Qt for Device Creation

László Agócs

@alpqr on Twitter / lagocs on Freenode

Senior Software Engineer

The Qt Company, Oslo, Norway

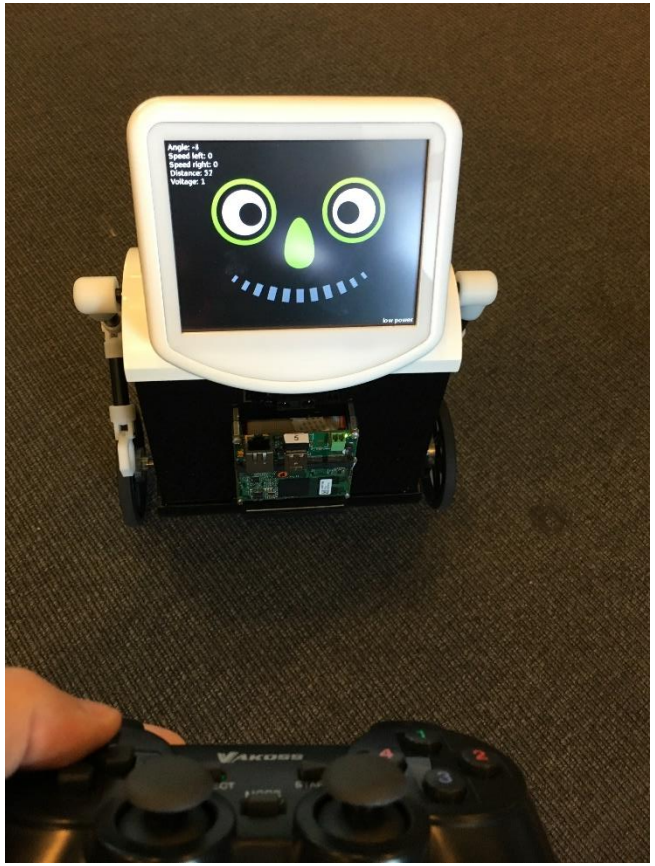
ECS2016 - 23 November 2016

Agenda

- The challenges of device creation
- What is Qt?
- How does Qt help in the embedded space?
- Getting started with Qt on Embedded Linux

What Kind of Devices?

- ARM, Intel or MIPS application processors
  - Typically ARM Cortex-A
- User interface
  - With or without touch or other types of input
- MCUs (e.g. ARM Cortex-M) are not in scope
  - Hybrid approaches work fine, though! Linux with Qt for UI & input on Cortex-A + custom RTOS on Cortex-M + interop…

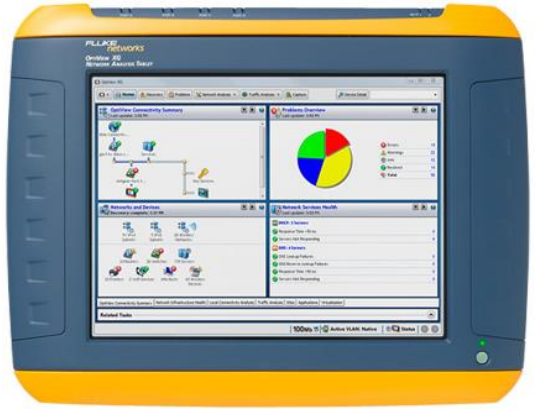


Embedded Development Boards

What Type of User Interfaces?

Automotive IVI

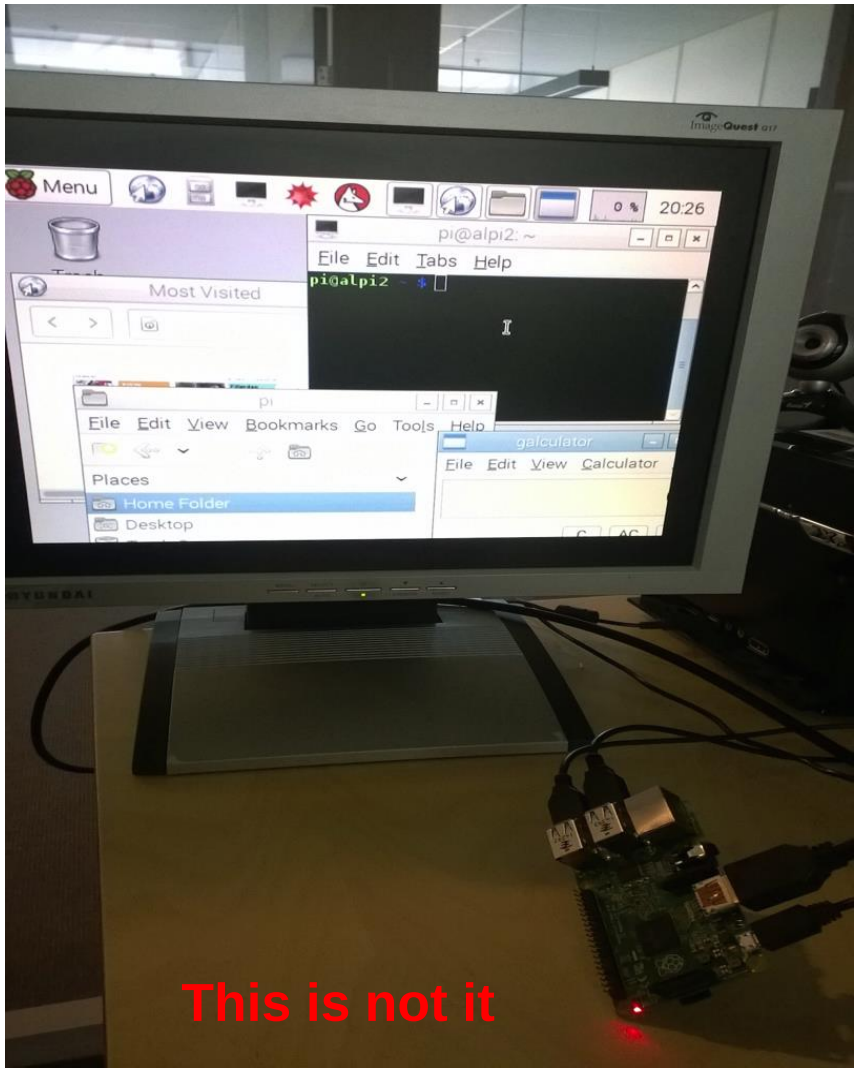
Refrigerators & Coffee Machines

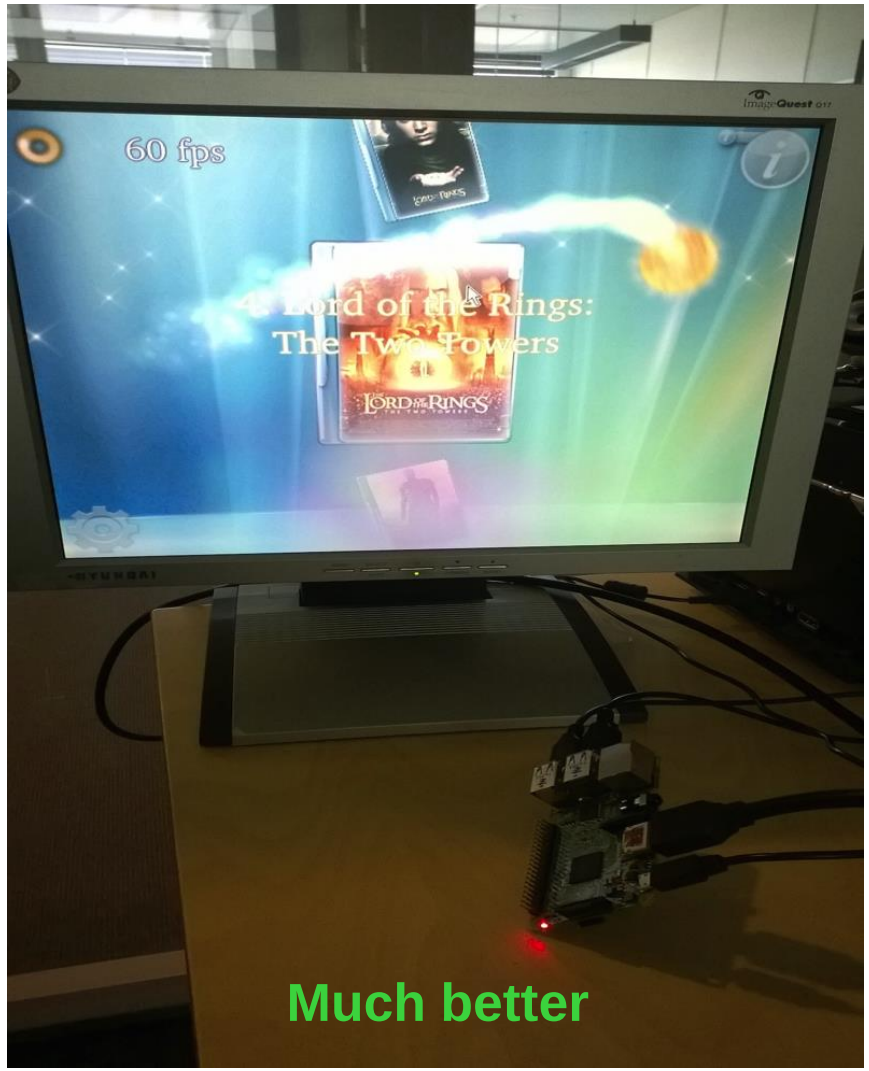


Network Analyzers

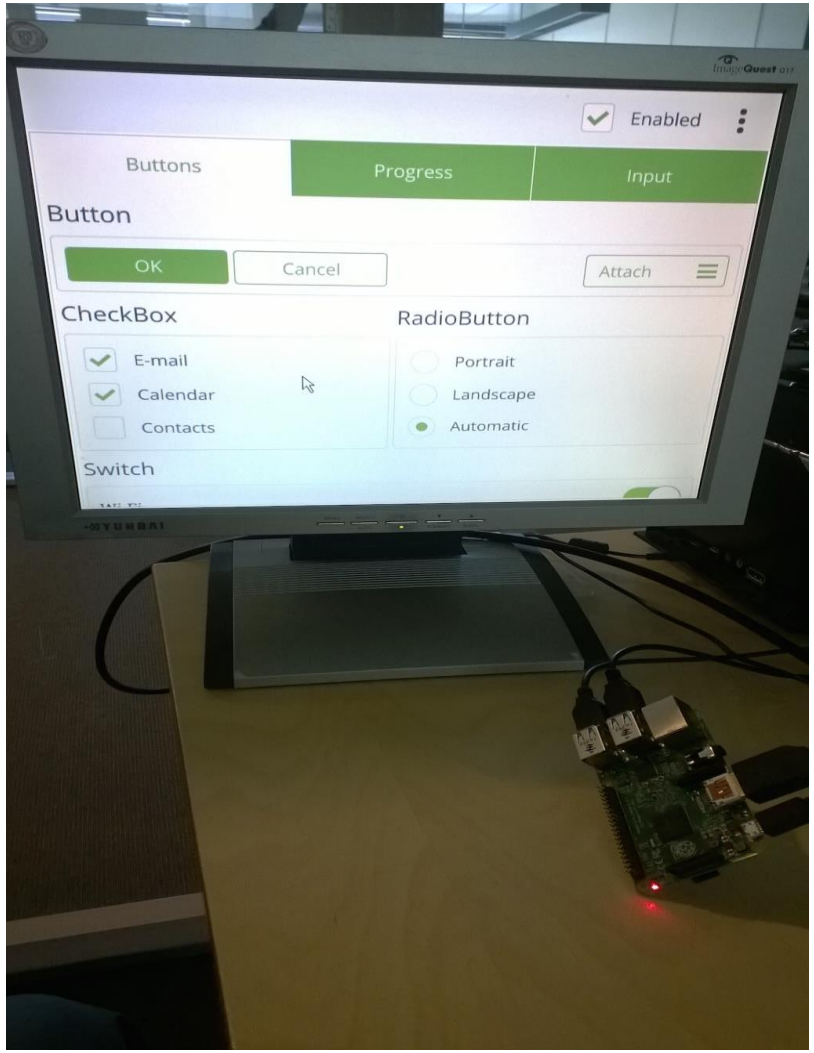
- Automotive, Maritime
- Medical Devices
- Home Automation
- Digital Photo Frames
- Set Top Boxes
- In-Flight Entertainment
- Industrial Control
- …

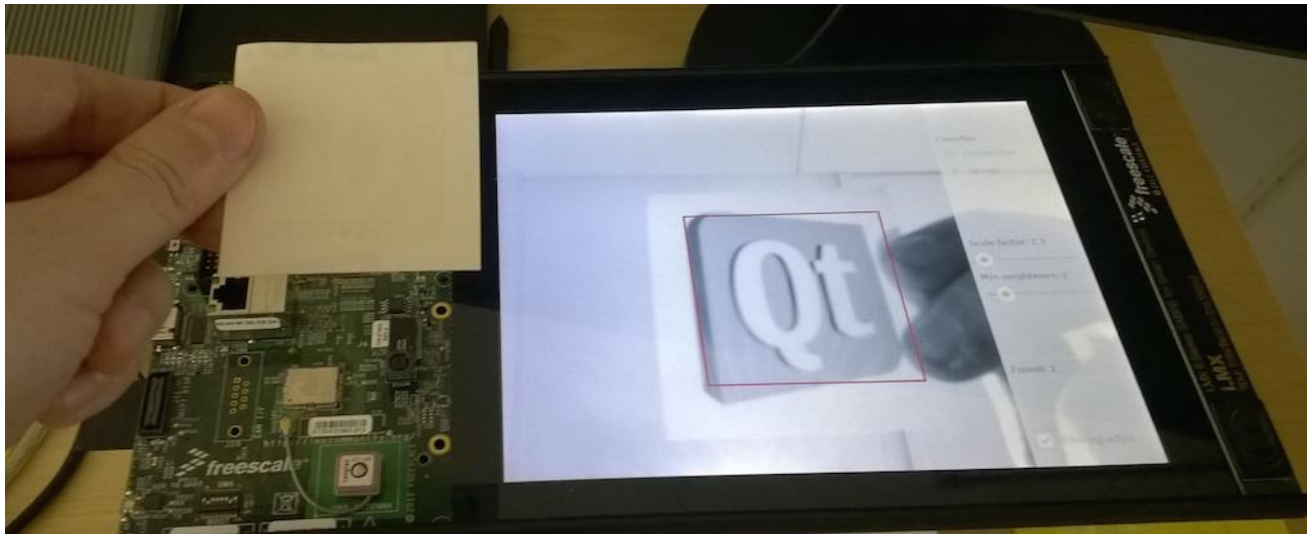
What Type of User Interfaces?



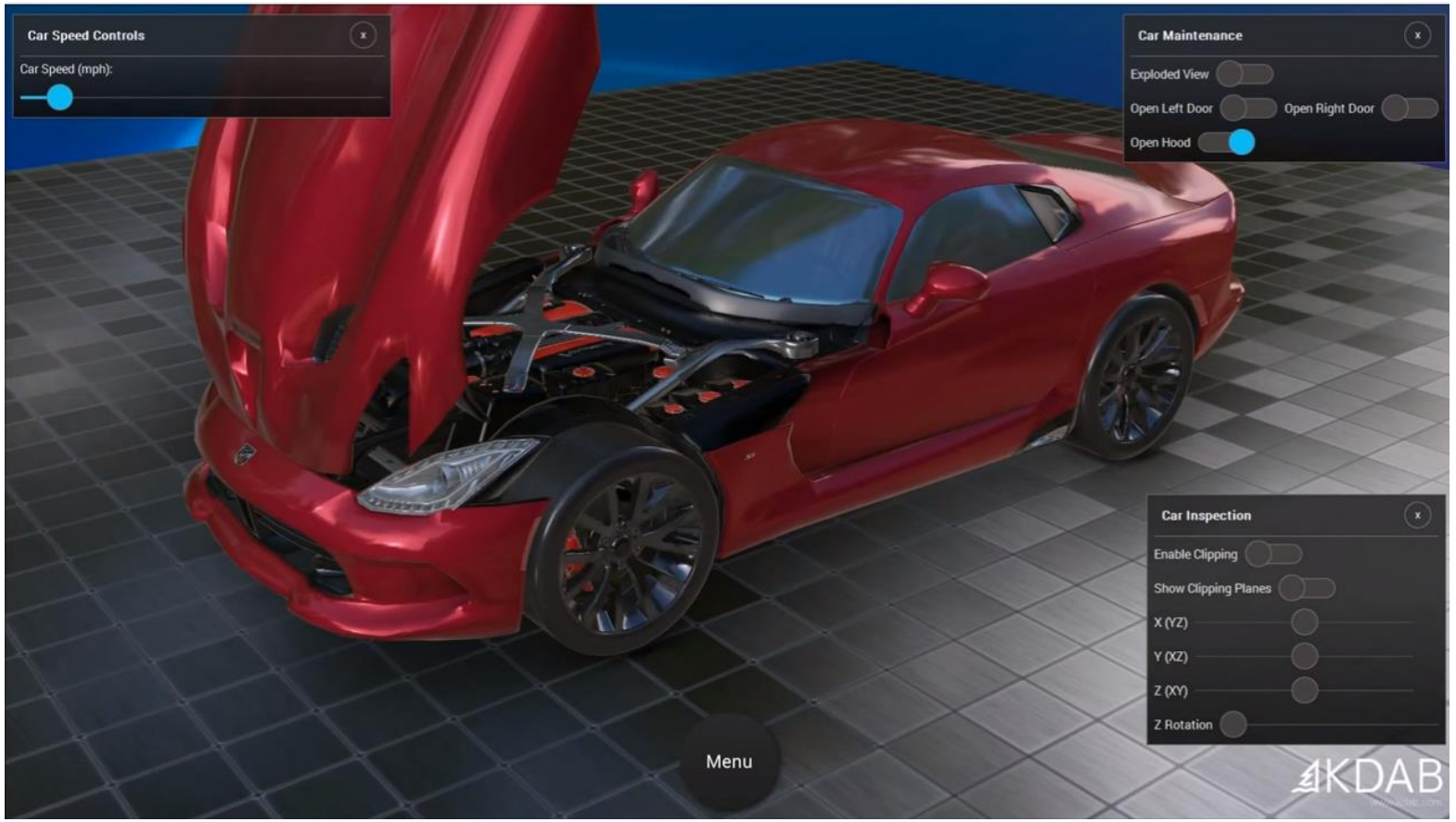


What Type of User Interfaces?



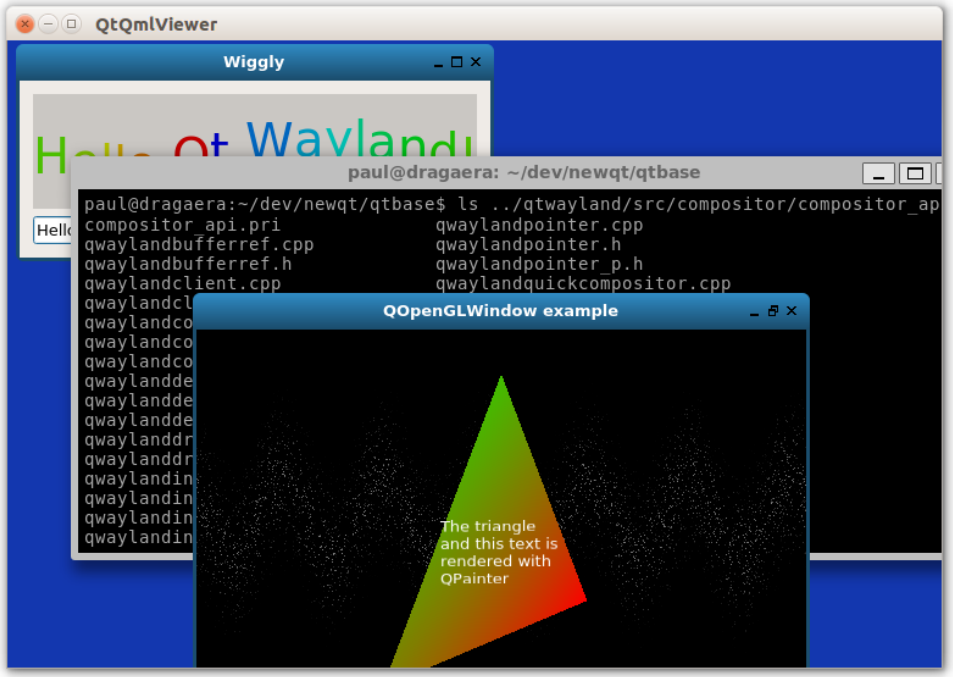


What Type of User Interfaces?



What Type of User Interfaces?

What Type of User Interfaces? (Multi-Process)



This is not it

Software Development Kits for Device Creation

- System Images
  - Software that runs on the hardware
- Toolchain (for cross-compilation)
  - Compilers
  - Tools
- Sysroot
  - Development files (headers, libraries) for the target

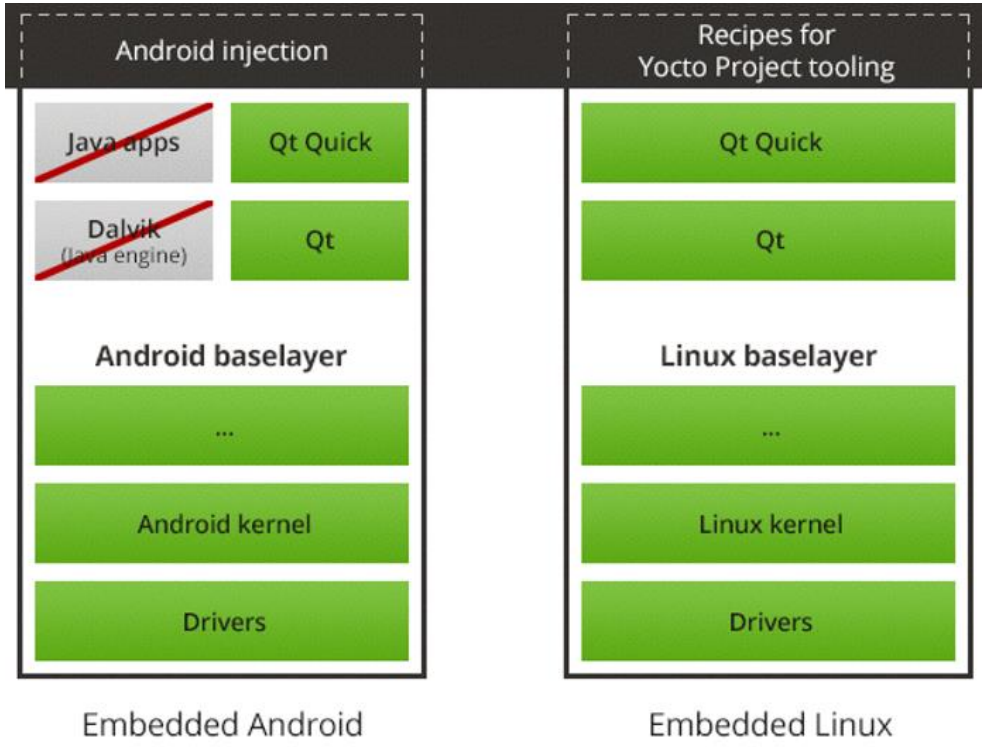
Operating System SDK
Linux

- Yocto
- Buildroot
- Ubuntu, Debian, …

- Typically customized by vendor
  - Board Support Packages (BSP)
    - Linux kernel (patches)
    - Graphics Drivers
    - Radio hardware firmware
    - Wi-Fi, Bluetooth, GSM, NFC, etc.

Operating System SDK
Other

- Android
  - Either “regular” Android with Dalvik etc.
  - Or could use just the baselayer, dropping all Dalvik and Java
- Windows Embedded
  - Pre-built images from the HW vendor
  - Microsoft provides and controls the SDK and tooling
- Windows 10 IoT
- QNX
- VxWorks
- INTEGRITY
  - Need to go through the OS vendor to get image and SDK



Problems when Getting Started

- Finding a software development kit
- Interfacing with hardware
  - How do I show something on the screen?
- Finding middleware to accelerate development
- Tooling that enables rapid iterative development
  - Deploy and debug on the device from an IDE?
  - Develop on desktop?
  - Target more than one embedded platform?
  - Target desktop and mobile as well?

Windowing System Considerations on Linux

› Have a full GPU…

…so just run a single, fullscreen GUI application via EGL + OpenGL ES?

- › Needs vendor/platform specifics to set up rendering, EGL not enough in itself
- › What about multiple screens?
- › Where does the mouse cursor come from?
- › …

› No GPU…

…so just use pure software rendering, writing directly into the framebuffer?

- › fbdev or DRM dumb buffers

› No full 3D but there is some form of 2D acceleration…

…DirectFB, OpenVG, PVR2D, Dispmanx, G2D, GAL2D, PXP, ???

Windowing System Considerations on Linux

- What if I need multiple GUI processes?
- X11
  - Not ideal for the embedded world.
- Wayland
  - Much better, lightweight, proper acceleration.
  - But now have to deal with the compositor application as well…

Windowing System Considerations: Wayland

- Where does the Wayland compositor come from?
  - Weston
    - Vendor/platform specific approaches to buffer sharing and/or efficient composing
      - Vendor-patched Weston versions…
    - Will have to customize it to get a good look and feel.
      - It's all C code…
  - Write your own (not really an option in practice)
  - KDE and Gnome developing their own compositors. Not an option for embedded, though.
  - Can we do better?

Multimedia

- How do I get accelerated video playback?
  - GStreamer with vendor-provided backends?
  - ffmpeg? libvlc?
  - OpenMAX?
  - Proprietary multimedia frameworks?
- And make it blend seamlessly with my UI?
  - Output to hardware layer with UI in another layer on top?
  - Live OpenGL textures?

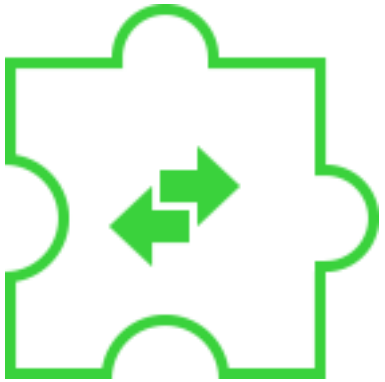
What is Qt?

Cross-Platform
Class Library

One Technology for All Platforms

Integrated
Development Tools

Shorter Time-to-Market



Cross-Platform
IDE, Qt Creator

Productive development environment

Used by over 1 million developers in 70+ industries

Proven & tested technology – since 1994

The Qt Project

- Qt is open source
- Development coordinated by the Qt Project
  - Led by The Qt Company
- Open governance
- Dual licensing: (L)GPL + Commercial

Qt is Used for…

Application Development

on Desktop,

Mobile and Embedded

Creating Powerful Devices

Device GUIs,

Ecosystems and whole SDKs

Accelerating Development with Qt Middleware

- Hardware accelerated graphics (OpenGL ES)
  - Not mandatory, Qt targets low-end (no GPU) as well.
  - Support for multiple embedded GPU vendors and their driver stacks.
- User Interface Primitives
  - Embedded has no standard widget toolkits and native look and feel!
  - Buttons, checkboxes, radio buttons, ...
  - Touch friendly
- Input methods (Virtual keyboards, touch/mouse/keyboard/pen/speech)
- Internationalization

Accelerating Development with Qt Middleware

- Multimedia: video, camera, radio
- Integrated Web Browser (based on Chromium)
- 3D
  - Integrate your own OpenGL rendering code
  - Qt 3D
  - Charts, Data Visualization
- Serial port, CAN, ModBus
- State machines, SCXML
- Connectivity: Wifi, Bluetooth
- All the traditional Qt modules for SQL, XML, networking, WebSockets, …

Accelerating Development with Qt Middleware

- Cross-platform by nature
  - Prototype and develop in the familiar desktop environment
  - Build and deploy applications to Embedded Linux devices from Windows hosts
- Not tied to windowing systems
  - Essential on Embedded Linux

Qt Developer Offering, Cross-Platform APIs

Essentials

GUI

Widgets

C++
Native LAF
Layouts
Styles
OpenGL

Qt Quick

QML
Controls
Layouts
Styles
OpenGL

WebEngine + WebView

HTML 5
Hybrid UIs

non-GUI

Core

Processes
Threads
IPC
Containers
I/O
Strings
Etc.

Multimedia

Audio
Video
Radio
Camera

Network

HTTP
FTP
TCP/UDP
SSL

Sql

SQL and Oracle databases

Qt Test

Add-ons

- Charts
- SVG
- Data Visualization
- Canvas 3D
- Virtual Keyboard
- Serial Port
- Bluetooth
- Positioning
- Concurrency
- Printing
- Scripting
- NFC
- Platform Extras
- XML
- Sensors
- Image formats
- In-App Purchasing

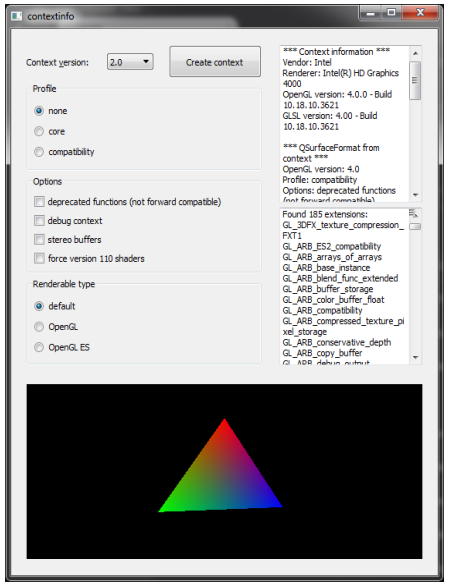
Qt UI Technologies

Qt Quick

C++ on the back, declarative UI design (QML) in the front for beautiful, modern touch-based User Experiences.

Qt Widgets

Customizable C++ UI controls for traditional desktop look-and-feel. Also good for more static embedded UIs for more limited devices / operating systems.



Web / Hybrid

Use HTML5 for dynamic web documents, Qt Quick for native interaction.

Rapid Workflow with Qt Quick

Designer

Developer

Declarative UI Design

Modern User Interfaces, written with QML.
Ideal for rapid UI prototyping.

Imperative Logic

Power of Cross-Platform Native Qt/C++

Core

Processes, Threads,
IPC, Containers,
I/O, Strings,
etc.

Network

HTTP
FTP
SSL

Sql

SQL &
Oracle
Databases

XML

Bluetooth

Positioning

NFC

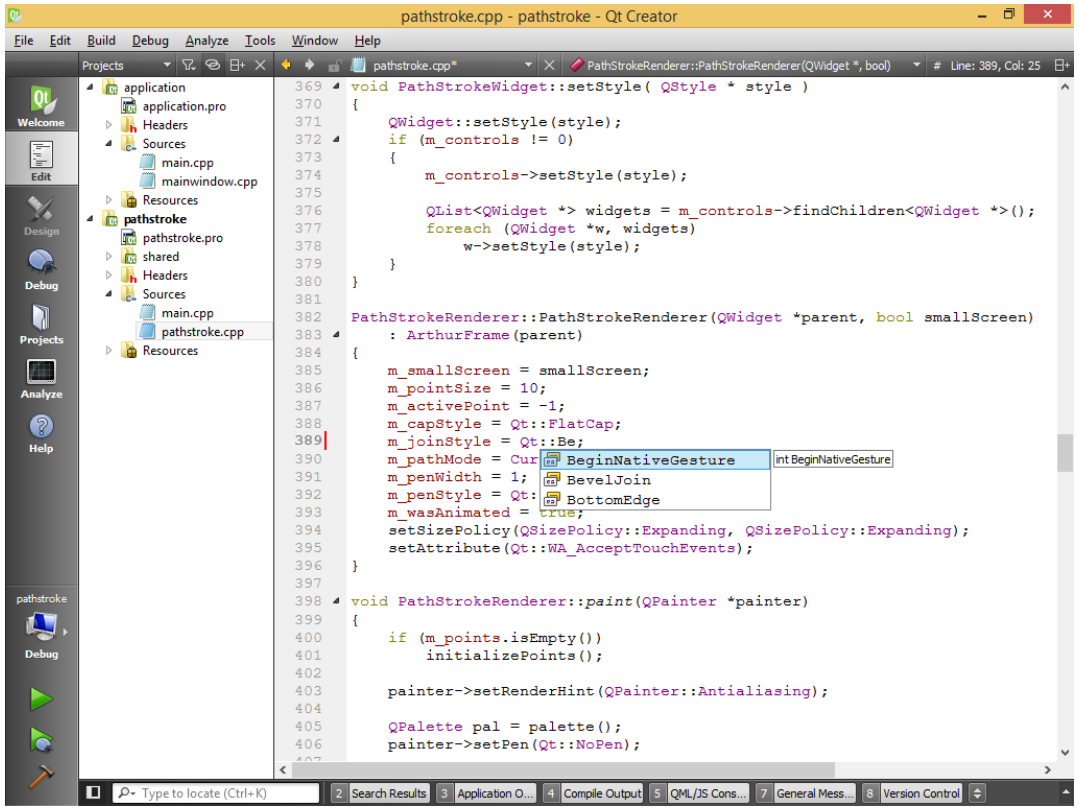
Serial Port

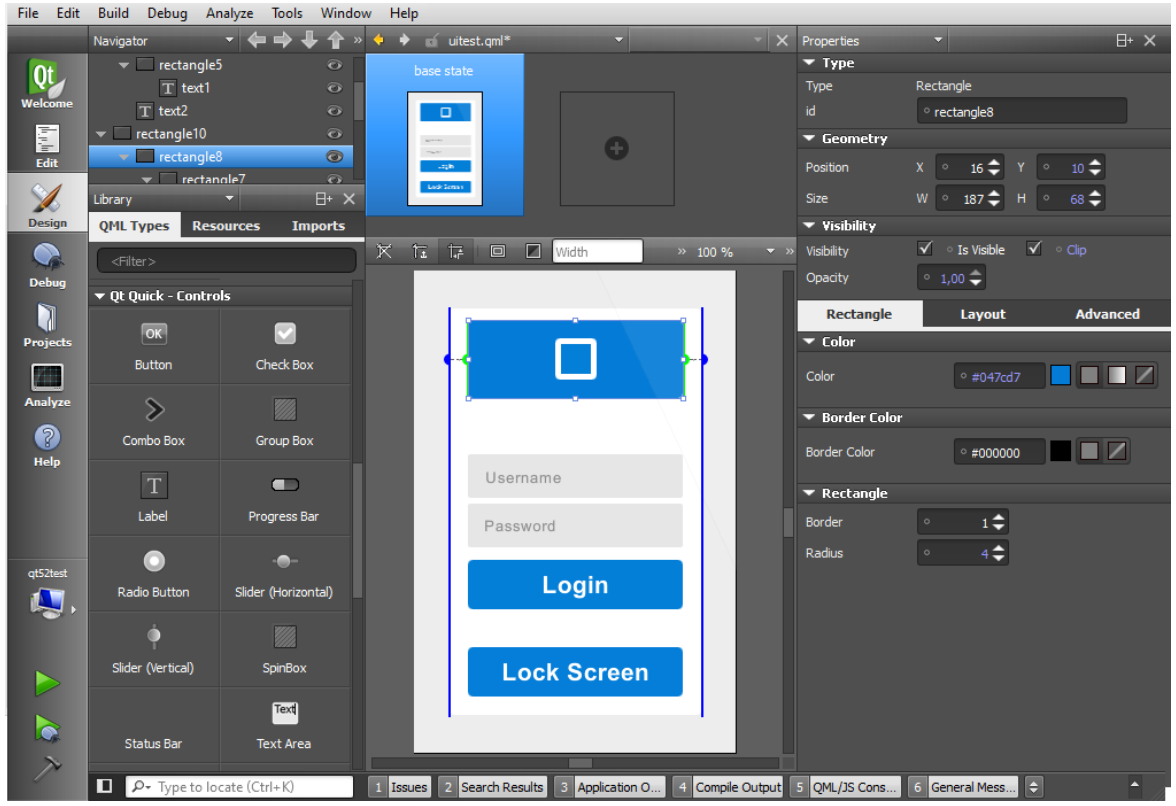
+ Direct Hardware Access

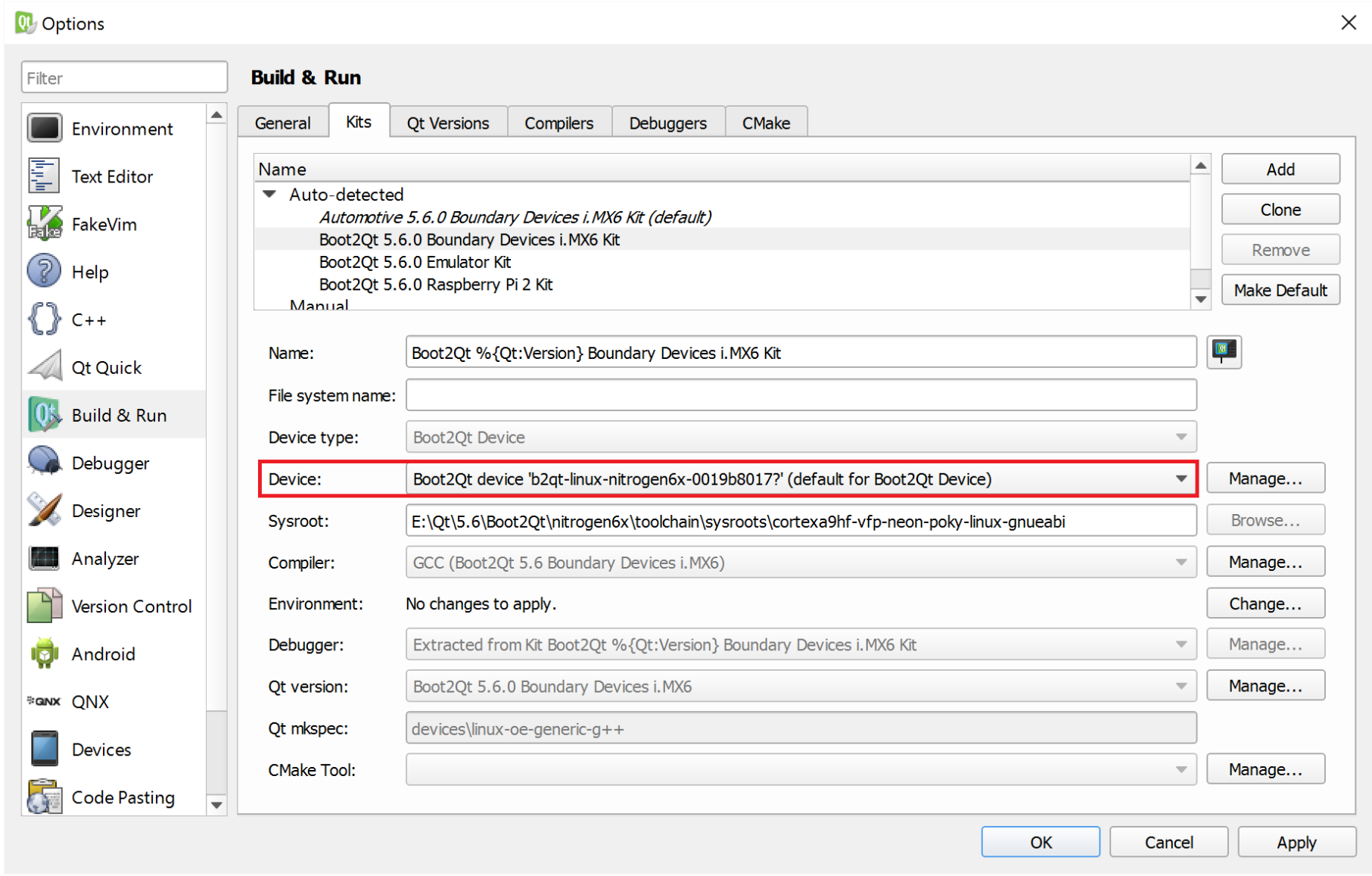
Tooling

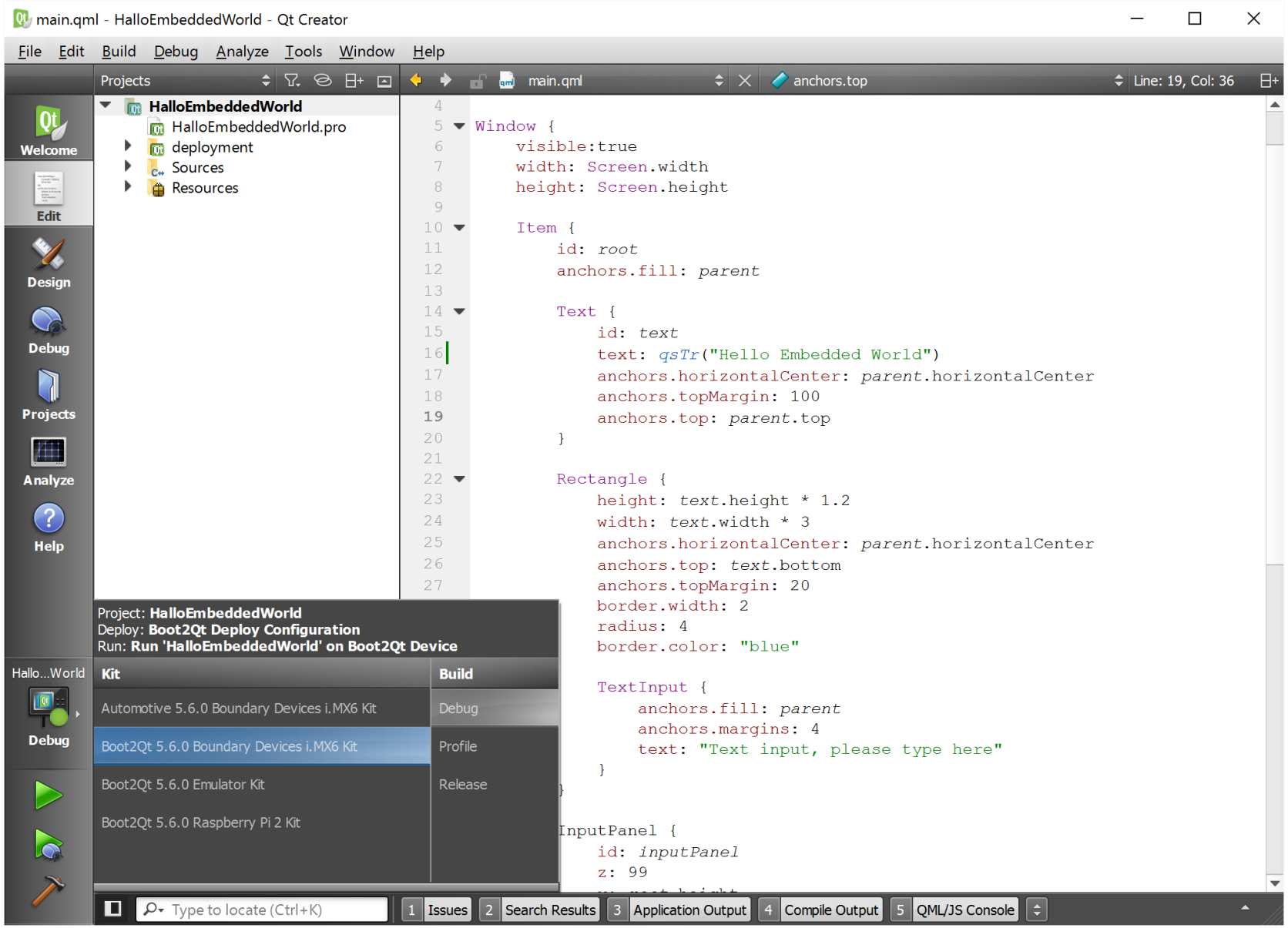
- Qt Creator integration: Build, deploy, debug and profile on target
- Integrated designer tools
- First class cross-compilation support
- Support for developing and deploying to devices on Windows
- Simulator
- Reference system images

Tooling

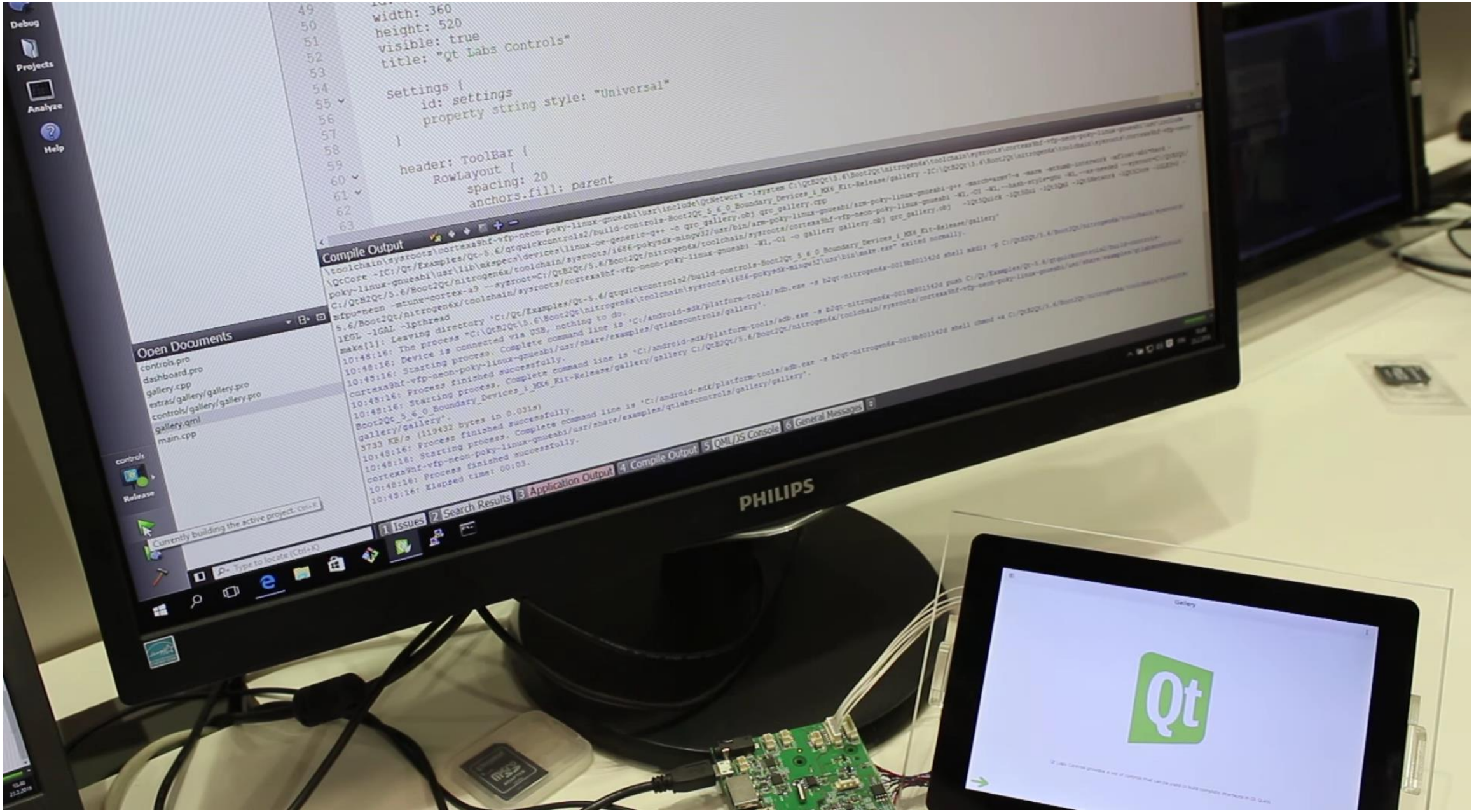

pathstroke.cpp - pathstroke - Qt Creator
File Edit Build Debug Analyze Tools Window Help
Projects
application
application.pro
Headers
Sources
main.cpp
mainwindow.cpp
Resources
pathstroke
pathstroke.pro
shared
Headers
Sources
main.cpp
pathstroke.cpp
Resources
pathstroke.cpp*
PathStrokeRenderer::PathStrokeRenderer(QWidget *, bool)
Line: 389, Col: 25
void PathStrokeWidget::setStyle(QStyle * style)
{
QWidget::setStyle(style);
if (m_controls != 0)
{
m_controls->setStyle(style);
QList<QWidget *> widgets = m_controls->findChildren<QWidget *>();
foreach (QWidget *w, widgets)
w->setStyle(style);
}
}
PathStrokeRenderer::PathStrokeRenderer(QWidget *parent, bool smallScreen)
: ArthurFrame(parent)
{
m_smallScreen = smallScreen;
m_pointSize = 10;
m_activePoint = -1;
m_capStyle = Qt::FlatCap;
m_joinStyle = Qt::Be;
m_pathMode = Cur
BeginNativeGesture
int BeginNativeGesture
BevelJoin
BottomEdge
m_penWidth = 1;
m_penStyle = Qt:
m_wasAnimated = true;
setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Expanding);
setAttribute(Qt::WA_AcceptTouchEvents);
}
void PathStrokeRenderer::paint(QPainter *painter)
{
if (m_points.isEmpty())
initializePoints();
painter->setRenderHint(QPainter::Antialiasing);
QPalette pal = palette();
painter->setPen(Qt::NoPen);
Type to locate (Ctrl+K)
2 Search Results
3 Application O...
4 Compile Output
5 QML/JS Cons...
7 General Mess...
8 Version Control

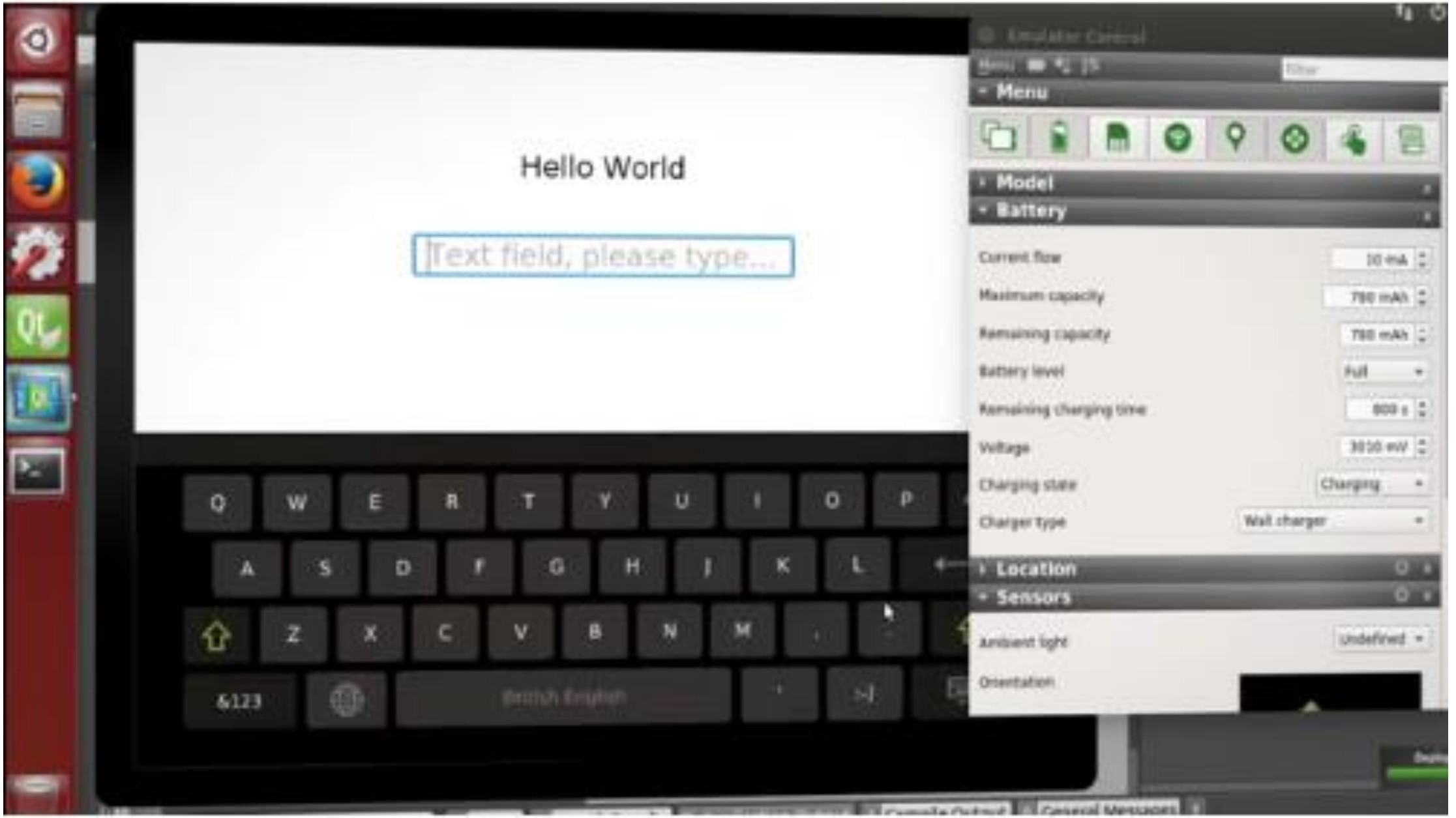

File Edit Build Debug Analyze Tools Window Help
Navigator
rectangle5
text1
text2
rectangle10
rectangle8
rectangle7
uitest.qml*
base state
Properties
Type
Type Rectangle
id rectangle8
Geometry
Position X 16 Y 10
Size W 187 H 68
Visibility
Visibility Is Visible Clip
Opacity 1,00
Rectangle Layout Advanced
Color
Color #047cd7
Border Color
Border Color #000000
Rectangle
Border 1
Radius 4
Library
QML Types Resources Imports
<Filter>
Qt Quick - Controls
Button
Check Box
Combo Box
Group Box
Label
Progress Bar
Radio Button
Slider (Horizontal)
Slider (Vertical)
SpinBox
Status Bar
Text Area
Width
100 %
Username
Password
Login
Lock Screen
qt52test
Type to locate (Ctrl+K)
1 Issues
2 Search Results
3 Application O...
4 Compile Output
5 QML/JS Cons...
6 General Mess...


Options
Filter
Environment
Text Editor
FakeVim
Help
C++
Qt Quick
Build & Run
Debugger
Designer
Analyzer
Version Control
Android
QNX
Devices
Code Pasting
Build & Run
General
Kits
Qt Versions
Compilers
Debuggers
CMake
Name
Auto-detected
Automotive 5.6.0 Boundary Devices i.MX6 Kit (default)
Boot2Qt 5.6.0 Boundary Devices i.MX6 Kit
Boot2Qt 5.6.0 Emulator Kit
Boot2Qt 5.6.0 Raspberry Pi 2 Kit
Manual
Add
Clone
Remove
Make Default
Name:
Boot2Qt %{Qt:Version} Boundary Devices i.MX6 Kit
File system name:
Device type:
Boot2Qt Device
Device:
Boot2Qt device 'b2qt-linux-nitrogen6x-0019b8017?' (default for Boot2Qt Device)
Manage...
Sysroot:
E:\Qt\5.6\Boot2Qt\nitrogen6x\toolchain\sysroots\cortexa9hf-vfp-neon-poky-linux-gnueabi
Browse...
Compiler:
GCC (Boot2Qt 5.6 Boundary Devices i.MX6)
Manage...
Environment:
No changes to apply.
Change...
Debugger:
Extracted from Kit Boot2Qt %{Qt:Version} Boundary Devices i.MX6 Kit
Manage...
Qt version:
Boot2Qt 5.6.0 Boundary Devices i.MX6
Manage...
Qt mkspec:
devices\linux-oe-generic-g++
CMake Tool:
Manage...
OK
Cancel
Apply




```
Window {
    visible:true
    width: Screen.width
    height: Screen.height

    Item {
        id: root
        anchors.fill: parent

        Text {
            id: text
            text: qsTr("Hello Embedded World")
            anchors.horizontalCenter: parent.horizontalCenter
            anchors.topMargin: 100
            anchors.top: parent.top
        }

        Rectangle {
            height: text.height * 1.2
            width: text.width * 3
            anchors.horizontalCenter: parent.horizontalCenter
            anchors.top: text.bottom
            anchors.topMargin: 20
            border.width: 2
            radius: 4
            border.color: "blue"

            TextInput {
                anchors.fill: parent
                anchors.margins: 4
                text: "Text input, please type here"
            }
        }

        InputPanel {
            id: inputPanel
            z: 99
```


Debug
Projects
Analyze
Help
Open Documents
controls.pro
dashboard.pro
gallery.cpp
extras/gallery/gallery.pro
controls/gallery/gallery.pro
gallery.qml
main.cpp
Compile Output
PHILIPS
Qt


Hello World
Text field, please type...
Menu
Model
Battery
Current flow
10 mA
Maximum capacity
780 mAh
Remaining capacity
780 mAh
Battery level
Full
Remaining charging time
800 s
Voltage
3810 mV
Charging state
Charging
Charger type
Wall charger
Location
Sensors
Ambient light
Undefined
Orientation
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
&123
British English

Getting Started with Qt on Embedded Linux

- How does Qt get installed into the image and sysroot?
  - Build and deploy manually
  - Build via Yocto, Buildroot, …
  - Distro-provided builds – don’t trust these
  - Pre-built SDKs from commercial Qt
- Qt 5.8 introduces the Qt Lite project – advanced configurability
  - Fine-grained, GUI-based configuration of which parts of Qt to include in the build

Getting Started with Qt on Embedded Linux

- Open-source vs. commercial
  - Licensing
    - From Qt 5.7 most of Qt is LGPL**v3** + Commercial.
    - Some modules GPLv3 + Commercial.
  - Commercial-only components
    - Opening up more and more.

Is My Board Supported?

- If it’s a normal ARM device with a usual toolchain and sysroot, the main question is the GPU.
- Beware that multimedia (video, camera) is often a whole separate story.
- New devices are added
  - upstream via *device specs* (just compiler flags and some graphics-related config)
  - Yocto and friends do not fully use the upstream device configuration system!
    - Up to them to manage.
- In some cases new graphics system backends are necessary too.

Is My Board's GPU supported?

› Fullscreen EGL/GLES via

1. vendor-specific fbdev glues,
2. DRM/KMS with GBM,
3. DRM/KMS with EGLDevice/EGLOutput/EGLStream,
4. or proprietary system compositor APIs

1. Vivante fbdev - Anything with NXP i.MX6
   Mali fbdev – ODROID and others
   PowerVR fbdev - Beaglebone

2. Anything with Mesa – e.g. Intel NUC
   Modern PowerVR systems
   Possibly others

3. NVIDIA Jetson Pro, DRIVE CX, DRIVE PX

4. Raspberry Pi 1/2/3

Is My Board's GPU supported?

- Wayland
  - Surprise, surprise: platform-specific bits!
    - EGLImage (Mesa and most embedded vendors) vs. EGLStream (NVIDIA)
    - Experimental special backends (Raspberry Pi)
  - Most tested platforms:
    - NVIDIA Tegra K1/X1 in automotive (e.g. DRIVE CX)
    - Intel
    - Vivante GC2000 and similar (e.g. i.MX6)
  - Many others may work too.
    - Make sure the vendor claims Wayland support *and* that the support is available in your BSP and software stack.
- X11
  - GLX and EGL

Is My Board’s GPU supported?

- What about systems without OpenGL?
- In some cases DirectFB may be supported.
- NXP i.MX7, Colibri VFxx, and similar have no 3D acceleration.
  - Can still use Qt Quick (QML) via its software renderer.
  - Previously commercial-only.
  - Opened up in Qt 5.7 as GPLv3+Commercial
  - Fully integrated as LGPLv3+Commercial in Qt 5.8. Enhanced performance.
- Stay tuned for news regarding OpenVG and possibly other APIs in Qt 5.9+

Building and Deploying a Qt application

- Manually
  - Once Qt is built, do qmake && make for apps.
  - Cross-compilation is conveniently managed by the build system.
  - Deploy the binaries (scp, rsync, etc.)
- Qt Creator
  - Once SDKs are set up, build and deploy with one click.
  - Deployment happens via sftp or adb.
  - Pre-built reference images + SDKs from commercial is likely the easiest to get started.
    - However any SDK and device can be set up with open-source as well, as long as sftp, scp or similar is functional.

Thank You!

› https://www.qt.io

› https://www.qt.io/qt-in-use/

› https://www.qt.io/device-creation/

› https://doc-snapshots.qt.io/qt5-5.8/embedded-linux.html

› https://blog.qt.io