Le développement d'applications Qt pour des systèmes Linux embarqués est désormais plus facile avec Qt Commercial 4.8 SDK

Par Samuli Piippo - Nanzilla - Digia Qt

Date de publication : 8 mars 2012

Dernière mise à jour : 8 avril 2020

Avec la sortie de la première version de Qt Commercial 4.8 SDK, vous allez enfin pouvoir apprécier les améliorations que nous avons réalisées pour le développement sous Linux embarqué. Le Qt Commercial SDK contient désormais tout ce dont vous avez besoin pour démarrer des développements Qt sur des cartes de développement de systèmes embarqués populaires telles que BeagleBoard-xM ou PandaBoard. Un outil de compilation croisée est inclus, ainsi que des bibliothèques Qt précompilées et les images rootfs appropriées pour les deux systèmes embarqués. Même en cas de différences matérielles, ces améliorations de Qt Creator Commercial seront utiles.

Commentez

I - L'article original..... 3
II - Le développement d'applications Qt pour des systèmes Linux embarqués est désormais plus facile avec Qt Commercial 4.8 SDK..... 4
III - Qu'est-ce que cette nouvelle version avec le support de Linux embarqué apporte ?..... 6
IV - Conclusion..... 7
V - Remerciements..... 8

I - L'article original

Cet article est une adaptation en langue française de **Embedded Linux Development Just Got Easier With with Qt Commercial 4.8 SDK**, de Samuli Piippo.

II - Le développement d'applications Qt pour des systèmes Linux embarqués est désormais plus facile avec Qt Commercial 4.8 SDK

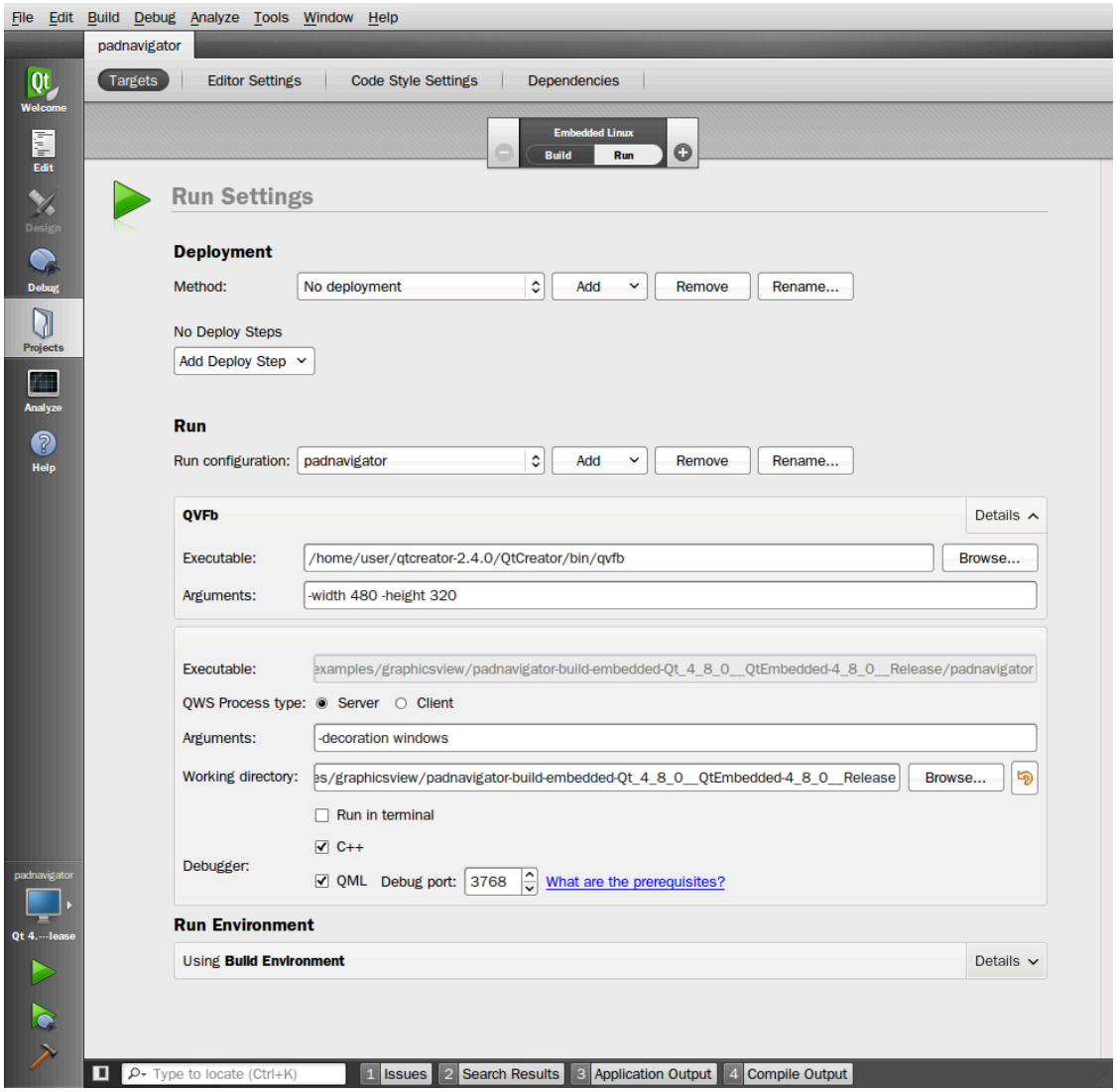
Après l'été, nous avons commencé à jouer avec la distribution **Ångström** et le framework **OpenEmbedded** pour voir s'il était possible de créer quelque chose pour faciliter le démarrage de développements Qt sur des systèmes embarqués.

Après le test de différentes cartes de développement, il a été décidé de poursuivre avec les cartes **Beagle** et **Panda**, un planning fut mis en place. Maintenant, le SDK fournit une option pour installer un outil de compilation croisée avec le framework OpenEmbedded et basé sur la distribution Ångström. Avec ce nouveau SDK et Qt Commercial Creator 2.4.0, il est maintenant aisé de créer des applications Qt pour des systèmes Linux embarqués fonctionnant sur une plateforme ARM.

Pour que les applications fonctionnent sur un tel dispositif, on peut utiliser l'une des images prêtes à l'emploi pour BeagleBoard-xM ou PandaBoard. Comme stipulé précédemment, il s'agit de distributions Ångström basées sur des paquets rootfs, avec de petits ajouts pour une intégration plus souple dans Qt Commercial Creator. Il suffit de déposer les images sur une carte mémoire avant de déployer les applications sur les dispositifs cibles. Les images contiennent déjà une version de Qt Commercial 4.8.0, configurée pour utiliser QWS en plus du framebuffer de Linux. Les images sont configurées avec des paquets issus du dépôt Ångström, extensibles simplement si nécessaire.

Un autre exemple d'image est aussi disponible, avec seulement les paquets nécessaires au démarrage de la carte avec la distribution Linux embarquée et les démos Qt. L'image complète nécessite maintenant 27 Mo, ce qui inclut le système Linux embarqué et Qt 4.8. Il est possible de l'alléger en n'incluant que les bibliothèques Qt requises par l'application. Une version plus légère de Qt, ne contenant que les fonctionnalités nécessaires au bon fonctionnement de l'application, est sur l'établi.

Avec le nouveau Qt Commercial Creator 2.4.0, le développement d'applications Qt pour les systèmes Linux embarqués est plus simple que jamais, grâce aux nouvelles cibles. Il est disponible dans le Qt Commercial SDK, fournissant déjà Qt 4.8.0 pour les plateformes Linux embarquées et les outils ARM installés et correctement configurés. Pour commencer de nouveaux projets pour Linux embarqué, il suffit ainsi de sélectionner le modèle Linux embarqué ou bien de l'ajouter dans des projets existants.


File Edit Build Debug Analyze Tools Window Help
padnavigator
Targets
Editor Settings
Code Style Settings
Dependencies
Embedded Linux
Build
Run
Welcome
Edit
Design
Debug
Projects
Analyze
Help
Run Settings
Deployment
Method:
No deployment
Add
Remove
Rename...
No Deploy Steps
Add Deploy Step
Run
Run configuration:
padnavigator
QVFb
Details
Executable:
/home/user/qtcreator-2.4.0/QtCreator/bin/qvfb
Browse...
Arguments:
-width 480 -height 320
Executable:
examples/graphicsview/padnavigator-build-embedded-Qt_4_8_0__QtEmbedded-4_8_0__Release/padnavigator
QWS Process type:
Server
Client
Arguments:
-decoration windows
Working directory:
es/graphicsview/padnavigator-build-embedded-Qt_4_8_0__QtEmbedded-4_8_0__Release
Run in terminal
Debugger:
C++
QML
Debug port:
3768
What are the prerequisites?
Run Environment
Using Build Environment
Type to locate (Ctrl+K)
1 Issues
2 Search Results
3 Application Output
4 Compile Output

III - Qu'est-ce que cette nouvelle version avec le support de Linux embarqué apporte ?

- La reconnaissance correcte de Qt pour les versions de Linux embarquées.
- L'intégration des fonctionnalités du Qt Virtual Framebuffer (QVFb) directement dans l'IDE Qt Commercial Creator (plus besoin de le gérer séparément).
- L'intégration de l'exécution, du débogage et de l'analyse des applications QWS sur les plateformes Linux hôtes.
- Toutes les options de configuration de QVFb peuvent être maintenant spécifiées par projet ; ainsi, il n'y a plus besoin de manipulation supplémentaire pour les gérer séparément en dehors des projets. De manière analogue, les options de configuration pour les applications QWS peuvent également être spécifiées par projet.

Les options de débogage et d'analyse des applications QWS écrites en C++ ou en QML sont disponibles aussi bien sur l'hôte Linux local que sur les périphériques Linux embarqués distants.

Plus généralement, cette nouvelle fonctionnalité a pour but de faciliter le développement sur des plateformes Linux embarquées avec Qt Commercial.

IV - Conclusion

Ce sont les trois principaux problèmes à surveiller, mais d'autres choses doivent être surveillées (voir la documentation : **Le support des threads avec Qt**).

V - Remerciements

Un grand merci à **Thibaut Cuvelier** pour sa relecture très attentive et ses conseils. Merci à **Jacques Thery** pour sa relecture orthographique.