Développement d'un Composant ESPHome
Partie 1 : Introduction

Présentation

Dans cette série je vais vous présenter :

- Une introduction rapide à l'utilisation et à la création de composants pour ESPHome
- Description de mon environnement de développement logiciel
- Description de mon environnement de développement matériel
- Description du processus de développement et de soumission de composant
- Exemple de développement d'un composant

Introduction

Terminologie :

- Un « **module** » (**device**) sert à désigner un ensemble de **composants** grouper par un utilisateur pour réaliser une fonction spécifique.
- Un « composant » (**component**) sert à désigner un élément de base de la bibliothèque d'ESPHome. Cela peut être un composant physique comme un « capteur de température » ou bien un composant logique comme un bus I²C.

La principale vocation d'ESPHome est de permettre à des « **utilisateurs »** sans compétence informatique particulière de créer automatiquement le logiciel pour un **module** électronique. Pour ce faire, l'utilisateur décrit la configuration de son module dans un fichier YAML. ESPHome utilise ensuite une bibliothèque de **composants** qu'il assemble en fonction des besoins de l'utilisateur, générant ainsi automatiquement le micrologiciel (firmware) nécessaire au bon fonctionnement du module. L'utilisateur doit ensuite rassembler les différents composants sur une carte électronique et la connecter à un ordinateur pour charger et exécuter le code généré par ESPHome.

Un « **développeur** » va créer de nouveaux composants pour enrichir la bibliothèque d'ESPHome. Cependant, cette tâche est cependant beaucoup plus complexe, car elle implique de décrire le comportement du composant en utilisant les langages C++ et Python. Une bonne connaissance du langage C++ ainsi que des bases du langage Python est nécessaire pour mener à bien cette démarche.

Ayant été confronté au besoin de créer un **composant** qui n'existait pas dans la bibliothèque d'ESPHome, je me suis interrogé sur les étapes et les outils requis pour réaliser cette tâche. Bien qu'il existe de nombreux tutoriels sur la création de **modules** dans ESPHome, très peu se penchent sur la création de nouveaux composants. J'ai donc procédé par essais et erreurs pour finalement établir un environnement et un processus à suivre. Cette expérience m'a apporté de précieuses connaissances, que je souhaite maintenant partager.

Il convient de noter qu'il existe certainement d'autres méthodes (potentiellement meilleures) pour accomplir cette tâche.

Vision à haut niveau d'ESPHome

Tout d'abord, permettez-moi de vous présenter le fonctionnement d'ESPHome à un niveau élevé d'abstraction, tel que je le comprends.

Lors de la création d'un nouveau module dans ESPHome, vous devez d'abord définir tous les composants que vous souhaitez utiliser et spécifier comment ils seront interconnectés. Par exemple, si vous désirez concevoir un module de « mesure de température dans une pièce », voici comment vous pourriez le décrire :

> Vous allez indiquer que vous souhaitez installer sur une carte de développement esp32dev un microcontrôleur ESP32 qui sera connecté à un capteur BMP085 par le biais d'un bus I²C. Ce

capteur sera chargé de mesurer la température, puis il enverra ces données à Home Assistant via l'API native d'ESPHome en utilisant une connexion Wi-Fi.

Le fichier Yaml va ressembler à cela :

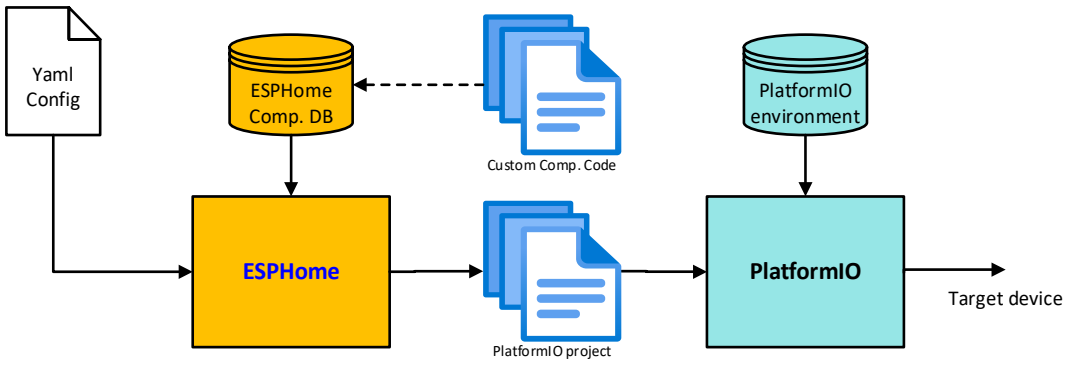
```
esphome :
  name: test-bmp085
  platform : esp32
  board: esp32dev

api :

wifi :
  ssid: !secret wifi_ssid
  password: !secret wifi_password

i2c:
  sda: 21
  scl: 22
  id: bus_i2c_id

sensor:
  - platform: bmp085
    i2c_id : bus_i2c
    address: 0x77
    temperature:
      name: "Outside Temperature"
```

Dans ce fichier on indique qu'on crée un nouveau module ESPHome appelé « test-bpm085 », que le microprocesseur utilisé est un « ESP32 », lequel est monté sur une carte de développement « esp32dev », que le module se connecte à votre réseau « Wi-Fi », qu'il communique avec Home Assistant grâce au « protocole api natif », qu'il utilise un bus « I2C », et qu'un « capteur BMP085 » connecté au bus I²C pour retourne la température.

À partir de cette description la *magie* d'ESPHome opère en exécutant les tâches suivantes :

- Dans un premier temps ESPHome va lire le fichier Yaml. Ce qui lui permet d'énumérer tous les composants utilisés et de connaitre toutes les interconnexions (création interne d'une netlist). Dans cette phase ESPHome fait une vérification de la syntaxe et de la sémantique du fichier Yaml. C'est la phase dite de préparation et de vérification.
- Dans un deuxième temps, ESPHome va rechercher dans sa bibliothèque tous les composants qui sont utilisés dans ce module. Par exemple : un composant wifi, un composant I2C, un composant Api, un composant BMP085, un composant capteur, etc. Il copie alors le code source de ces composants dans un répertoire créer spécifiquement pour le module en cours d'assemblage.
- Ensuite il génère un fichier en C++ (main.cpp) afin « d'instancier » et « de connecter » les composants entre eux à partir de la netlist interne. C'est la phase de génération de code. Au cours de cette phase, ESPHome génère également tout l'environnement nécessaire à PlatformIO. Il lance ensuite PlatformIO qui va se charger de créer, télécharger, et lancer l'exécution du firmware résultant sur la cible matérielle.

Ce diagramme décrit le processus :



Donc pour résumer, après validation du fichier de configuration, les fichiers sources des différents composants utilisés sont copiés dans le projet PlatformIO, un fichier main.cpp ainsi que le fichier plaformio.ini sont générés et le contrôle est passé à PlatformIO.

PlatformIO commence par vérifier que l'environnement de développement nécessaire, tel que les compilateurs, les bibliothèques, les utilitaires de programmation, etc. sont bien présents et à jour (cet environnement est spécifié dans le fichier de configuration `platformio.ini` généré par ESPHome. PlatformIO va ensuite compiler et télécharger le code résultant dans l'appareil cible.

Dans cette série de présentations nous allons nous intéresser non pas à **l'utilisation** mais à la **création** de composants.

Le processus de création d'un composant comporte les étapes suivantes :

- l'écriture d'une documentation qui indiquera comment utiliser votre composant.
- l'écriture de fichiers Python qui serviront à valider et générer du code C++ à partir du fichier de configuration.
- L'écriture de fichiers C++ pour « modéliser » le composant. Cette partie du développement est la plus longue et la plus complexe car elle nécessite d'analyser en détail les spécifications du composant, de comprendre un minimum l'architecture interne d'ESPHome, afin d'écrire le code qui modélisera le comportement du composant.
- La soumission de votre nouveau composant à la bibliothèque d'ESPHome. C'est dans cette phase que votre code et votre documentation seront scrutés pour vérifier que vous respectez les contraintes très strictes d'ESPHome. Votre composant doit au final être approuvé par une personne ayant des droits privilégiés…
- Bien que cela ne soit mentionné nulle part, il semble que lorsque vous ajoutez un nouveau composant à ESPHome il est implicitement entendu que vous vous engagez à maintenir ce composant (bug-fix et enhancement). Pour cela votre nom de développeur GitHub est associé au composant.