Stroomlijnen van **MCU-ontwikkeling** met **ESP-IDF Privilege Separation**

Harshal Patil, Espressif

Dit artikel introduceert privilege separation in microcontroller-toepassingen met behulp van ESP-IDF van Espressif Systems. Hiermee wordt firmware in een beschermde kern en een gebruikerstoepassing gescheiden, wat de ontwikkeling vereenvoudigt. Het artikel behandelt stappen om aan de slag te gaan, waardoor MCU-ontwikkeling efficiënter verloopt.

Toepassingen op microcontrollers (MCU's) worden meestal ontwikkeld als monolithische firmware. Maar in een algemeen besturingssysteem zijn er twee werkingsmodi, de kernel- en de gebruikersmodus. In de kernelmodus heeft het programma directe en onbeperkte toegang tot systeembronnen, terwijl in de gebruikersmodus het applicatieprogramma geen directe toegang heeft tot systeembronnen. Om toegang te krijgen tot die systeembronnen, moet een *system call* gedaan worden. Het belangrijkste idee achter het realiseren van deze scheiding is om de ontwikkeling van de eindtoepassing gemakkelijker te maken zonder dat men zich zorgen hoeft te maken over de onderliggende veranderingen in het systeem, net zoals desktop- of smartphone-toepassingen worden ontwikkeld: het onderliggende besturingssysteem handelt de kritieke functionaliteit af en de eindtoepassing kan de interface gebruiken die door het besturingssysteem wordt aangeboden.

ESP Privilege Separation

Traditioneel wordt elke ESP-IDF-applicatie op een Espressif SoC gebouwd als één enkele monolithische firmware zonder enige scheiding tussen de 'kern'-componenten (besturingssysteem, netwerken enzovoort) en de 'applicatie'- of 'bedrijfs'-logica. In het ESP Privilege Separation framework splitsen we de firmware-image op in twee afzonderlijke en onafhankelijke binaries: de beveiligde en de gebruikerstoepassing.

Aan de slag

Laten we beginnen met het bouwen van ons eerste project, Blink. De knipper-LED is het "Hello World"-equivalent in embedded systeemwereld en demonstreert het eenvoudigste wat je met een MCU kunt doen om een fysieke uitvoer te zien. Aan de slag dus!

Stap 0: hardwarevereisten voor het project

- Een ESP32-C3 of ESP32-S3-gebaseerd development board met een on-board LED. Enkele devkits die gebruikt kunnen worden zijn de ESP32-C3-DevKitM-1 of de ESP32-S3-DevKitC-1. We kiezen de ESP32-C3-DevKitM-1 voor de volgende hands-on beschrijving.
- Een USB 2.0-kabel (standaard-A naar micro-B)
- Een computer met Windows, Linux of macOS

Stap 1: het ESP Privilege Separation-project ophalen

- Kloon de ESP Privilege Separation project-repository

```
git clone https://github.com/espressif/esp-privile-
ge-separation.git
```

Stap 2: setup van de ESP-IDF-omgeving

- Kloon het ESP-IDF project-repository

```
git clone -b v4.4.3 --recursive https://github.com/
espressif/esp-idf.git
```

- Setup van de tools

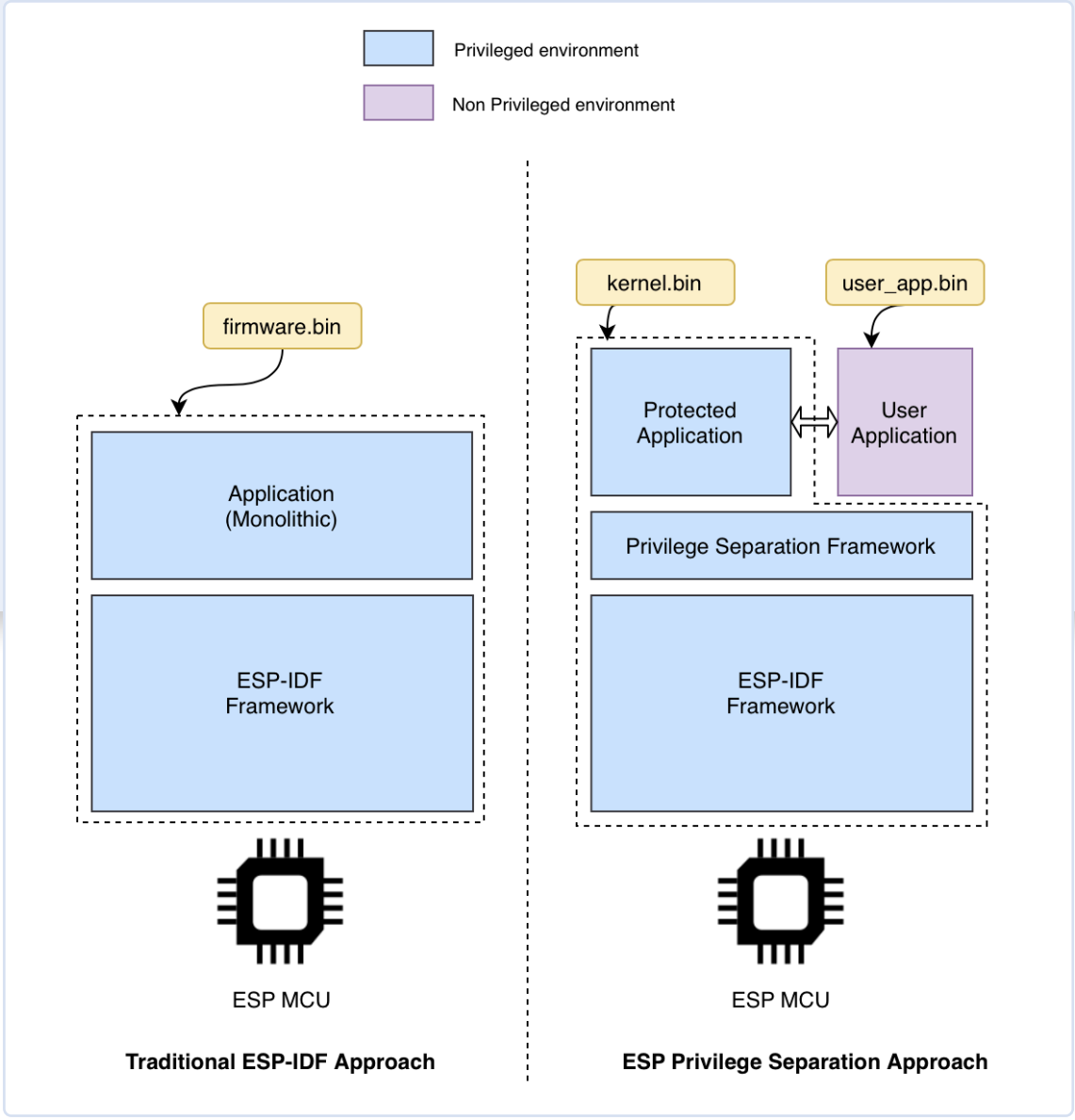
```
cd esp-idf
./install.sh
```

- Setup van de omgevingsvariabelen

```
source ./export.sh
```

- Toepassen van de ESP Privilege Separation-specifieke patch op ESP-IDF

```
git apply -v /idf-patches/
privilege-separation_support_v4.4.3.patch
```



Afbeelding 1. Traditionele ESP-IDF gebouwd als één enkele monolithisch blok firmware versus het ESP Privilege Separation-framework, waarbij het firmware-image is opgesplitst (bron: Espressif).

Stap 3: het Blink-voorbeeld uitvoeren

- Bouw van het voorbeeld

```
cd /examples/blink
idf.py set-target esp32c3
idf.py build
```

- Het voorbeeld flashen en uitvoeren

```
idf.py flash monitor
```

Een duik in de implementatie

Nu hebben we onze LED aan het knipperen, maar waarom verschilt dit van elk ander knipper-project? Antwoord: Privilege Separation. Laten we de implementatie proberen te begrijpen:

- Het voorbeeld is opgesplitst in twee applicaties: de beveiligde app en de gebruikersapp.
- De beveiligde app zoekt een gebruikerspartitie in de partitietabel en laadt de app in de door de gebruiker gedefinieerde secties.
- Vervolgens worden de geheugensecties voor de regio met lagere rechten (WORLD1) geconfigureerd en wordt toegang verleend zoals geconfigureerd door de ontwikkelaar.
- Tenslotte wordt er een aparte taak met lage privileges gestart met het toegangspunt voor de gebruiker.
- Zodra de gebruikersapp is geladen, wordt een taak gestart die de LED laat knipperen die is aangesloten op GPIO8 (ESP32-C3-DevKitM-1).

Integratie met ESP RainMaker

Nu hebben we onze LED-knipperende devkit, maar we zijn nog steeds niet in staat om de LED op ons commando te laten knipperen. Om het te kwalificeren als een echt 'connected' apparaat en de LED op commando te bedienen, moeten we het integreren met ESP RainMaker. ESP RainMaker is een lichtgewicht AIoT Cloud-software, waarmee gebruikers op maat gemaakte AIoT-oplossingen kunnen bouwen, ontwikkelen en implementeren met een minimum aan code en met maximale veiligheid. Laten we beginnen met het bouwen van het `rmaker_switch` voorbeeld in het ESP Privilege Separation-project.

Stap 0: vereisten

- ESP-IDF voor ESP Privilege Separation moet zijn ingesteld voordat er een voorbeeld wordt gebouwd
- ESP RainMaker Mobile Application (Android/iOS)
- WiFi-netwerk

Stap 1: setup van ESP RainMaker

- Kloon de ESP RainMaker project-repository

```
git clone --recursive https://github.com/espressif/
esp-rainmaker
git checkout 00bcf4c0c30d96b8954660fb396ba313fb6c886f
export RMAKER_PATH=
```

Stap 2: het *rmaker_switch* voorbeeld uitvoeren

- Bouw van het voorbeeld

```
cd /examples/rmaker_switch
idf.py set-target esp32c3
idf.py build
```

- Het voorbeeld flashen en uitvoeren

```
idf.py flash monitor
```

Stap 3: provisioneren van het apparaat

- Zodra het voorbeeld draait, verschijnt er een QR-code in de terminal. Deze QR-code kan gebruikt worden voor WiFi provisioning
- Meld je aan bij de mobiele applicatie ESP RainMaker en klik op *Add Device*. Dit opent de camera voor het scannen van QR-codes
- Volg de Provision-workflow zodat je apparaat verbinding kan maken met je WiFi-netwerk

Stap 4: de LED-schakelaar bedienen

- Uiteindelijk zie je dat een *Switch* is toegevoegd aan het startscherm van de mobiele app
- Je kunt nu proberen het Switch-pictogram om te schakelen om zo de LED te bedienen

Maar hoe hebben we deze naadloze ESP RainMaker integratie bereikt? We introduceren een `rmaker_syscall` component in ons knipper-voorbeeld, dat systeemaanroepen voor het ESP Rainmaker framework blootlegt, en wanneer de gebruikersapplicatie opstart, initialiseert het de ESP Rainmaker-service in de beveiligde applicatie, en creëert het een ESP RainMaker switch-device. Omdat de ESP RainMaker-component in de beveiligde applicatie wordt geplaatst, zijn systeemaanroepen nodig voor alle openbare API's die door deze component worden blootgelegd. De eenvoudige uitbreidingsmogelijkheden van ESP Privilege Separation laten je toe om toepassingsspecifieke aangepaste systeemaanroepen toe te voegen, en deze laag gebruikt de data opgeslagen in de apparaatcontext om user-space lees- en schrijf-callbacks uit te voeren.

OTA firmware-updates met ESP Privilege Separation

Over-the-air (OTA) firmware actualiseren is een van de belangrijkste functies voor elk connected apparaat. Het stelt ontwikkelaars in staat om nieuwe functies en bugfixes uit te brengen door de applicatie op afstand bij te werken. In ESP Privilege Separation zijn er twee applicaties – *protected_app* en u*ser_app* – waarvoor het framework de mogelijkheid biedt om beide binaries onafhankelijk van elkaar bij te werken. De beschermde applicatie, die een hoger privilege heeft, kan worden bijgewerkt door gewoon de normale OTA-interface te volgen die wordt geleverd door ESP-IDF, terwijl de OTA-update van de lager geprivilegieerde gebruikersapplicatie mogelijk wordt gemaakt doordat ESP Privilege Separation een systeemaanroep voor de gebruikersapplicatie, `usr_esp_ota_user_app()`, beschikbaar stelt om het OTA-proces uit te voeren.
Laten we het OTA-voorbeeld van de gebruikersapp uit het ESP Priviledge Separation-project eens uitproberen.

Stap 0: vereisten

- ESP-IDF voor ESP Privilege Separation moet zijn ingesteld voordat er een voorbeeld wordt gebouwd
- een openbare URL van de gehoste image van de nieuwe gebruikersapp

Stap 1: het *rmaker_switch* voorbeeld uitvoeren

- Bouw van het voorbeeld

```
cd /examples/esp_user_ota
idf.py set-target esp32c3
idf.py build
```

- Het voorbeeld flashen en uitvoeren

```
idf.py flash monitor
```

Stap 2: de OTA-update starten

Voer de OTA-URL in omdat de gebruikersapp de OTA-URL accepteert met het commando `user-ota` via de console. Dit initieert een OTA en de nieuwe gebruikersapp start op zodra de OTA is voltooid. Nu hebben we ons connected apparaat met de belangrijkste functies klaar om te implementeren.
Je kunt de QR-code scannen om het ESP Privilege Separation-voorbeeld te bekijken. Dit voorbeeld demonstreert een echt praktijkgeval van een OTA-update van een gebruikersapp met behulp van ESP Rainmaker en ESP Privilege Separation.

Zoals gezegd wordt het succes van elk product door de gebruikers bepaald, en dat zijn jullie! We horen graag wat je ervan vindt, dus deel je ervaringen, suggesties en beoordelingen met ons. Je feedback voedt onze innovatie en helpt ons om ons product te verfijnen zodat het beter aan je behoeften voldoet. Bezoek onze website [1], stuur ons een e-mail, maak contact met ons op sociale media om je inzichten te delen of open een nieuw probleem in de GitHub-repository van het project [2]. Als je een specifieke wens hebt waarvan je denkt dat die goed in dit framework past, of als het oplossen van dergelijke problemen je enthousiast maakt, bespreken we een eventuele samenwerking graag met je!

230598-03

Vragen of opmerkingen?

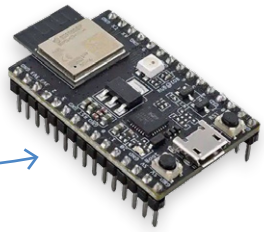
Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur via harshal.patil@espressif.com of naar de redactie van Elektor via redactie@elektor.com.

Over de auteur

Harshal Patil werkt sinds een jaar als software-engineer bij Espressif Systems. Hij is gepassioneerd over het oplossen van echte problemen door veilige, verbonden systemen te bouwen. Buiten zijn werk verkent hij graag nieuwe innovatieve gadgets en speelt hij voetbal.

Gerelateerde producten

- **ESP32-C3-DevKitM-1**
  www.elektor.nl/20324
- **ESP32-S3-DevKitC-1**
  www.elektor.nl/20697



WEBLINKS

[1] Espressif: https://espressif.com
[2] De project-repository: https://github.com/espressif/esp-privilege-separation/issues