intégration MQTT dans Home Assistant. Rendez-vous dans *Settings*, puis dans *Devices & Services*, et sélectionnez *Integrations*. Cliquez sur *Add Integration*, recherchez *MQTT* et sélectionnez-le. Home Assistant détectera automatiquement le courtier MQTT en cours d'exécution. Si nécessaire, configurez les paramètres MQTT, tels que l'adresse IP du courtier - utilisez localhost si celui-ci est installé sur le même appareil que Home Assistant (ou bien core-mosquitto pour le même résultat). Laissez le port par défaut à *1833.* Assurez-vous que les informations d'identification de l'utilisateur sont celles d'un utilisateur actuel dans Home Assistant comme le montre la **figure 2**. Vous pouvez également créer un utilisateur spécifique pour MQTT dans Home Assistant.

Il est essentiel de configurer Home Assistant pour qu'il reçoive correctement les messages de votre compteur d'énergie ESP32. Pour tester la connexion, abonnez-vous à un sujet « # » afin de recevoir tous les messages envoyés à votre courtier MQTT (voir **figure** 3)**.** Cette page est accessible en cliquant sur *Configure* dans *Integration entities* sous l'élément *MQTT integration*.

Avec le courtier MQTT activé sur Home Assistant, vous pouvez connecter votre compteur d'énergie ESP32. Dans le fichier *config.ino* de votre micrologiciel ESP32, définissez `HOMEASSISTANT_IP` à l'adresse IP de votre instance de Home Assistant. Configurez ensuite `DEVICE_NAME`, `USER_ID`, et `PASSWORD` si l'authentification est activée sur le courtier MQTT. Après avoir configuré ces paramètres, flashez l'ESP32 avec le micrologiciel et assurez-vous qu'il se connecte au courtier MQTT avec succès. Une fois connecté, le compteur d'énergie ESP32 commencera à envoyer des données énergétiques au courtier MQTT, que vous pouvez surveiller dans Home Assistant en vous abonnant aux sujets MQTT appropriés.

Enfin, pour visualiser les données énergétiques dans Home Assistant, des entités MQTT seront automatiquement créées pour chaque sujet publié par votre compteur d'énergie ESP32. Si ce n'est pas le cas, ce qui peut arriver parfois, vous devrez définir les entités MQTT dans le fichier *configurations.yaml* de Home Assistant.

Vous trouverez les instructions nécessaires dans l'encadré « **Définir les données MQTT comme capteurs dans Home Assistant** ».

Vous pouvez accéder aux entités des capteurs sous *Settings* dans Home Assistant, puis *Devices & Services*, et *Entities*. Utilisez ces entités pour créer des tableaux de bord personnalisés dans Home Assistant, ce qui vous permettra de visualiser les données énergétiques en temps réel, de créer des graphiques et de configurer des alertes basées sur des seuils de consommation, comme illustré dans les **figures 4** et **5**. Avec MQTT et Home Assistant, vous pouvez également automatiser des actions basées sur les données énergétiques comme le montre la **figure 6**, intégrer d'autres appareils intelligents et obtenir des informations précieuses sur l'utilisation de l'énergie dans votre maison.

Pour les débutants, je recommande le document *Getting Started Guide by Home Assistant* [5], qui offre une introduction complète à la configuration et à l'utilisation de Home Assistant. Consultez également la documentation sur l'intégration MQTT de Home Assistant [6][7] pour des détails sur la configuration de MQTT et l'intégration efficace de vos appareils. Ces ressources vous aideront à démarrer avec Home Assistant et MQTT, facilitant ainsi l'installation de votre maison intelligente de manière efficace et conviviale.

240349-04

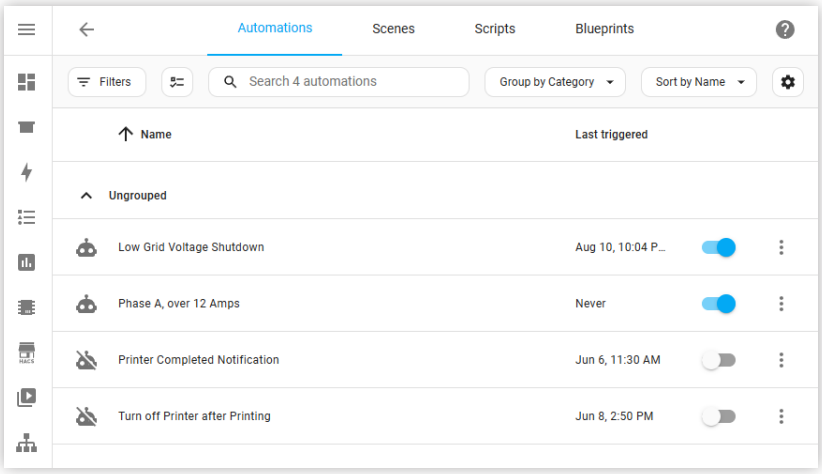


Figure 6. Actions automatisées dans Home Assistant basées sur les données énergétiques reçues du compteur d'énergie ESP32 via MQTT.

Questions ou commentaires ?

Envoyez un courriel à l'auteur (saad.imtiaz@elektor.com), ou contactez Elektor (redaction@elektor.fr).

Produits

- **Home Assistant Green**
  www.elektor.fr/20725
- **Raspberry Pi 5 (2 GB RAM)**
  www.elektor.fr/20951

À propos de l'auteur

Saad Imtiaz, ingénieur senior chez Elektor, est spécialisé en mécatronique. Il dispose d'une solide expertise dans les systèmes embarqués et le développement de produits. Au cours de sa carrière, il a collaboré avec une diversité d'entreprises, allant des start-up novatrices aux grandes corporations internationales, et a dirigé des projets de prototypage et de développement à la pointe de la technologie. Fort de son expérience approfondie dans l'industrie aéronautique et en tant que leader d'une start-up technologique, Saad apporte à Elektor un mélange précieux de compétences techniques et de vision entrepreneuriale. Il contribue au développement de projets logiciels et matériels.

LIENS

[1] Saad Imtiaz, «mise à jour du projet #3 : compteur d'énergie basé sur ESP32» Elektor 7-8/2024 : https://elektormagazine.fr/240244-04
[2] Dépôt Github du compteur d'énergie ESP32 : https://github.com/ElektorLabs/esp32-energymeter
[3] Bibliothèque ATM90E32 pour Arduino par CircuitSetup : https://github.com/CircuitSetup/ATM90E32
[4] Bibliothèque MQTTPubSubClient de hideakitai : https://github.com/hideakitai/MQTTPubSubClient
[5] Getting started, Home Assistant : https://www.home-assistant.io/getting-started/
[6] MQTT Integration, Home Assistant : https://www.home-assistant.io/integrations/mqtt/
[7] MQTT Sensor, Home Assistant : https://www.home-assistant.io/integrations/sensor.mqtt/