sauver la planète avec la domotique ?

MQTT sur l'Arduino Nano RP2040 Connect

Clemens Valens (Elektor Lab)

Vous pouvez automatiser votre maison avec les bons composants et un peu d'ingéniosité, et ce projet basé sur l'Arduino Nano RP2040 Connect est un excellent point de départ. En prime, vous contribuerez peut-être à sauver la planète.

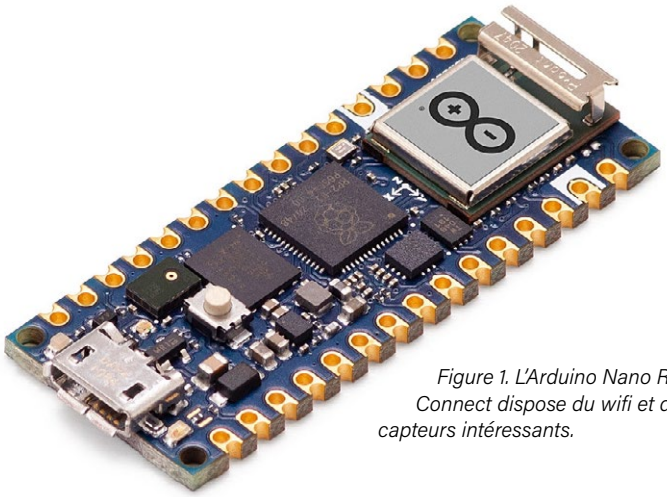

Figure 1. L'Arduino Nano RP2040 Connect dispose du wifi et de quelques capteurs intéressants.

Aujourd'hui, l'environnement semble être la préoccupation numéro un de nombreuses personnes. Notre planète est grande, et, s'il existe de nombreux pollueurs sur lesquels vous n'avez aucun contrôle, il y a un endroit où vous pouvez faire la différence : votre habitation. Son automatisation peut la rendre plus économe en énergie et contribuer ainsi à préserver la planète. Voici de quoi commencer.

La domotique s'applique généralement aux domaines suivants :

- Climatisation
- Éclairage
- Gestion de l'énergie
- Contrôle d'accès
- Utilisation de l'eau

Les trois premiers concernent tous les économies d'énergie, ce qui est fait en contrôlant la température et l'humidité dans le bâtiment par chauffage, refroidissement, ventilation et stores, et en coupant l'alimentation des éclairages, des appareils et des machines lorsqu'ils ne sont pas nécessaires. Le contrôle de l'eau est également utile, sauf s'il s'agit d'arroser la pelouse, ce qui est toujours un gaspillage.

Nous avons besoin de capteurs

Le contrôle de paramètres, tels que la température d'une pièce ou la consommation de courant d'une machine, nécessite des capteurs et des actionneurs. Dans cet article, je vais présenter une sorte de dispositif universel qui lit les capteurs et transmet les données acquises à un contrôleur domotique en utilisant MQTT. Les actionneurs qui agissent directement sur des moteurs, des pompes, des relais et des éclairages, etc. ne sont pas abordés ici.

Pour ceux qui ne sont pas des familiers de la domotique, un contrôleur est le cœur d'un système domotique qui fournit l'intégration de tous les appareils en un système, y compris vous et les autres utilisateurs (si vous acceptez d'être considéré comme un appareil vivant). MQTT est un protocole de gestion d'échange de données qui est devenu assez populaire dans l'IdO et l'automatisation.

L'Arduino Nano RP2040 Connect

Le dispositif de capteurs décrit ici est basé sur une carte Arduino Nano RP2040 Connect (**figure 1**) qui comporte un gyroscope à 3 axes, un accéléromètre à 3 axes et un microphone. L'usage de tels capteurs peut sembler étrange dans le cadre d'un système domotique, mais ils peuvent être utiles. (De plus, cela change des projets habituels de température-humidité).

Par exemple, lorsqu'il est monté sur une porte ou une fenêtre, le gyroscope peut détecter ses mouvements, les entrées et sorties prévues ou imprévues, ou signaler une fenêtre laissée ouverte. Les accéléromètres ont de nombreuses applications dans la détection des vibrations (le moteur du congélateur fonctionne-t-il en permanence ?) et un microphone peut détecter des sons là où il ne devrait pas y en avoir (robinet qui coule) ou remarquer un changement de bruit de fond ou de fréquence (moteur hors contrôle ?). Mais si votre projet n'en a pas l'utilité, le ***framework*** est facile à adapter à d'autres capteurs.

L'Arduino Nano RP2040 Connect possède une connectivité wifi grâce à son module sans fil NINA (un ESP32 déguisé) et nous allons l'utiliser pour nous connecter au réseau Wi-Fi de la maison. Dans mon cas, le contrôleur domotique est Home Assistant (HA) [1] tournant sur une carte Raspberry Pi 3B+, mais tout autre contrôleur capable de gérer le protocole MQTT peut également faire l'affaire (ce qui est le cas d'environ 99 % d'entre eux). Le contrôleur se connecte également au

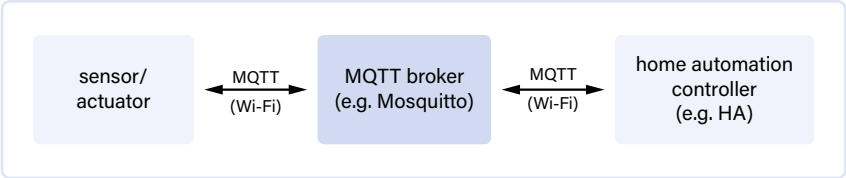

Figure 2. Une vue d'ensemble de haut niveau du système domotique basé sur MQTT présenté dans cet article.

réseau wifi. La **figure 2** donne un aperçu de haut niveau du système.

Préliminaires et exigences

Parce qu'il est urgent de sauver la planète, nous n'allons pas nous perdre ici dans toutes sortes de détails techniques sur la carte Arduino – nous allons passer directement aux choses sérieuses.

Si vous ne disposez pas encore d'un contrôleur domotique, commencez par en installer et en configurer un. C'est plus facile à dire qu'à faire. Pour plus de détails, voir par exemple [1]. N'oubliez pas que le contrôleur a besoin de capacités MQTT, pour lesquelles il peut avoir besoin d'un complément. Si, comme moi, vous utilisez HA comme contrôleur, vous pouvez installer l'intégration Mosquitto (avec deux « t ») [2], d'usage très répandu. Il s'agit d'un courtier MQTT, un logiciel qui gère des échanges de messages, par exemple entre la carte Arduino et HA.

Préparation de l'EDI Arduino

Lorsque vous avez un contrôleur domotique fonctionnel avec des capacités MQTT, vous pouvez passer à la configuration de l'environnement de développement Arduino :

1. Installez l'EDI Arduino sur votre PC. Il en existe de nombreuses versions ; j'ai utilisé la 1.8.19.
2. En utilisant le gestionnaire de cartes de l'EDI Arduino (*Outils → Type de carte Gestionnaire de cartes…*) installez le paquet de cartes « *Arduino Mbed OS Nano Boards* » (j'ai utilisé la version 3.2 ; voir **figure 3**).
3. Dans l'EDI, sélectionnez la carte Arduino Nano RP2040 Connect - voir **figure 4** (*Outils → Type de carte → Arduino Mbed OS Nano Boards → Arduino Nano RP2040 Connect*).
4. En utilisant le gestionnaire de bibliothèques de l'EDI (*Outils → Gérer les bibliothèques…*), installez les bibliothèques suivantes (les versions que j'ai utilisées sont entre parenthèses) :
   - *WiFiNINA* (v1.8.13)
   - *ArduinoMqttClient* (v0.1.6)
   - *Arduino_LSM6DSOX* (v1.1.0)
5. Téléchargez mon croquis depuis [3] et décompressez-le dans le dossier *sketchbook* de l'EDI. Notez que le croquis se compose de deux fichiers, dont l'un est nommé `arduino_secrets.h`. Entrez vos informations d'accès réseau dans ce fichier.

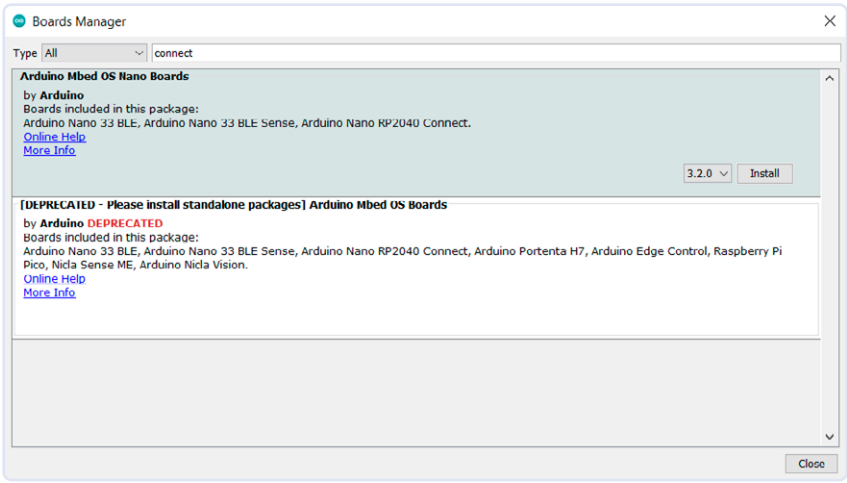

Figure 3. Utilisez le gestionnaire de cartes de l'EDI pour installer le support pour l'Arduino Nano RP2040 Connect.

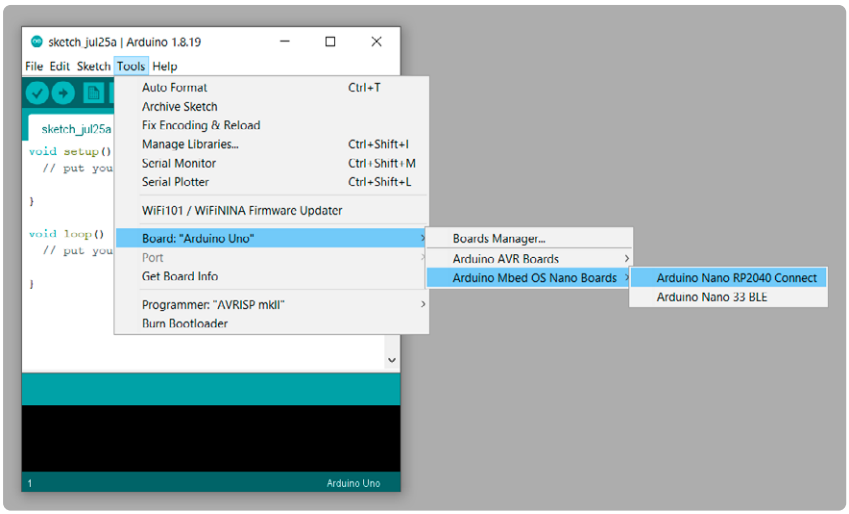

Figure 4. Avant d'essayer de télécharger quoi que ce soit sur la carte, assurez-vous de sélectionner la bonne carte (et son port série !).

Configuration du programme

Avant de pouvoir compiler mon croquis, vous devez collecter quelques informations auprès de votre courtier MQTT. Dans HA, le broker *Mosquitto* vous demande de définir un utilisateur et un mot de passe ; sans quoi, aucun trafic MQTT ne passera. Saisissez les informations d'identification MQTT dans le fichier `arduino_secrets.h` en tant que `SECRET_MQTT_USER` et `SECRET_MQTT_PASSWORD`. Entrez également le SSID et la phrase de passe de votre réseau wifi dans ce fichier.

Dans le fichier principal du croquis (ligne 37 au moment de la rédaction), saisissez l'adresse IP du courtier MQTT. Dans HA, il s'agit simplement de l'adresse IP de HA, que vous pouvez trouver dans *Settings → System Network* (c'est là qu'elle se trouvait dans HA Core v2022.8.6 avec HA OS v8.4) :

```
const char broker[] =
   'xxx.xxx.xxx.xxx';
```

Si, pour une raison quelconque, vous avez modifié le port MQTT par défaut, vous devrez également modifier la ligne suivante :

```
int port = 1883;
```

Une fois le croquis configuré correctement, vous pouvez le compiler et télécharger l'exécutable sur la carte Arduino Nano RP2040 Connect. Il ne devrait pas y avoir d'avertissement ou d'erreur, même si le compila-

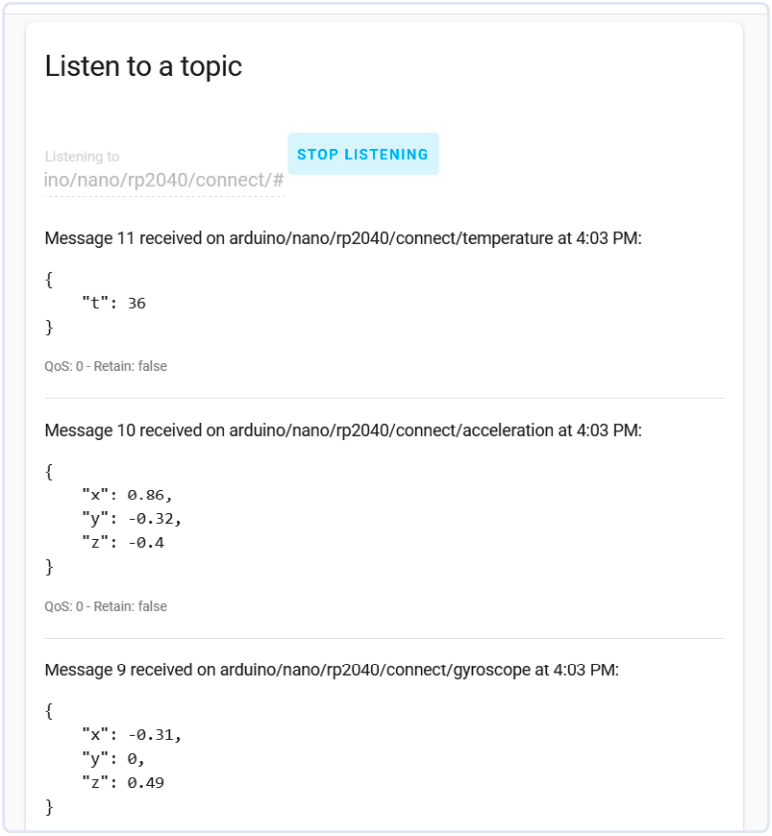

Figure 5. Réception de messages MQTT avec Mosquitto dans Home Assistant.

teur est réglé sur « sortie verbeuse » et que tous les avertissements sont activés dans l'EDI Arduino (*Fichier → Préférences → Paramètres →Avertissement du compilateur → Tout*).

Essayez-le

En supposant que tout se soit bien passé et que vous n'ayez pas fait d'erreur lors de la saisie des mots de passe, etc., la carte devrait maintenant se connecter au réseau wifi et commencer à envoyer des messages MQTT à un rythme de 0,1 Hz (c'est-à-dire toutes les 10 secondes). La LED RVB de la carte clignote à chaque fois qu'un message est envoyé. Si la LED ne clignote pas ou reste allumée, quelque chose ne fonctionne pas. Le rouge indique l'absence de connexion réseau et le bleu un problème de capteur.

Le contrôleur domotique doit recevoir les messages MQTT. Dans HA avec le module complémentaire Mosquitto, cliquez sur « Configure » sur la carte d'intégration du courtier Mosquitto, et faites défiler vers le bas jusqu'à « Listen to a topic ». Entrez `arduino/nano/rp2040/connect/#` et cliquez sur « Start listening. » Des messages devraient s'afficher à raison d'un toutes les dix secondes (**figure 5**).

Mon croquis envoie les données de gyroscope et d'accélération toutes les dix secondes, ainsi que la température (Surprise ! C'est que la puce IMU LSM6DSOX possède un capteur de température intégré). Le plus souvent, la valeur de la température sera plus élevée que la température ambiante, car le capteur est chauffé par le module wifi de la carte.

Lorsque le microphone entend un son fort, il envoie un message. Frappez dans vos mains pour l'essayer. (N'oubliez pas qu'il peut y avoir jusqu'à dix secondes de latence).

Si vous saisissez dans « Publier un paquet » `arduino/nano/rp2040/connect/incoming/xxx` avec `xxx` (le « sujet ») remplacé par un mot ou un nombre, puis cliquez sur *Publier*, vous devriez le voir apparaître sur le moniteur série de l'EDI Arduino. Vous pouvez éventuellement ajouter une charge utile (n'importe quoi fera l'affaire).

Que faire à partir de maintenant ?

A partir de maintenant vous êtes livré à vous-même. Il y a plusieurs options :

a. Ajouter des routines de traitement de données et de signaux au logiciel du capteur pour qu'il n'envoie des messages que si certains événements sont détectés.
b. Ajouter des capteurs différents ou supplémentaires.
c. Ajouter des règles d'automatisation au contrôleur domotique pour qu'il réponde de manière utile aux messages reçus.
d. Tout ce qui précède.
e. Ne pas faire de domotique (voir l'encadré).

Pour vous aider à démarrer avec les options « a » et « b », je vais vous expliquer brièvement le fonctionnement du croquis. Pour l'option « c », veuillez vous référer à la documentation du contrôleur domotique.

Fonctionnement du croquis

Le croquis commence par définir certaines choses comme les détails du réseau, mais aussi les sujets MQTT qui seront utilisés. Tous les messages envoyés par notre appareil commencent par

`arduino/nano/rp2040/connect/`

En outre, l'appareil n'écoute que les messages qui commencent par ce préfixe. C'est un long préfixe, mais il est idéal pour le débogage. Le préfixe est suivi d'un « sujet » qui peut être n'importe quoi. La seule chose importante ici est que le courtier ou la destination MQTT doit écouter le même sujet, sinon les messages seront simplement ignorés. Une bonne pratique consiste à utiliser des sujets significatifs. Il peut y avoir autant de sujets que vous le souhaitez, et leur longueur peut être (presque) illimitée.

Traitement du son

Les capteurs sont lus avec des fonctions d'aide qui ont des noms significatifs, sauf pour le microphone, qui est appelé *PDM* (c'est un microphone numérique). Autre différence par rapport aux autres capteurs, le microphone fonctionne dans une sorte de fil en arrière-plan qui envoie continuellement des échantillons audio dans un tampon. Les autres capteurs sont interrogés par la boucle principale toutes les dix secondes. Une dernière différence entre les données audio et les autres est que les échantillons audio sont filtrés par un filtre passe-bas avec une fréquence de coupure de 10 Hz pour qu'il ne réponde qu'aux signaux à basse fréquence (boums et fracas). Quand un tel son est détecté, un message est envoyé avec le sujet `microphone/alarm` (plus le préfixe, bien sûr). Les données des autres capteurs sont transmises sans aucun filtrage.

Sujets MQTT

Les quelques fonctions d'aide pour compo-

Ne vous lancez pas là-dedans à la maison

Bien sûr, la technologie peut vous aider à économiser de l'énergie à la maison et au bureau, mais elle a un coût qui peut l'emporter sur son gain potentiel. Le projet décrit dans cet article utilise une petite carte Arduino RP2040 Nano Connect qui communique par wifi avec un Raspberry Pi exécutant Home Assistant. Tous ces appareils, y compris le routeur wifi, consomment continuellement de l'énergie car ils sont allumés en permanence. Vous pouvez en estimer le coût en épluchant votre facture d'électricité : il est appréciable et à votre charge. Mais il existe aussi des coûts cachés qu'on oublie facilement : les ressources consommées pour fabriquer tout ce que vous utilisez.

Je n'ai aucune idée de l'empreinte carbone de la production d'une carte Arduino ou Raspberry Pi, mais elle n'est évidemment pas nulle (n'oubliez pas d'inclure celle des composants). De même, le logiciel utilisé par notre système, y compris notre propre petit croquis, a été développé sur de nombreux ordinateurs dans le monde entier et stocké sur des serveurs dans le nuage. Comme notre petit système, cet énorme écosystème consomme (beaucoup) d'énergie pour rester en ligne. Et n'oubliez pas les ressources utilisées pour assembler et fabriquer le tout. Bien sûr, ces coûts sont partagés par de nombreux utilisateurs, mais il faut en tenir compte.

Où avez-vous acheté le matériel de votre système ? Il y a de fortes chances qu'une partie au moins ait été commandée en ligne. Quel que soit l'endroit où vous l'avez acheté, il a été transporté de quelque part sur la planète jusqu'à chez vous, consommant là encore de précieuses ressources.

Enfin, la technologie ne cessant de progresser, notre système domotique sera bientôt obsolète et devra être mis à niveau ou éliminé, ce qui ajoutera une fois de plus de nombreux coûts cachés.

Ainsi, vous économiserez peut-être un peu d'énergie chez vous, mais pour y parvenir, vous en aurez sans doute dépensé beaucoup plus que ce que ça vous rapportera. Il est donc préférable pour la planète de ne pas essayer d'économiser de l'énergie en faisant des trucs astucieux de domotique. Prendre l'habitude d'allumer et d'éteindre les lumières soi-même est plus efficace et vous oblige à marcher un peu, ce qui est bon pour la forme. Pour sauver la planète, rien de mieux que le bon vieux bon sens.

ser les messages MQTT sortants sont assez explicites. Une chose à savoir est que l'interface d'impression de `MqttClient print` ne s'applique qu'à la partie charge utile d'un message – le sujet doit être assemblé d'une autre manière.
Tous les messages MQTT entrants qui commencent par le préfixe

`arduino/nano/rp2040/connect/incoming/`

sont acceptés. Ceci est géré par la bibliothèque *MqttClient* qui appelle la fonction `mqtt_message_receive` lorsque toutes les conditions de réception sont remplies. Par défaut, le croquis s'abonne au sujet générique « # », ce qui signifie que tous les sujets sont valides. Vous pouvez changer cela en le remplaçant par des sujets plus spécifiques.
Vous pouvez, bien sûr, vous abonner à plusieurs sujets simultanément. Les abonnements sont gérés par le courtier MQTT, et non par le croquis. C'est donc le courtier qui peut imposer des limites. La quantité de mémoire disponible pour le croquis n'a pas de réelle influence.

Un peu de JSON

Les données du capteur sont ajoutées comme charge utile aux sujets. J'ai utilisé un formatage de type JSON pour cela, mais manuellement, sans l'aide d'une bibliothèque JSON spéciale. L'avantage de JSON est qu'il est compris par beaucoup d'autres programmes, ce qui rend son analyse beaucoup plus facile.

Conclusion

Voilà, c'est tout pour le moment. Même si j'ai fait de mon mieux pour que les choses restent simples et claires, il y a de fortes chances que vous rencontriez des problèmes à un moment ou à un autre. Le sujet est plus complexe que ce que cet article a pu vous laisser croire. N'hésitez pas à consulter Internet pour plus d'informations – il existe des foules de sites sur MQTT et Home Assistant (et même les deux), qui regorgent de personnes utiles, de conseils, d'astuces et de trucs. Et quand vous rencontrez des difficultés et que les choses cessent de fonctionner, revenez à la dernière configuration où ça marchait encore.

(220420-04) — VF : Helmut Müller

LIENS

[1] « la domotique, c'est facile avec… », C. Valens, Elektor Magazine 9-10/2020 : ww.elektormagazine.fr/magazine/elektor-156/58991
[2] MQTT dans Home Assistant : www.home-assistant.io/docs/mqtt/broker/
[3] Téléchargements pour cet article : www.elektormagazine.fr/220420-04

À propos de l'auteur

Clemens Valens est le technologue créatif d'Elektor. Il est titulaire d'un BSc en électronique et d'un MSc en électronique et technologie de l'information. Clemens a commencé à travailler pour Elektor en 2008 en tant que rédacteur en chef d'Elektor France. Il produit actuellement des tutoriels d'ingénierie et des revues de produits pour Elektor TV. Clemens est également responsable du site web de la communauté Elektor Labs, où les passionnés d'électronique peuvent publier leurs travaux et interagir avec leurs pairs du monde entier.

Produits

- **Arduino Nano RP2040 Connect avec des connecteurs**
  www.elektormagazine.fr/arduino-nano-rp2040-connect
- **Livre en anglais « Mastering Microcontrollers Helped by Arduino », Clemens Valens (SKU 17967)**
  www.elektor.fr/17967
- **Elektor Ultimate Sensor Kit (SKU 19104)**
  www.elektor.fr/19104