yocto PROJECT

Formation développement Linux embarqué avec Yocto Project et OpenEmbedded

Durée de la formation

3 jours – 24 h

Langue

Transparents	Anglais
Présentation	Français
	Anglais
	Portugais
	Italien

Formateur

Un des ingénieurs suivants

- Alexandre Belloni
- Antonin Godard
- Jérémie Dautheribes
- João Marcos Costa
- Luca Ceresoli

Contact

training@bootlin.com
+33 4 84 25 80 96

bootlin
bootlin.com

Formation en présentiel

Public visé

Sociétés et ingénieurs intéressés par l'utilisation de Yocto Project pour construire leur système Linux embarqué.

Objectifs opérationnels

- Être capable de comprendre le rôle et le principe d'un build system Linux embarqué, et comparer Yocto Project/OpenEmbedded aux autres outils offrant des fonctionnalités similaires.
- Être capable de configurer et de réaliser la compilation d'un système Linux embarqué simple avec Yocto, et d'installer le résultat sur une plateforme embarquée.
- Être capable d'écrire ou d'étendre des recettes de paquets, pour vos propres paquets ou personnalisations.
- Être capable d'utiliser des *layers* de recettes existants, et de créer votre propre nouveau *layer*.
- Être capable d'intégrer le support pour votre carte embarqué dans un *layer BSP*.
- Être capable de créer des images personnalisées.
- Être capable d'utiliser le SDK du Yocto Project pour développer des applications.
- Être capable d'utiliser devtool pour développer une recette.

Prérequis

- **Connaissance et pratique des commandes UNIX ou GNU/Linux** : les participants doivent être à l'aise avec l'utilisation de la ligne de commande Linux. Les participants manquant d'expérience sur ce sujet doivent se former par eux-mêmes, par exemple en utilisant nos supports de formation.
- **Expérience minimale en développement Linux embarqué** : les participants doivent avoir une compréhension minimale de l'architecture d'un système Linux embarqué : rôle du noyau Linux par rapport à l'espace utilisateur, développement d'applications espace utilisateur en C. Suivre la formation Linux embarqué de Bootlin permet de remplir ce pré-requis.
- **Niveau minimal requis en anglais : B1**, d'après le *Common European Framework of References for Languages*, pour nos sessions animées en anglais. Voir la grille CEFR pour une auto-évaluation.

Méthodes pédagogiques

- Présentations animées par le formateur : 40% de la durée de formation
- Travaux pratiques réalisés par les participants : 60% de la durée de formation
- Version électroniques de supports de présentation, des instructions et des données de travaux pratiques. Les supports sont librement disponibles ici.

Modalités d'évaluation

Seuls les participants qui auront assisté à l'intégralité des journées de formation, et qui auront obtenu plus de 50% de réponses correctes à l'évaluation finale recevront une attestation individuelle de formation de la part de Bootlin.

Handicap

Les participants en situation de handicap qui ont des besoins spécifiques sont invités à nous contacter à l'adresse *training@bootlin.com* afin de discuter des adaptations nécessaires à la formation.

Équipement nécessaire

Pour les sessions en présentiel délivrées chez nos clients, notre client devra fournir :

- Projecteur vidéo
- Un ordinateur sur chaque bureau (pour une ou deux personnes), avec au moins 16 Go de RAM et Ubuntu Linux 24.04 installé dans une partition dédiée d'au moins 30 Go.
- Les distributions autres que Ubuntu Linux 24.04 ne sont pas supportées, et l'utilisation de Linux dans une machine virtuelle n'est également pas supportée.
- Connexion à Internet rapide et sans filtrage : au moins 50 Mbit/s de bande passante en téléchargement, et pas de filtrage des sites Web et protocoles.
- Les ordinateurs contenant des données importantes doivent être sauvegardés avant d'être utilisés dans nos sessions.

Pour les sessions en présentiel organisés dans les locaux de Bootlin, Bootlin fournit l'ensemble de l'équipement.

Plateforme matérielle pour les travaux pratiques

Plateforme STM32MP1

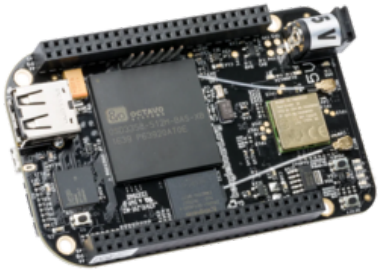
Une de ces cartes de STMicroelectronics : **STM32MP157A-DK1**, **STM32MP157D-DK1**, **STM32MP157C-DK2** ou **STM32MP157F-DK2**

- Processeur STM32MP157, double Cortex-A7, de STMicroelectronics
- Alimentée par USB
- 512 Mo DDR3L RAM
- Port Gigabit Ethernet port
- 4 ports hôte USB 2.0
- 1 port USB-C OTG
- 1 connecteur Micro SD
- Debugger ST-LINK/V2-1 sur la carte
- Connecteurs compatibles Arduino Uno v3
- Codec audio
- Divers : boutons, LEDs
- Écran LCD tactile (uniquement sur cartes DK2)

BeagleBone Black

Carte **BeagleBone Black** ou **BeagleBone Black Wireless**

- Un processeur ARM AM335x de Texas Instruments (à base de Cortex-A8), avec accélération 3D, etc.
- 512 Mo de RAM
- 2 ou 4 Go de stockage eMMC
- USB hôte et device
- Sortie HDMI
- Connecteurs à 2 x 46 broches, pour accéder aux UARTs, aux bus SPI, aux bus I2C, et à d'autres entrées/sorties du processeur.
- Ethernet ou WiFi



BeaglePlay

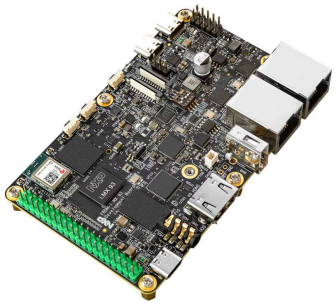
Carte **BeaglePlay**

- SoC Texas Instruments AM625x (CPU 4xARM Cortex-A53)
- SoC avec accélération 3D, MCU intégré et de nombreux autres périphériques.
- 2 GB de RAM
- 16 Go de stockage eMMC
- USB hôte et device, microSD, HDMI
- WiFi 2.4 and 5 GHz, Bluetooth et aussi Ethernet
- 1 Header MicroBus (SPI, I2C, UART, ...), connecteurs OLDI et CSI.

NXP i.MX93 FRDM

Carte de développement **NXP FRDM-IMX93**

- Processeur NXP i.MX93 (Dual Cortex-A55 + Cortex-M33)
- 2 GB LPDDR4X, 32 GB eMMC
- Double Ethernet Gigabit
- USB 2.0 Type-C + USB Type-A
- Interface CAN
- Slot microSD, EEPROM
- Wi-Fi 6 + Bluetooth 5.4 + 802.15.4 (MAYA-W276)
- Sortie HDMI (via LVDS), MIPI DSI et CSI
- Audio jack (MQS), buttons and LEDs
- debug via SWD et UART



Programme de la formation

Jour 1 - Matin

Cours	Introduction aux outils de compilation de systèmes Linux embarqué	▪ Vue d'ensemble de l'architecture d'un système Linux embarqué ▪ Méthodes pour compiler un système de fichiers ▪ Utilité des outils de compilation
Cours	Vue d'ensemble de Yocto Project et du système de référence Poky	▪ Présentation de l'outil de compilation Yocto / OpenEmbedded et de sa terminologie. ▪ Vue d'ensemble du système de référence Poky
Cours	Bases de l'utilisation de Yocto Project	▪ Mise en place du répertoire de travail et de l'environnement ▪ Configuration de l'outil de compilation ▪ Compilation de l'image d'un système de fichiers racine ▪ Structure des fichiers générés
TP	1ère compilation avec Yocto Project	▪ Téléchargement du système de référence Poky ▪ Configuration de l'outil de compilation ▪ Compilation d'une image système

Jour 1 - Après-midi

TP	Flasher et booter	▪ Flasher et booter l'image du système sur la carte
Cours	Utilisation de Yocto Project - Utilisation avancée	▪ Assignation des variables, opérateurs et surcharge ▪ Paquetages : variantes de paquetages ▪ Options de la commande bitbake
TP	Utilisation de NFS et configuration de la compilation	▪ Configurer la carte pour démarrer via NFS ▪ Rajouter un paquetage au système de fichiers racine ▪ Apprendre à utiliser le mécanisme `PREFERRED_PROVIDER` ▪ Se familiariser avec les options de la commande bitbake

Jour 2 - Matin

Cours	Écriture de recettes - Fonctionnalités de base	▪ Recettes : vue d'ensemble ▪ Organisation d'un fichier de recette ▪ Application de patches ▪ Exemples de recettes
TP	Ajouter la compilation d'une application	▪ Création d'une recette pour *ninvaders* ▪ Résolution de problèmes liés à la recette ▪ Résolution de problèmes liés à la compilation croisée ▪ Ajout d'*ninvaders* à l'image finale
Cours	Écriture de recettes - Fonctionnalités avancées	▪ Extension et redéfinition de recettes ▪ Paquetages virtuels ▪ Familiarisation avec les classes ▪ Inclusion d'exemples avec BitBake ▪ Mise au point des recettes ▪ Configuration de l'utilisation du réseau par BitBake

Jour 2 - Après-midi

Cours	Layers	▪ Ce que sont les *layers* ▪ Où trouver les *layers* ▪ Création d'un *layer*

TP	Écriture d'un layer	▪ Apprendre à écrire un *layer* ▪ Ajouter le *layer* à la compilation ▪ Inclure *ninvaders* dans le nouveau *layer*

Jour 3 - Matin

TP	Étendre une recette	▪ Étendre la recette pour le noyau pour rajouter des patches ▪ Configurer le noyau pour compiler le pilote du nunchuk ▪ Modifier la recette ninvaders pour rajouter des patches ▪ Jouer avec *ninvaders*
Cours	Écriture d'un BSP	▪ Introduction aux layers BSP ▪ Ajout d'une nouvelle machine ▪ Configuration du chargeur de démarrage ▪ Linux : la classe kernel.bbclass et la recette linux-yocto
TP	Création d'une configuration spécifique pour une machine	▪ Types d'images ▪ Écriture et utilisation de groupes de paquetages
Cours	Création d'une image sur mesure	▪ Rajouter une recette de base pour une image ▪ Sélectionner les fonctionnalités et les paquetages de l'image ▪ Ajouter un groupe de paquetages sur mesure ▪ Ajouter une variante d'image pour le débogage

Jour 3 - Après-midi

Cours	Images	▪ Écriture d'une recette d'image ▪ Types d'images ▪ Écriture et utilisation de groupes de paquetages
TP	Création d'une image sur mesure	▪ Écrire une recette d'image personnalisée ▪ Ajouter *ninvaders* à l'image sur mesure
Cours	Écriture de recettes - Pour aller plus loin	▪ Le sysroot de chaque recette ▪ Utilisation de code Python code dans les méta-données ▪ Drapeaux de variables ▪ Fonctionnalités de paquetages et PACKAGECONFIG ▪ Fonctionnalités conditionnelles ▪ Découpage de paquetages
Cours	Licences	▪ Gestion de licences open source
Cours	SDK pour le projet Yocto	▪ Objectifs du SDK ▪ Compilation et personnalisation d'un SDK ▪ Utilisation d'un SDK pour le projet Yocto
TP	Développement d'une application au moyen du SDK de Poky	▪ Construction d'un SDK ▪ Utilisation du SDK de Yocto Project
Cours	Devtool	▪ Présentation devtool ▪ Cas d'utilisation de devtool
TP	Utilisation de devtool	▪ Création d'une nouvelle recette ▪ Modification d'une recette pour ajouter un nouveau patch ▪ Mettre à jour une recette pour prendre en charge une nouvelle version

Cours	Gestion automatique de layers	▪ Gestion automatique de layers
Cours	Gestion de paquetages à l'exécution	▪ Introduction à la gestion de paquetages à l'exécution ▪ Configuration de la compilation ▪ Configuration d'une serveur de paquetages ▪ Configuration de la machine cible