

LORENZO MANCINI

Qt Specialist

# Introduzione a Qt for MCU

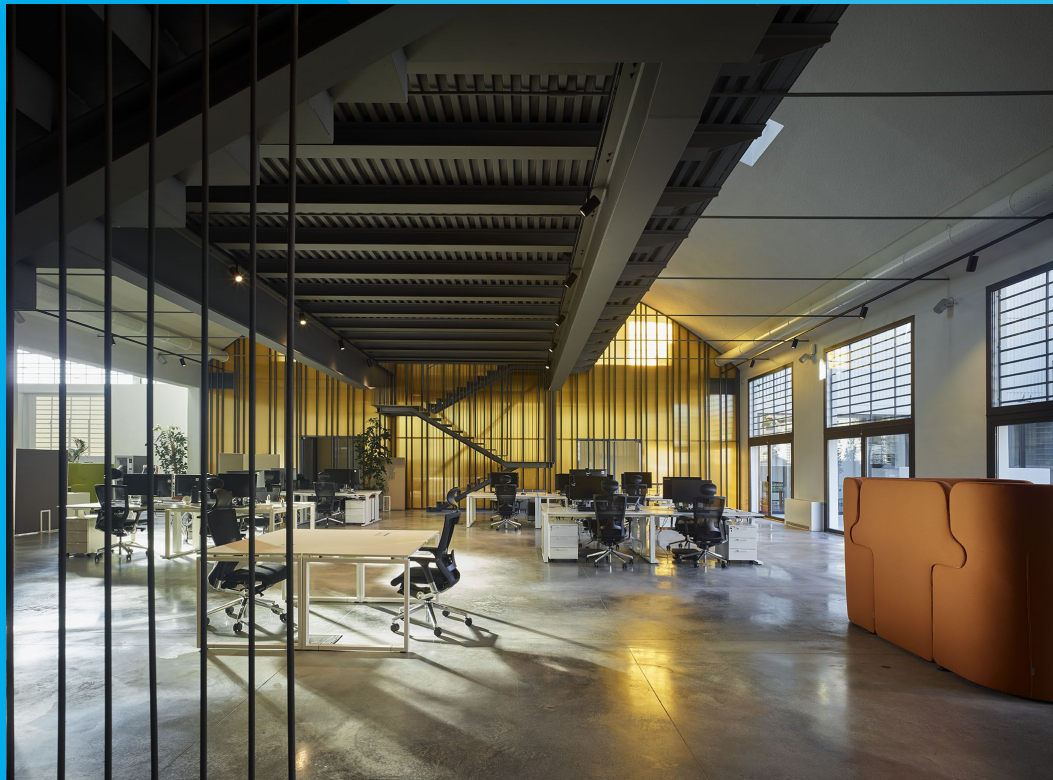
Techlabs 2020

08/07/2020

develer

# I CHI SIAMO

Develer un'Azienda che  
che progetta e realizza  
soluzioni hardware e  
software in ambiti  
industriali innovativi.



## I COSA FACCIAMO

- Sviluppo software
- Sistemi embedded
- Corsi
- Eventi

Hai bisogno dei nostri servizi?

[www.develer.com/servizi](http://www.develer.com/servizi)

Vuoi lavorare con noi?

[www.develer.com/lavora-con-noi](http://www.develer.com/lavora-con-noi)

# Lorenzo Mancini – Mi presento

- Qt Specialist in Develer
  - uso Qt dal 2004 (3.3)
  - 10+ anni di sviluppo su Qt / Qt Quick
  - 100+ programmatori formati su Qt
- Appassionato di programmazione low level
  - QtDay 2015: [GPU, la risorsa segreta dei dispositivi embedded](#)
  - QtDay 2017: Interfacce grafiche a 60fps, senza GPU
  - Porting Python su Nintendo DS: <https://github.com/develersrl/dspython>

# Piano del webinar

Piattaforma di sviluppo MCU

Qt for MCU

Concezione e compromessi

Caratteristiche

Demo

Qt for MCU in pratica

Integrazione C++

D&R

# Piano del webinar

## Piattaforma di sviluppo MCU

Qt for MCU

Concezione e compromessi

Caratteristiche

Demo

Qt for MCU in pratica

Integrazione C++

D&R

## Ruolo dei microcontrollori nel 2020

Non sono più i microcontrollori degli anni '90

- Baseline: ~60Mhz, 2MB RAM, display, acceleratore 2D

Opportunità offerte oggi dai MCU:

- Basso consumo (STM32F7: < 100mA max carico)
- Boot istantaneo (ok, non proprio ma ~1")
- Piccole dimensioni
- Basso costo (Cortex M7 \$0.99, STM32F7 \$5.99)

Tuttavia...

## Ruolo dei microcontrollori nel 2020

No sistema operativo, hardware eterogeneo

- Chrom-Art (STM), Pxp (NXP), 2D Draw Engine (Renesas)

Framework grafici platform-specific

- Widget statici, old-style, no layout
- Alcune basate su wizard

Userbase ridotta

- Difficile assumere personale
- Difficile trovare aiuto online



# Piano del webinar

Piattaforma di sviluppo MCU

**Qt for MCU**

**Concezione e compromessi**

**Caratteristiche**

Demo

Qt for MCU in pratica

Integrazione C++

D&R

## Idea: Qt su MCU, primi passi

Qt su MCU è un'ottima idea: cross-platform, API Qt-style, librerie ottimizzate, documentazione, etc...

Detto: “Qt gira ovunque ci sia un compilatore C++”

- “...e un sistema operativo, e 64MB di RAM”

L'idea non è nuova:

- Qt 4: qconfig permetteva di compilare via pezzi di Qt per risparmiare memoria
- 2014: su qt-interest appaiono le prime patch per far girare QtCore senza Linux

## Idea: Qt su MCU, primi passi

Le difficoltà riscontrate:

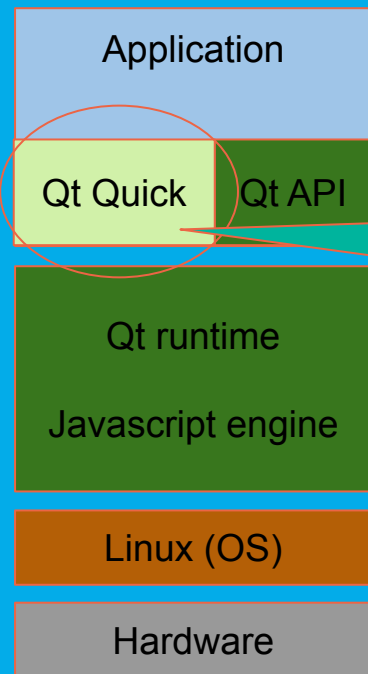
- QtCore è molto legato al kernel
- Il footprint in memoria di QtCore e QtGui sarebbe troppo grande per un MCU

Serve cambiare completamente approccio

2016: viene avviato il progetto Qt Quick Ultralite

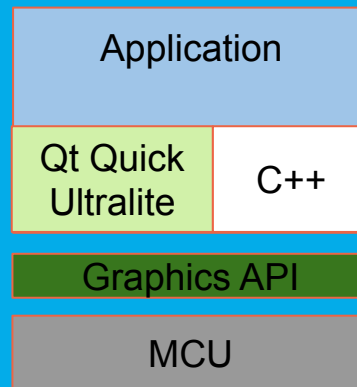
# Qt Quick Ultralite, l'idea

Qt for Device Creation



Idea portante:  
Qt Quick è la API

Qt for MCU



## Qt for MCU: dove siamo oggi

Prodotto basato su Qt Quick Ultralite

Rilasciato Qt for MCU 1.3

- 3 ore fa

Licenza commerciale fornita da The Qt Company

# Cos'è Qt for MCU

Ricco subset di QML per realizzare UI  
Motore di rendering accelerato hardware  
Runtime ottimizzato e minimale, MCU-aware

## Cos'è Qt for MCU (2)

Subset di Qt Quick Controls per widget già pronti

Subset di Charts

Internazionalizzazione

Integrato con Qt Creator

Documentazione Qt-Style

- abstract di ogni componente, esempi, tutorial

## Cosa non è Qt for MCU

Non è un port completo di Qt su MCU  
Non vuole diventarlo (questo è un bene!)

In particolare: no QtCore, no QtGui, no funzioni di rete, no filesystem, no multimedia.

Qt for MCU copre la sola parte di interfaccia grafica

- ...per tutto il resto si possono ovviamente integrare librerie di terze parti



## Piattaforme deploy supportate

No Sistema Operativo => Nessuna astrazione hardware

NXP (i.MXRT10xx)

Renesas (RH850)

ST (STM32 serie F, H, L)

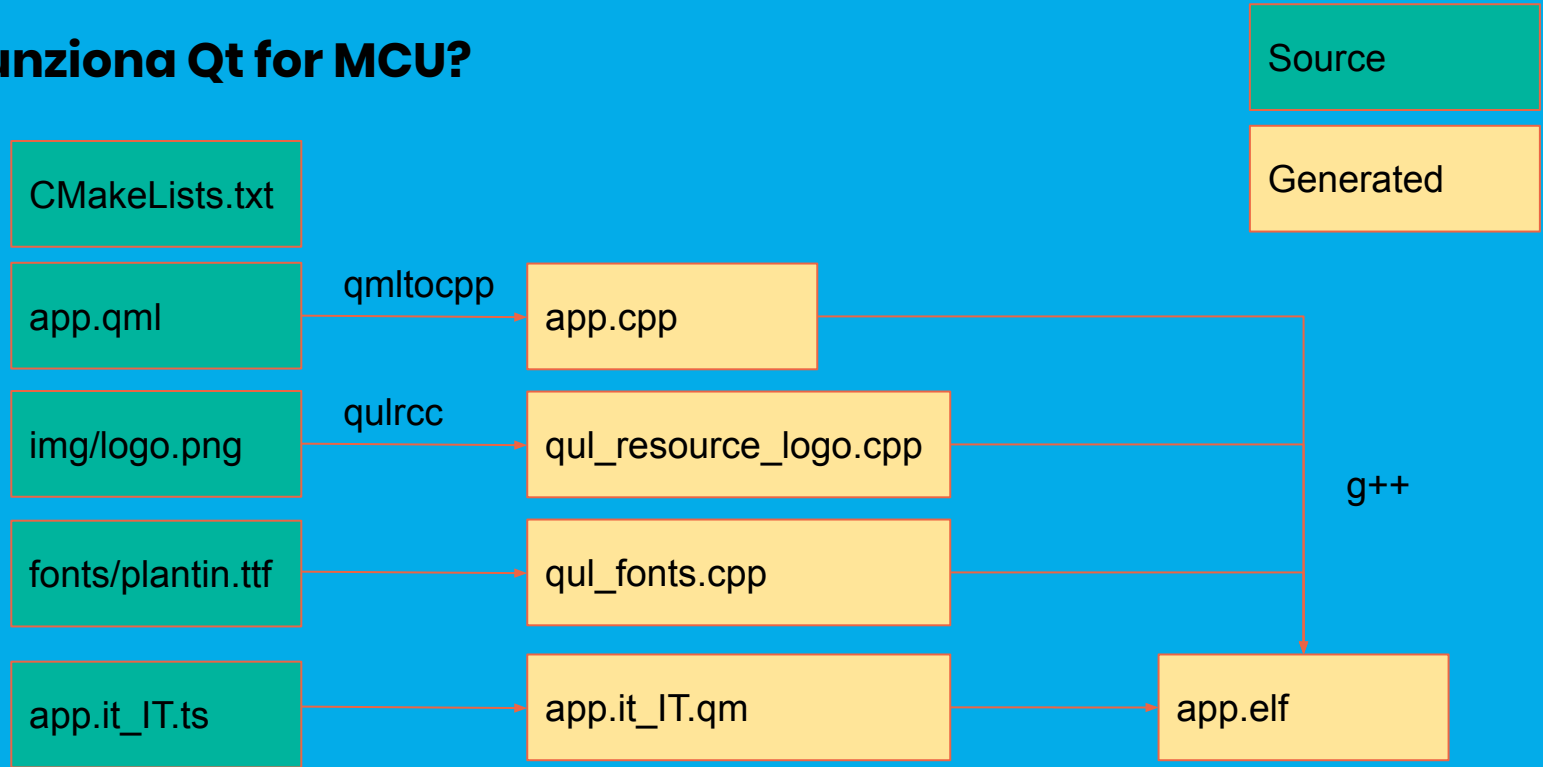
Supporto per FreeRTOS

## Piattaforme di sviluppo supportate

Per adesso solo Windows 10

Si possono eseguire le applicazioni Qt Quick Ultralite anche su desktop (MSVC2017)

# Come funziona Qt for MCU?



# Come funziona Qt for MCU?

## Build con CMake

- Non è solo lo strumento di build
- Configurazione di basso livello dell'applicazione
  - QUL\_COLOR\_DEPTH
  - QUL\_DEFAULT\_FONT\_FAMILY, QUL\_FONTS\_DIR
  - QUL\_TEXTURE\_CACHE
- Aggiunta proprietà alle risorse
  - QUL\_COPY\_TO\_RAM
- Tutte queste cose tipicamente stanno nel kernel o nello userspace

## Come funziona Qt for MCU?

### Conversione QML in C++

- Subset di QML
- No runtime Qt Quick
- No ECMAScript
- No garbage collection

## Come funziona Qt for MCU?

Immagini incluse nell'eseguibile

- Note a compile time
- Decomprese prima dell'inclusione (default)
- No caricamento dinamico
  - In Qt for MCU non ho un filesystem
  - ...allora come faccio a fare l'applicazione album fotografico?

## Come funziona Qt for MCU?

Font inclusi nell'eseguibile

- Attenzione, vengono inclusi *\*tutti\** quelli nella font dir, non solo quelli usati

Internazionalizzazione

- Flusso standard Linguist

## Qt for MCU chicche

StaticText

Testo fisso e noto a compile time

ColorizedImage

Effetto colorize eseguito a runtime



## Qt for MCU limitazioni

### Qt Creator

- No QML debugger
- No profiler

### Stati

- Solo per l'elemento root del QML

### Trasformazioni

- Solo per le Image

## Qt for MCU limitazioni

Supportato subset di ECMAScript

- Per compilarlo in C++, i tipi devono essere noti o deducibili a compile time

```
function getColor(i : int, pressed : bool) : color {  
    var isEven = i % 2 == 0  
    if (pressed) {  
        return "#AAFFAA"  
    }  
    else if (isEven) {  
        return "#AACCAA"  
    }  
    return "#CCAACC"  
}
```

## Qt for MCU da sapere

Le versioni dei package per l'import sono ignorate

Nessun background di default

No clear all'inizio del fotogramma

## Qt for MCU in pratica (STM)

Hardware: STM32F769

Schermo: 480x272

Depth: 32 bit

RAM: 16 MB

CPU: Cortex M7 (216Mhz)

## Piano del webinar

Piattaforma di sviluppo MCU

Qt for MCU

Concezione e compromessi

Caratteristiche

### Demo

Qt for MCU in pratica

Integrazione C++

D&R

# Live demo

Configurazione

Creare nuova applicazione

Aggiungere font

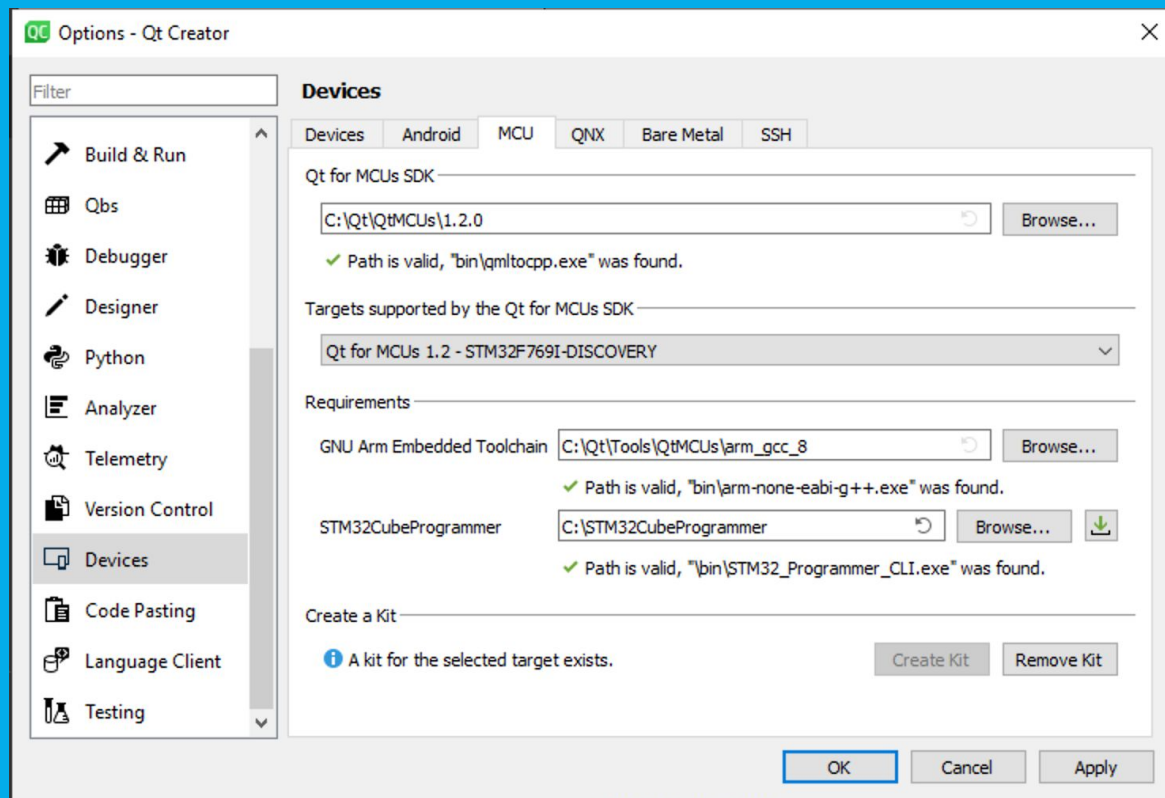
Usare immagini

Usare Controls

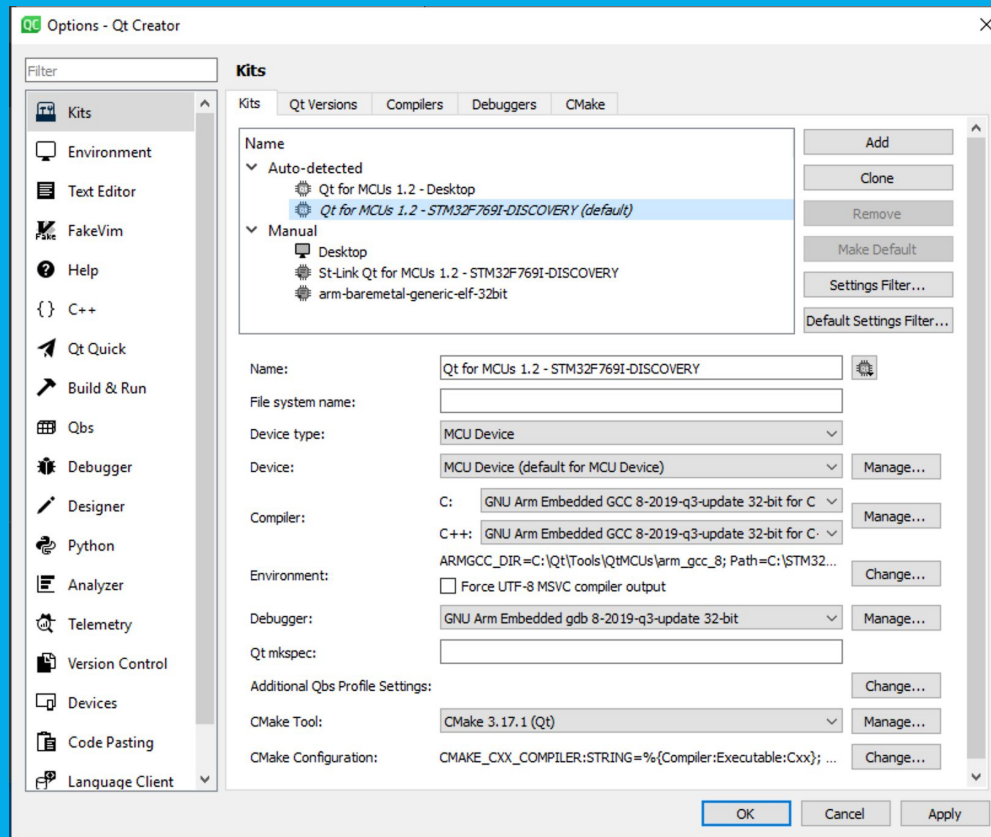
Integrazione C++

Localizzazione

# Configurazione



# Configurazione





# Nuova applicazione

Wizard “MCU Support Application”

Diamo un’occhiata a cosa c’è dentro

- CMakeLists.txt
- Un QML
- Alcuni file C++ a corredo

## Aggiungere font

Creare una directory fonts/ con i ttf dentro

CMakeLists.txt

```
find_package(Qu1)
```

```
set(QUL_FONTS_DIR "${CMAKE_CURRENT_SOURCE_DIR}/fonts")
```

```
set(QUL_DEFAULT_FONT_FAMILY "Source Code Pro")
```

```
add_executable(webinar ${OS}/main.cpp)
```

## Usare immagini

Creare directory images/ con png dentro

app.qml

```
Image {
```

```
    source: "images/qt-logo.png"
```

```
}
```

CMakeLists.txt

```
add_executable(webinar ${OS}/main.cpp)
```

```
qml_target_qml_sources(webinar webinar.qml)
```

```
qml_add_resource(webinar FILES images/qt-logo.png)
```

## Usare Controls

CMakeLists.txt

```
target_link_libraries(webinar Qml::QuickUltralite  
Qml::QuickUltraliteControlsStyleDefault)
```

app.qml

```
import QtQuick.Controls 2.0
```

# Integrazione C++

Aggiungere i file di libreria nella directory di  
piattaforma (“BareMetal”)

# Integrazione C++

led\_wrapper.h

```
#include <qul/singleton.h>
```

```
#include "led.h"
```

```
struct LED : public Qul::Singleton<LED>
```

```
{
```

```
    void setEnabled(bool enabled)
```

```
{
```

```
    if (enabled)
```

```
        green_led_switch_on();
```

```
    else
```

```
        green_led_switch_off();
```

```
}
```

```
};
```

# Integrazione C++

## CMakeLists.txt

```
target_include_directories(webinar PUBLIC ${CMAKE_CURRENT_SOURCE_DIR})  
target_include_directories(webinar PUBLIC ${CMAKE_CURRENT_BINARY_DIR})  
qul_target_generate_interfaces(webinar ${OS}/led_wrapper.h)  
target_sources(webinar  
    PRIVATE  
    ${OS}/led.h  
    ${OS}/led_stm32f769.cpp  
    ${OS}/led_wrapper.h  
)
```

# Localizzazione

app.qml

```
text: qsTr("Hello World!")
```

CMakeLists.txt

```
qul_target_embed_translations(webinar webinar.it_IT.ts)
```

Clonare build configuration -> "Update translations"  
Scegliere target "update\_translations"



# Localizzazione

Inserire traduzioni e rilanciare il programma

```
Qt.uiLanguage = checked ? "it_IT" : "";
```

## Dimensioni finali

Codice, no asset, no qt: 350Kb

Codice, con asset, no qt: 700Kb

Codice, con asset, con qt: 1.8Mb (.elf finale)

## Piano del webinar

Piattaforma di sviluppo MCU

Qt for MCU

Concezione e compromessi

Caratteristiche

Demo

**Qt for MCU in pratica**

Integrazione C++

D&R

# Qt for MCU in pratica

Ambiente di sviluppo “reale”:

- Itero su desktop, ogni tanto deploy
- Ho una CI che fa build per me
- Posso fare debug su target

## Qt for MCU in pratica

*Itero su desktop, ogni tanto deploy*

Strada 1: Uso Qt Quick Ultralite anche su desktop

- + Se uso API incompatibili me ne accorgo subito
- + Tutta la base di codice è unica (non solo QML)

Strada 2: Uso Qt Quick desktop

- + Posso usare QML debugger e profiler
- + Posso riusare l'interfaccia anche su altre piattaforme

## Qt for MCU in pratica

*Ho una CI che fa build per me*

Ok, tutto controllabile da riga di comando

## Qt for MCU in pratica (STM)

*Posso fare debug su target*

```
ST-LINK_gdbserver.exe -d -v -cp  
"<STM32_CUBE_PROG_INSTALL_PATH>\bin"
```

```
arm-none-eabi-gdb.exe webinar.elf
```

```
> target remote 127.0.0.1:61234
```

## Qt for MCU in pratica (STM)

Come leggere i messaggi in console?

Virtual COM Port

Baud rate: 115200

Data bits: 8

Stop bits: 1

Parity: None

Flow control: None



# Qt for MCU in pratica (STM)

Integrazione GDB con Qt Creator

Configurare plugin BareMetal

Installare Python 2.7

Usare gdb-py al posto di gdb

## Piano del webinar

Piattaforma di sviluppo MCU

Qt for MCU

Concezione e compromessi

Caratteristiche

Demo

Qt for MCU in pratica

**Integrazione C++**

D&R

# Integrazione C++

Da C++ è possibile:

- Definire componenti QML
  - Proprietà
  - Segnali per comunicazione C++ / QML
- Esporre funzioni

```
#include <qml/QtObject.h>
#include <qml/property.h>
#include <qml/signal.h>

struct Score : public Qml::Object
{
    Qml::Property<int> score;
    int add(int bonus);
    Qml::Signal<void>() changed;
};
```

QML:

```
Item {
    Score {
        id: playerScore
        score: 0
        onChanged: {if score == 1000 ...}
    }

    ...
    playerScore.add(100)
}
```

# Registrare interfacce con qml\_target\_generate\_interfaces

## Struct deve ereditare da Qml::Object

## Default constructable

```
#include <qml/QtObject.h>
#include <qml/property.h>
#include <qml/signal.h>
```

```
struct Score : public Qml::Object
{
    Qml::Property<int> score;
    int add(int bonus);
    Qml::Signal<void()> changed;
};
```

QML:

```
Item {
    Score {
        id: playerScore
        score: 0
        onChangeed: {if score == 1000 ...}
    }

    ...
    playerScore.add(100)
}
```

# Comm C++/QML tramite segnali

Lato codice C++:  
add(100) → emissione changed

Lato codice QML:  
onChangeed → gestione valore

## Link Utili

<https://www.qt.io/download>

- per scaricare licenza di valutazione (Try Qt)

<https://www.qt.io/product/develop-software-micro-controllers-mcu>

- Prodotto Qt for MCU

<https://doc.qt.io/QtForMCUs/index.html>

- Documentazione

<https://forum.qt.io/category/62/qt-for-mcus>

- Qt for MCUs forum

# Conclusioni

Valore di Qt for MCU:

- Riutilizzo competenza Qt Quick
- Performance come da standard Qt
- Libreria Controls / Chart vasta

Attenzione a:

- Impostare ambiente di sviluppo
- Piattaforme supportate

# Lorenzo Mancini

lmancini@develer.com

Vuoi rimanere aggiornato sugli eventi Develer?  
Seguici nei nostri canali social:



**develer**

[www.develer.com](http://www.develer.com)