DEVELOPER DAYS 2014 EUROPE

Device Creation with Qt Enterprise Embedded

Andy Nichols

Overview

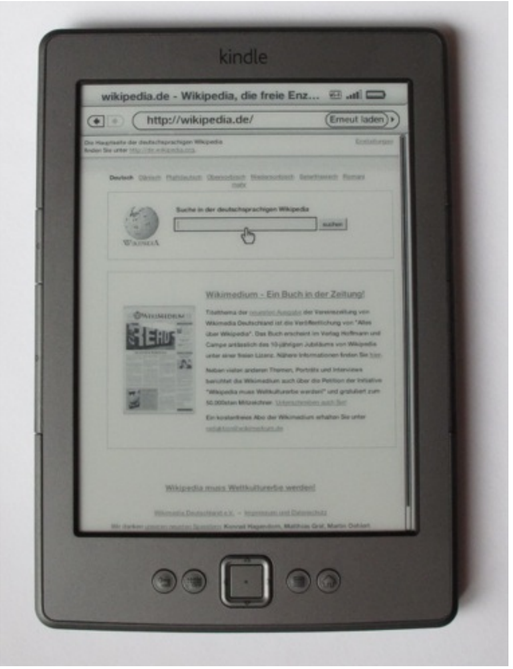
- The challenges of device creation
- What is Qt Enterprise Embedded
- Prototyping a device
- Device creation

The Challenges of Device Creation

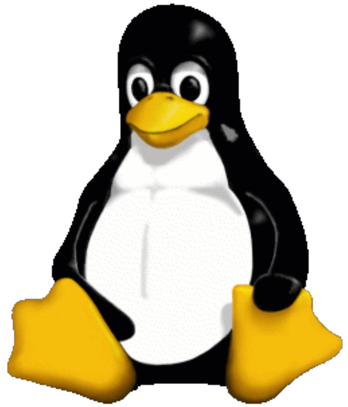
The Application is just one step

- Qt is used to more easily produce Applications
- Easy to share code across multiple platforms

The Application dictates the hardware



The Application and Hardware dictate the OS



Provide your own SDK

- Toolchain to generate binaries for your hardware
- System image to flash to device
- Sysroot containing the development files

Device Creation with Qt

- Qt abstracts away the details of the OS and hardware
- It is still up to you to provide the platform

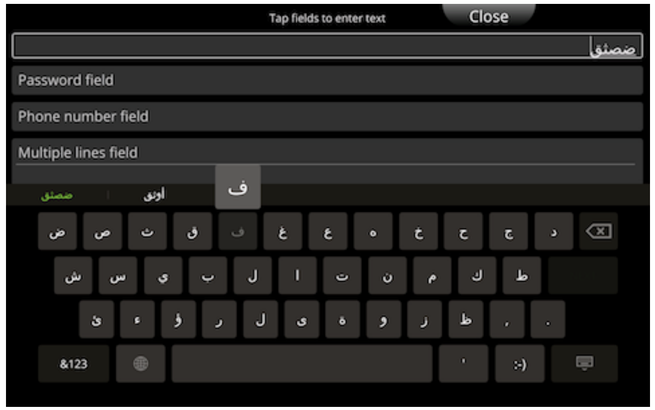
What is Qt Enterprise Embedded?

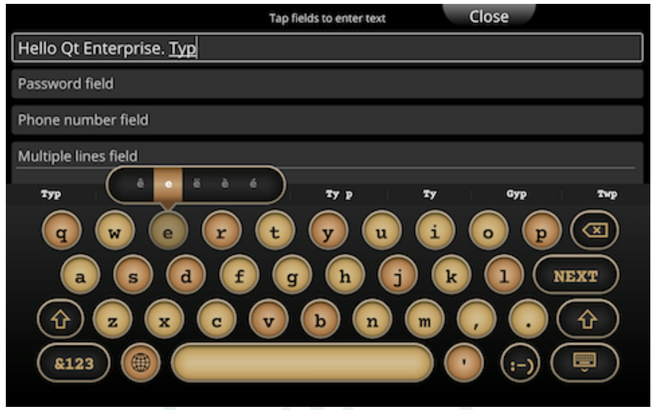
- **Qt**
- Platform
- Tooling

Qt + Additional tools for Device Creators

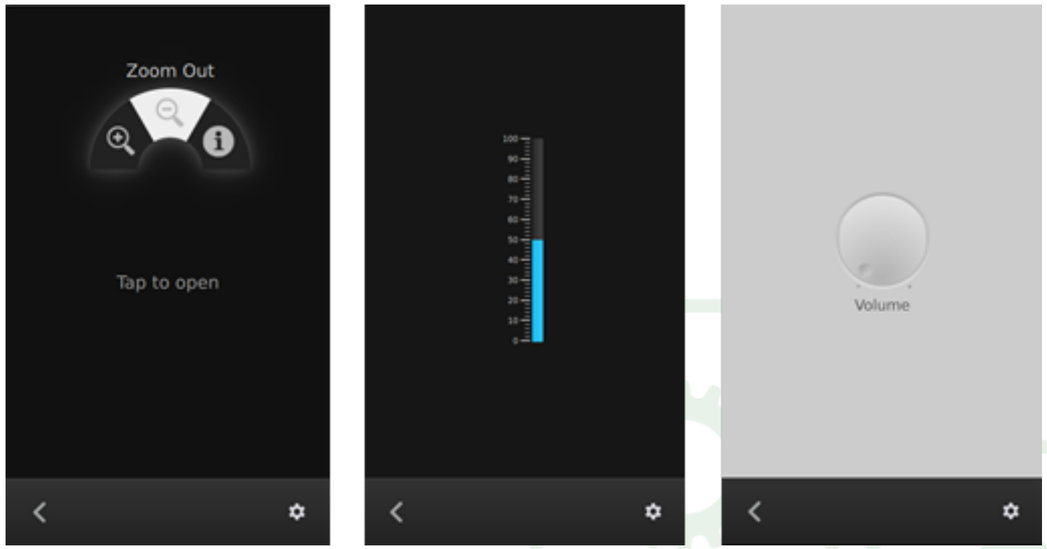
- Latest Qt Release
- Virtual Keyboard
- Enterprise Qt Quick Controls
- Wifi Utilities
- Charts API
- 3D Data Visualisation
- Qt WebEngine
- Qt Quick Compiler

Virtual Keyboard

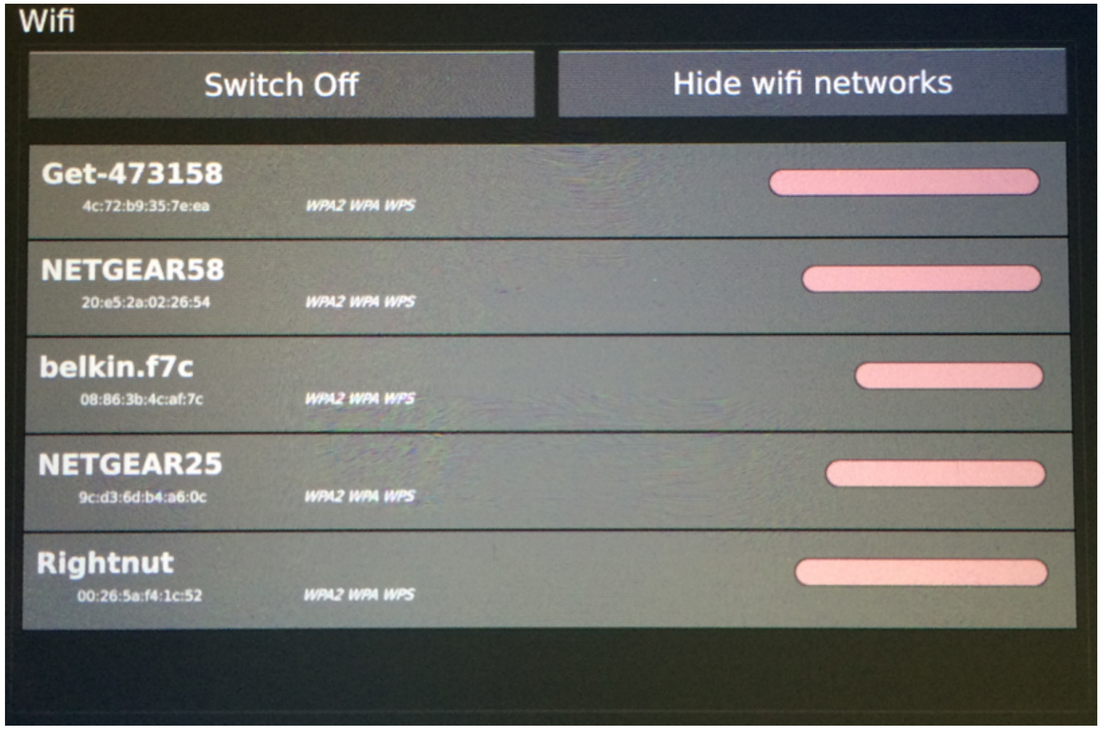




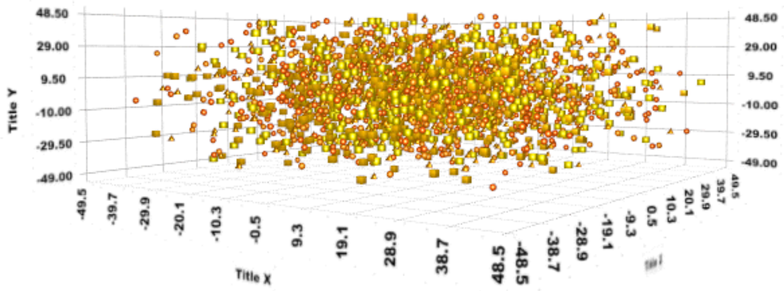
Enterprise Qt Quick Controls



Wifi Utilities



Charts and 3D Data Visualisation APIs

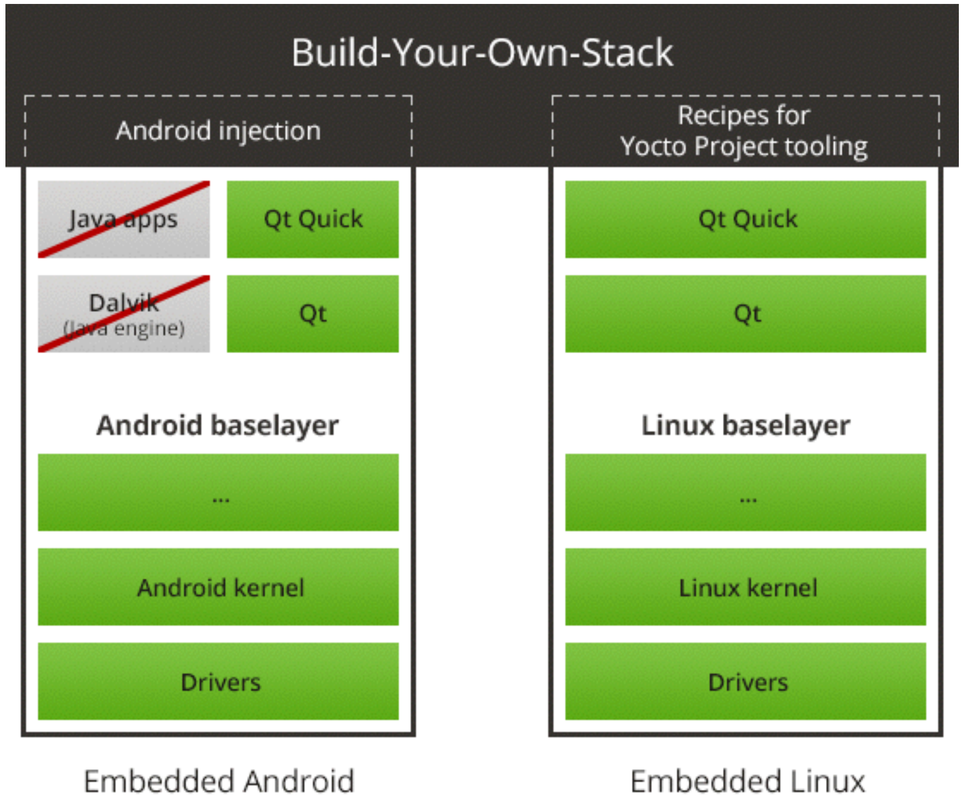


Qt WebEngine

What is Qt Enterprise Embedded?

- Qt
- **Platform**
- Tooling

Qt Enterprise Embedded provides the platform



Embedded Linux Platform

- Using Yocto Project tools
- b2qt-meta layer to provide barebones distro for Qt 5
- Yocto is the industry standard
- Many meta layers available for you to mix-n-match

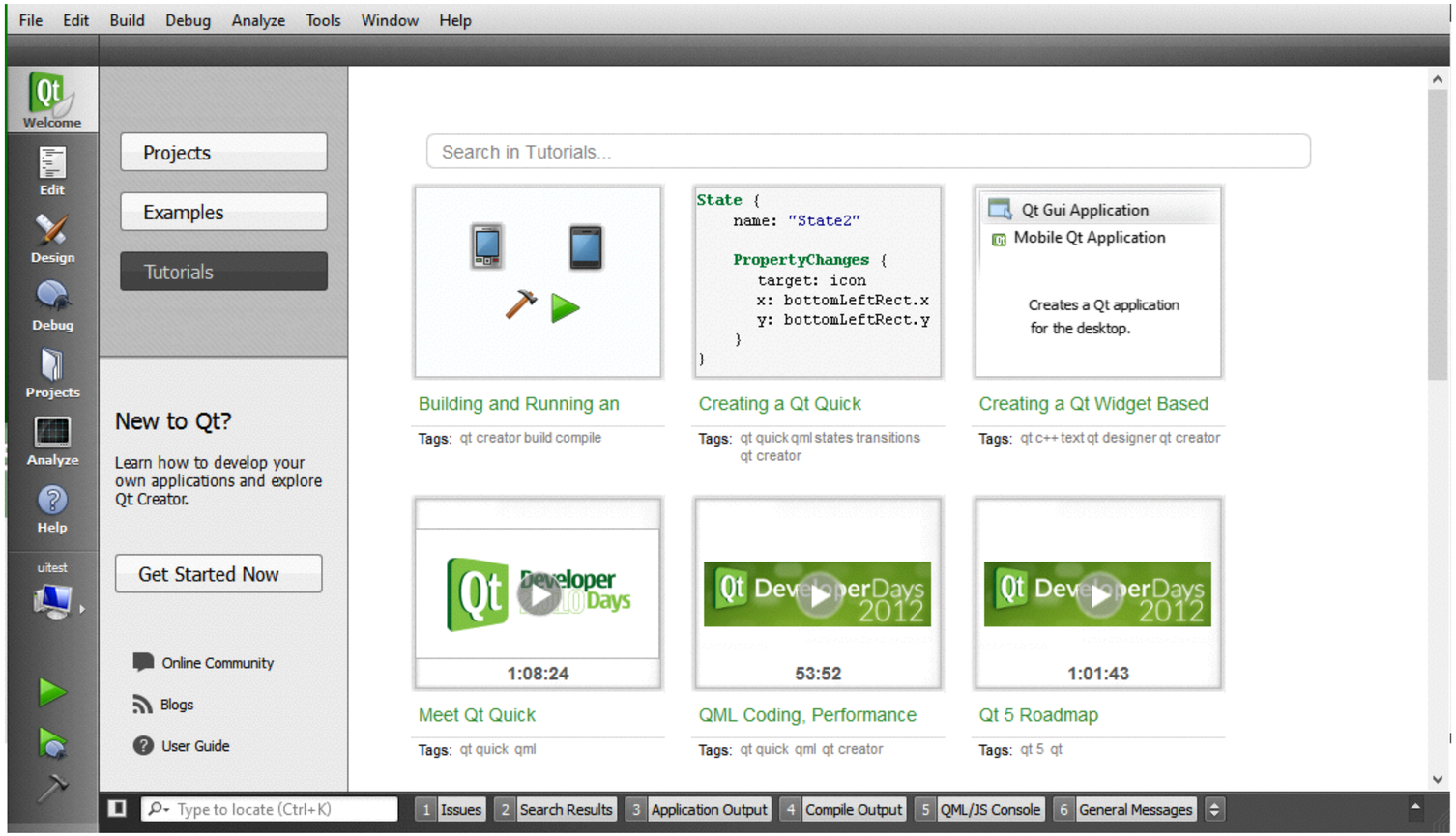
Embedded Android

- Android Injection
- No need to build the AOSP, use existing device images
- Android without Java, Using Platform Developers APIs
- Take advantage of a device that already fits your needs

What is Qt Enterprise Embedded?

- Qt
- Platform
- **Tooling**

Qt Creator IDE



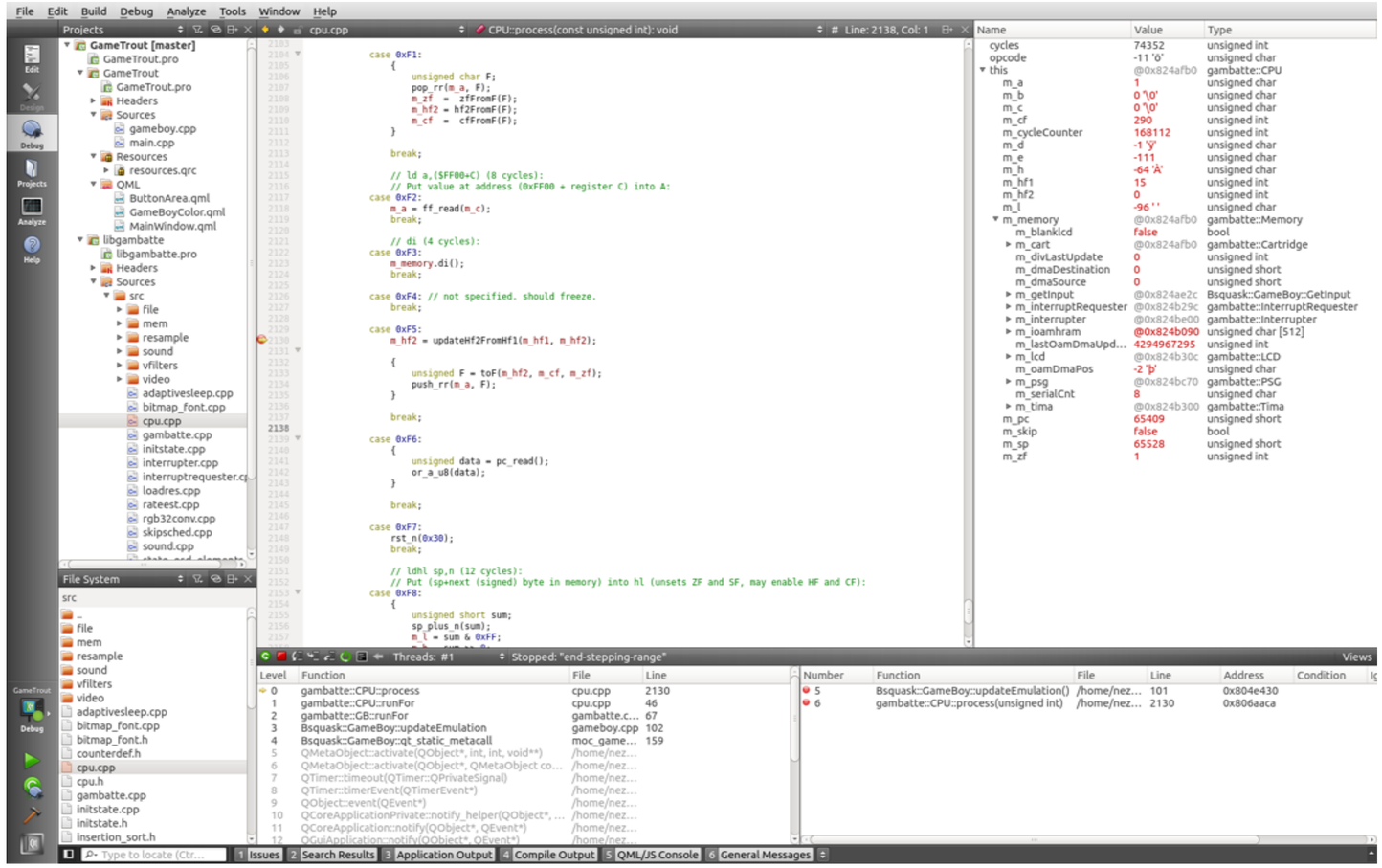
Pre-configured Kits for your Device

- Linux: Use the SDK built by Yocto
- Android: Just select the version of Android that is on your device

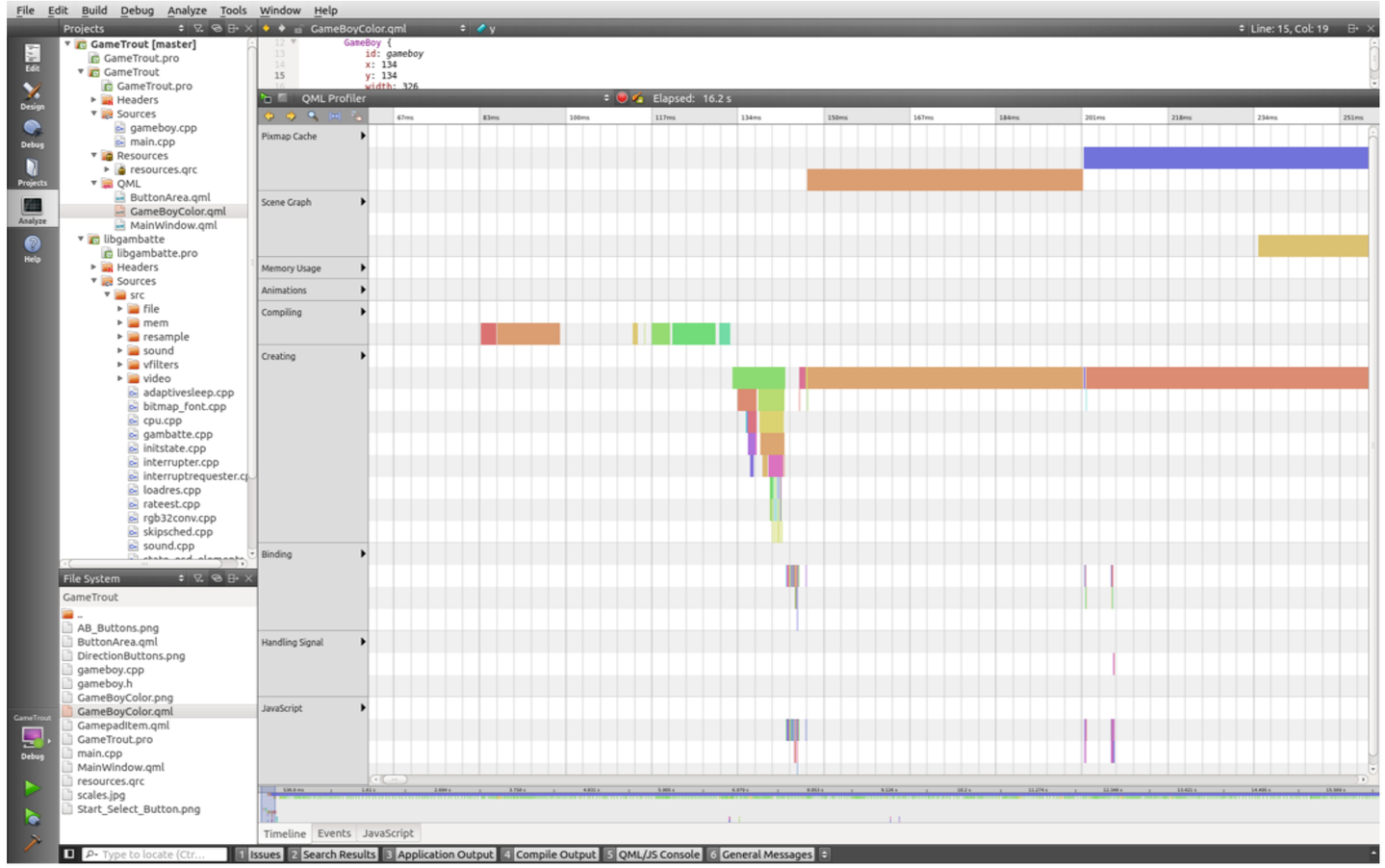
One Click Deploy

- USB
- Ethernet

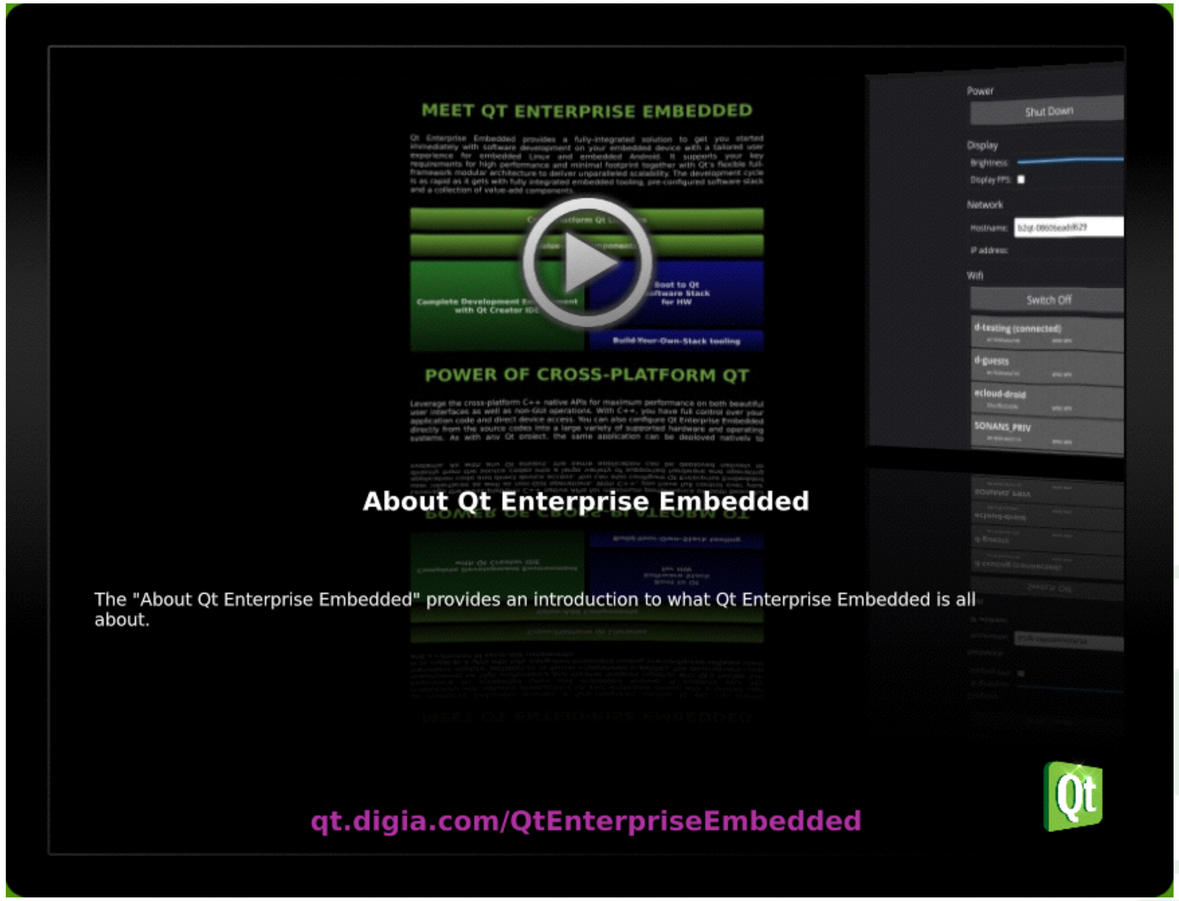
Remote Debugging of C++ and QML



Remote Profiling



Device Emulator



Prototyping a Device

Reference Hardware

- Pre-built Images
- SDKs
- Start prototyping on hardware from Day 1

Linux Reference Hardware

- Beaglebone Black
- Raspberry Pi
- Emulator
- Toradex Apalis iMX6
- Freescale Sabre
- Boundary Devices Nitrogen6x

Android Reference Devices

- Nexus 7 (2012/2013)
- Beaglebone Black
- Freescale Sabre
- Boundary Devices Nitrogen6x
- Emulator

Getting Started

- Install the SDK for the device from the Qt Installer
- Flash the image to the device with our deploy scripts
- Plug in the USB and launch your App from Qt Creator

Benefits of Prototyping on real hardware

- Test an SOC to see if its a good fit for your product
- Have a mock up to show your boss or VC
- The code can be reused for the actual product

Device Creation

Linux device SDK

- Use Yocto to generate base system image and SDK
- Build Qt, Add-ons, and B2Qt Utilities
- Register new SDK with Qt Creator

Using Yocto

- Base image and SDK for device
- Vendor specific meta-layers
- Define additional features and 3rd party libraries
- Modify the kernel to meet your needs

Build your own stack

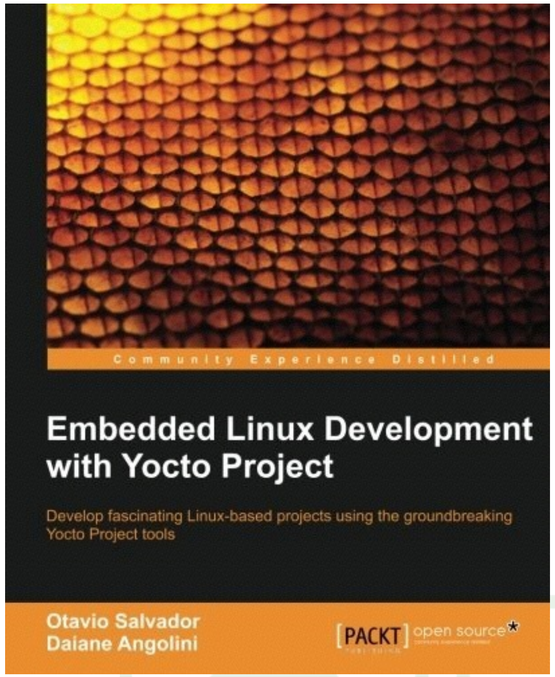
```
#Setup Yocto tools/repos
cd <BuildDir>
<INSTALL_DIR>/Boot2Qt-3.x/sources/b2qt-yocto-meta/b2qt-init-build-env .
export TEMPLATECONF=meta-b2qt/conf
export MACHINE=beaglebone
source poky/oe-init-build-env build-beaglebone

#build base image
bitbake b2qt-embedded-image

#build SDK (toolchain/sysroot)
bitbake meta-toolchain-b2qt-embedded-sdk
```

Need help with Yocto?

- Yocto User Manual
- Yocto eBook
- The Qt Company and its partners



Building Qt, Add-ons, B2Qt Utils

```
#Setup Build Environment
<INSTALL_DIR>/Boot2Qt-3.x/sources/b2qt-build-scripts/embedded-common/
init_build_env.sh <INSTALL_DIR>/Boot2Qt-3.x/sources/b2qt-build-scripts/embedded-
linux/config.beaglebone

#Build Qt Libraries
<INSTALL_DIR>/Boot2Qt-3.x/sources/b2qt-build-scripts/embedded-linux/build_qt.sh

#Build Add-ons and Utils
<INSTALL_DIR>/Boot2Qt-3.x/sources/b2qt-build-scripts/embedded-linux/build_extras.sh

#Generate new boot Image
<INSTALL_DIR>/Boot2Qt-3.x/sources/b2qt-build-scripts/embedded-linux/build_image.sh
```

Building Qt for custom machines

- config.${MACHINE}
- appcontroller.conf.${MACHINE}
- qt.conf.${MACHINE}

config.${MACHINE}

```
#beaglebone config
export PLATFORM="eLinux"
export MACHINE="beaglebone"
export DEVICE="linux-beaglebone-g++"
export ARCH="arm"
export SYSROOT="armv7ahf-vfp-neon-poky-linux-gnueabi"
export COMPILER="arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-"
```

appcontroller.conf.${MACHINE}

```
#beaglebone appcontroller environment
env=QML2_IMPORT_PATH=/data/user/qt/qmlplugins
env=QT_IM_MODULE=qtvirtualkeyboard
env=QT_QPA_EGLFS_FORCE888=0
base=linux
platform=beaglebone
```

qt.conf.${MACHINE}

```
#beaglebone qt.conf
[Paths]
Sysroot=../../toolchain/sysroots/armv7ahf-vfp-neon-poky-linux-gnueabi
Prefix=/bin/../../../../qt5
HostPrefix=..
```

Register new SDK with Qt Creator

```
#Register new Kit in Qt Creator
<INSTALL_DIR>/Boot2Qt-3.x/sources/b2qt-build-scripts/embedded-common/
setup_qtcreator.sh
```

Android Device SDK

- Android 4.2 or Android 4.4
- Root access
- Unlocked bootloader

Android Injection

- Modify init.rc
- Deploy payload based on Android version

Android Injection: Nexus 4

```
#Modify boot image
abootimg -x ../boot.img
mkdir initrd
cd initrd
cat ../initrd.img | gunzip | cpio -vid

#Modify the init.rc file using the sed file
sed -f <INSTALL_DIR>/Boot2Qt-3.x/generic-4.2-eAndroid/images/generic/modify_init_rc.sed -i
init.rc

#Repackage the boot.img
find . | cpio --create --format='newc' | gzip > ../myinitrd.img
abootimg --create myboot.img -f bootimg.cfg -k zImage -r myinitrd.img

#Flash the new boot.img to the device
fastboot flash boot myboot.img
fastboot reboot
```

Android Injection: Nexus 4

```
#Push Data Payload to the device
mkdir data
tar xf <INSTALL_DIR>/Qt/Boot2Qt-3.x/generic-4.2-eAndroid/images/data.tar.xz -C data/
find data -type d -exec adb shell mkdir -p /{} \;
adb push data/. /data 2>&1 | grep -v '\ ->\ '
#Push System Payload to the device
mkdir system
tar -xf <INSTALL_DIR>/Boot2Qt-3.x/generic-4.2-eAndroid/images/system.tar.xz -C
system/
cp <INSTALL_DIR>/Boot2Qt-3.x/generic-4.2-eAndroid/images/nexus7/appcontroller.conf .
cp appcontroller.conf system/bin/
cp <INSTALL_DIR>/Boot2Qt-3.x/generic-4.2-eAndroid/images/common/gdbserver system/bin/
adb remount
find system -type d -exec adb shell mkdir -p /{} \;
adb push system/. /system 2>&1 | grep -v '\ ->\ '
```

Demo Time!

Future Research

- Build Everything in Yocto
- Low end profile (DirectFB)
- Multi-process support (Wayland)

Questions?