Embedded Multimedia
Building Qt 5.10.1 with Python Skript

Gruppe 6
Daniel Scholtyssek
Mat-Nr: 133750
daniel.scholtyssek@tu-dortmund.de

TU Dortmund, Fakultät für Informatik

Abstract. Dieses Paper soll einen kurzen Überblick über die Verwendung eines Python Skript geben, mit dem es Möglich ist, seinen Raspberry Pi mit Qt5.10.1 auszustatten. Als Hilfsmittel würden [1] und [2] verwendet. Sinn und Zweck des Skripts ist es den Installationsprozess (vor allem auf privaten Raspberry Pi's) zu vereinfachen und damit den Teilnehmern der Vorlesung *Embedded Multimedia* Arbeit abzunehmen.

1 Voraussetzungen

Aufgrund der Komplexität der automatischen Installation von Software, trifft das Skript einige Annahmen, um den Prozess zu vereinfachen.

1. Es wird die aktuellste Version des *NOOBS* Raspbian Betriebssystems verwendet.
2. Dabei werden keine Anpassungen am Benutzer ("pi") oder am Root Passwort gemacht.
3. Außerdem ist genügend Festplattenspeicher verfügbar, um die benötigte Software zu installieren.

2 Download

Neben dem Download aus dem *Embedded Multimedia* Moodle Raum, ist auch ein direkter Download von meiner GitHub Seite[1] möglich. Dazu muss nur ein Terminal geöffnet und folgende Befehle eingegeben werden:

Listing 1.1. Befehl für das Klonen des Repository

```
$ cd /home/pi/
$ git clone https://github.com/RuUnation/EMM2018.git
$ cd EMM2018/
$ chmod +x installQt.py
```

Mit dieser Reihe von Befehlen würde das Skript heruntergeladen und ausführbar gemacht.

[1] https://github.com/RuUnation/EMM2018

3 Ausführung und mögliche Optionen

Es handelt sich um ein *Python3* Skript, welches nun ausgeführt werden kann. Wichtig ist, dass das Skript mit dem *Sudo* Befehl gestartet wird.

Listing 1.2. Hilfstext des Skripts

```
$ sudo ./installQt.py -h
-------------------------------------------------
usage: installQt.py [-h] [-d DOWNLOADPATH]
       [-b BUILDPATH] [-j {1,2,3,4}] [-p PLATFORM]
       [-a] [--bluetooth] [--audio] [--database]
       [--print] [--wayland] [--accessibility]
       [--distupgrade] [--rpiupdate]

optional arguments:
  -h, --help            show this help message and
                        exit

  -d DOWNLOADPATH, --downloadpath DOWNLOADPATH
                        downloadpath for the
                        qt5.10.1 source. Default is
                           home/pi/Downloads.

  -b BUILDPATH, --buildpath BUILDPATH
                        shadow build directory.
                        Default is home/pi/build

  -j {1,2,3,4}, --jobs {1,2,3,4}
                        number of jobs for make
                        (j4 in the doc)

  -p PLATFORM, --platform PLATFORM
                        which platform, default
                        is linux-rasp-pi2-g++.
                        Other Platforms are:
                        linux-rasp-pi-g++,
                        linux-rasp-pi3-g++
                        and linux-rasp-pi3-vc4-g++

  -a, --all             install all optional
                        development packages


  --bluetooth           install optional development
                        packages for bluetooth
```

```
    --audio                 install optional development
                            packages for audio

    --database              install optional development
                            packages for databases

    --print                 install optional development
                            packages for printing

    --wayland               install optional development
                            packages for wayland
                            support

    --accessibility         install optional development
                            packages for accessibility

    --distupgrade           edits sources.list and
                            performes dist-upgrade

    --rpiupdate             updates the pi's firmware
```

Mit dem Argument *-h* wird der Hilfstext ausgegeben. Hier wird gezeigt, wie das Skript zu benutzen ist und welche Argumente angegeben werden können. Alle Argumente sind optional und müssen nicht angegeben werden. Die Standard-einstellung für *–platform* ist linux-rasp-pi2-g++, damit Raspberry Pi EGLFS korrekt erkannt wird. Um ein reibungslosen Ablauf zu gewährleistet, muss das Skript dreimal aufgerufen werden. Dies ist nötig, da einige Dinge installiert und durchgeführt werden, die einen Neustart benötigen. Folgende Aufrufe müssen in dieser Reihenfolge getätigt werden:

Listing 1.3. Befehlsreihenfolge

```
$ sudo ./installQt.py --distupgrade
--------------  reboot -------------
$ sudo ./installQt.py --rpiupgrade
--------------- reboot -------------
$ sudo ./installQt.py --all --jobs=2
-----------------reboot -------------
```

Mit den Optionen im dritten Durchlauf werden alle optionalen Development Packages installiert (siehe [2]) und die Anzahl von Jobs beim *Make* Vorgang auf zwei gesetzt. Der Versuch *–jobs* auf 3 oder sogar 4 zu stellen (-j3 bzw. -j4), haben einen Raspberry pi 3 ohne aktive Kühlung, immer in den Bereich des *thermal throttling* gebracht (ca. 80°+). Mit *–jobs=2* ist er immer relativ stabil bei 72° geblieben.

ACHTUNG: Der ganze Vorgang kann einige Stunden dauern.

Das Herunterladen der *qt-everywhere-src* kann verhindert werden. Das Skript prüft, ob das Archive evtl. schon vorhanden ist, wodurch es möglich ist das Herunterladen der Software Übersprungen werden kann, beispielsweise wenn man das Archive schon auf einem USB Stick hat. Dann muss nur Mittels *-d* oder *–downloadpath* dieser Ordner angegeben werden, oder das Archive in den Standardpfad */home/pi/Downloads* kopiert werden.
Falls der Build-Vorgang aus versehen abgebrochen wird, kann das Skript einfach wieder gestartet werden. Die bisher gebauten Sachen gehen nicht verloren.

Nach erfolgreichem Durchlauf des Skriptes muss nur noch der QtCreator konfiguriert werden. Dazu sind folgende Schritte nötig:

1. In Qt Creator im Menü Extras/Enstellungen.. links Reiter **Erstellung und Ausführung** wählen, darin zweiten Tab namens **Compiler**.
2. **Hinzufügen - GCC** klicken und für **C** den Compiler Pfad **/usr/bin/gcc** angeben.
3. **Hinzufügen - GCC** klicken und für **C++** den Compiler Pfad **/usr/bin/c++** angeben.
4. Nun zum Reiter **Qt Versionen** wechseln. Falls **Qt5.10.1** unter dem Pfad **/opt/Qt5.10.1/bin/qmake** noch nicht vorhanden ist, diesen hinzufügen.
5. Nun zum Reiter **Kits** wechseln und das bereits vorhandene Desktop Kit anpassen:
   - Sysroot: /
   - Compiler C: Den zuvor erstellten GCC Compiler auswählen
   - Compiler C++: Den zuvor erstellten GCC Compiler auswählen
   - Debugger: GDB von System in /usr/bin/gdb
   - CMake-Werkzeug: System-CMake in /usr/bin/cmake
6. Danach auf **Anwenden** klicken

Anschließend können Projekte mit **Erstellen - Projekt erstellen (Str + B)** gebaut werden. Nun muss nur ein Terminal geöffnet werde, zum gerade erstellten Buildverzeichnis navigieren (Im Ordner, wo ihr auch euer Projekt gespeichert habt. Standardname ist *build-Projektname-Kitname-[Debug—Release]*) und dort eure Anwendung mit dem Befehl **./Projektname -platform xcb** starten.

4 Offene Fragen/mögliche Fehler und Änderungen

- Welche Version von Raspbian soll genommen werden? Veilleicht geht es bei mir nicht wegen der falschen Betriebssystemversion. Aktuell verwende ich *Raspbian Stretch with desktop*
- Fehler in der angepassten Doku im Moodle. Im *Configure the build* Teil sind einige Fehler drin, welche im Original (siehe [1]) nicht drin sind. Es wird z.B. noch von *"qt-everywhere-opensource-src-5.10.0"* gesprochen, obwohl es in der neuen Version *"qt-everywhere-src-5.10.1"* heißt.

- *"PKG_CONFIG_SYSROOT_DIR=..qt-everywhere-opensource-src-5.10.0"* ist nicht korrekt und wird auch in der Original Doku anders gesetzt. Was soll man jetzt hier machen.
- In der Qt eignen Doku zur Installation
- Das Package *libatspi-dev* ist in der aktuellen Debian Raspbian Version nicht mehr in den Respositories zu finden. Optionales Feature Accessibility.
- Warum wird in der Moodle Dokumentation die Optionen *-system-freetype*, *-fontconfig* und *-glib* nicht gesetzt?
- In der Anleitung wird für *linux-rasp-pi2-g++* gebaut, obwohl wir Pi's der dritten Version bekommen haben. Gibt es da einen Grund?
- In der Anleitung steht *"The module qtscript is deprecated, but required for Qt Creator. Therefore it has to be built as well: sudo make"*. Ohne zu sagen, wo genau, was genau, gebaut werden soll. Hier muss möglicherweise noch das Skript angepasst werden. Aktuell wird einfach in den Unterordner *qtscript* navigiert und *make* aufgerufen. Im Skript wird außerdem noch die Erweiterung gemacht, dass QtCreator aus dem Repository installiert wird.
- Selbst Leute in den Kommentaren der Original Anleitung sagen, dass sie immer *EGLFS Rasberry Pi ..... no* bekommen. Also muss zwingend die Version für den Raspberry Pi 2 gebaut werden, da so die EGLFS korrekt mit *yes* angezeigt wird.
- Es muss außerdem noch *cmake* und *cmake für Qt* installiert werden, damit das Ganze mit dem QtCreator verwendet werden kann.
- Warum chmod a+rwx (nicht rekursiv, nur auf Ordner) anstatt wie in anderen Tutorials *sudo chown -R pi:pi /usr/local/qt5pi*.
- Fehler im fertigen Produkt, weil -fontconfig nicht gesetzt wurde beim configure. Dadurch müssen entweder Fonts programmatisch in die eigene Anwendung[2] oder man kopiert die benötigten Fonts nach */opt/Qt5.10.1/lib/fonts*. Im Skript werden alle System Fonts (*/usr/share/fonts)* automatisch kopiert.
- *-system-freetype* geht nicht, da hier versucht wird die auf dem System installierte Lib zu verwenden, wenn diese nicht da ist, scheitert das Ganze[3]. Verwendung von *-qt-freetype*

References

1. http://www.tal.org/tutorials/building-qt-510-raspberry-pi-debian-stretch
2. Anleitung aus dem Moodle Raum "Qt 5.10.1 auf dem Raspberry Pi 3 compilieren"

[2] https://stackoverflow.com/questions/20751604/static-qt5-build-on-linux-how-to-handle-fonts-when-deploying
[3] http://doc.qt.io/qt-5/configure-options.html