

Programmdokumentation

Protokollierung

Andreas Daub (718170)



Christian Boigenreif (716732)



Inhaltsverzeichnis

Vorwort.....	4
Aufgabenstellung.....	4
Endprodukt.....	4
Erfasste Informationen.....	4
Gewonnene Erkenntnisse	5
Qt.....	5
Exceptionhandling.....	5
Fehler in moc-Dateien beim Kompilieren oder „qmake doesnt build below the.“	5
Keine Regel für „Dateiname“ vorhanden.....	5
Multithreading.....	5
GUI-Relevante Operationen außerhalb des Mainthreads.....	5
Sauberes Beenden von Threads.....	6
Versionsverwaltung Perforce.....	6
Build funktioniert vor dem hochladen ins Depot, beim herunterladen nicht mehr.....	6
Protokollierung von live Daten	6
Direkte Anbindung.....	7
WLAN Anbindung.....	7
LAN Anbindung.....	7
Aufzeichnen eines Videos.....	7
Programmbeschreibung.....	8
Klassendiagramm.....	8
Screens.....	8
MainWindow.....	8
Log-View.....	9
Chart-View.....	9
Stat-View.....	10
Vplayer (Replay).....	10
Navigation.....	11
Tests.....	12
Datenaufzeichnungsgeschwindigkeit.....	12
Beenden des Programms während einer Aufzeichnung.....	12
Schnelles drücken des Record-Buttons.....	12
Codedokumentation.....	12
Installationshandbuch.....	13
Beschreibung der Zielplattform.....	13
Kurze Einführung in die Entwicklungsumgebung und notwendige Pakete.....	13
Qt SDK.....	13
Qwt.....	13
SQLite3.....	13
SQLiteman.....	13
Phonon.....	14
VLC.....	14
Mancoder.....	14
Anwenderhandbuch.....	14
Starten des Programms	14
Datenverbindungen in Netzwerken	14
Beenden des Programms.....	14
Tipps und To Do für Nachfolger.....	15
Hilfreiche Quellen.....	15
Qt	15

Qwt.....	15
SQLite in Qt.....	15

Vorwort

Aufgabenstellung

Ziel ist die Protokollierung und Visualisierung von, bei der Fahrt anfallenden, Daten. Der Nutzer erhält die Möglichkeit seine Fahrt aufzuzeichnen, während dann die besagte Protokollierung stattfindet, welche anschließend visuell über Diagramme und Skizzen dargestellt werden soll. Zusätzlich sollen die GPS-Daten und ein Videobild erfasst werden, welches synchron zu den abgespielten Daten läuft.

Endprodukt

- Lauffähige Datenbank
- Visualisierung von relevanten Daten durch Diagramme / Tabellenform / Statistiken
- Aufzeichnen von Dummydaten inklusive Videobild und GPS-Koordinaten
- Abspielen einer Aufgezeichneten Fahrt

Erfasste Informationen

- Engine load value
- Engine coolant temperature
- Intake manifold pressure
- Engine RPM
- Vehicle speed
- Intake air temperature
- Air flow rate
- Throttle position
- Fuel rail pressure
- Trouble codes
- Wheel RPM
- Steering wheel direction
- Fuel type
- Fuel fill level
- Accelerator position
- Steering wheel protection
- Engine torque
- Gear
- Wheel rotation
- GPS coordinates

Gewonnene Erkenntnisse

Qt

Exceptionhandling

Beim Ausführen eines Programms in Qt kann es passieren dass es ohne ersichtlichen Grund abstürzt. Meldung des Compilers ist in der Regel „Programm abgestürzt“ oder mit etwas Glück die Fehlermeldung einer eingebundenen Library. Um ganz sicher zu gehen, dass es sich wirklich um einen Fehler handelt, der von dem werfen einer Exception verursacht wird, hilft es das Programm Schritt für Schritt zu debuggen. Beim werfen einer Exception kommt dann die Meldung des „SIGABRT“.

Grund dafür dass Qt standartmäßig kein Exceptionhandling betreibt ist, dass es für verschiedene Plattformen vorgesehen ist, welche aber nicht alle Exceptionhandling unterstützen.

Lösung: Beim bauen des Qt Creators bzw. des Qt SDKs gibt es ein Flag für Exceptions. Dieser ist standartmäßig so gesetzt, dass kein Exceptionhandling stattfindet.

Fehler in moc-Dateien beim Kompilieren oder „qmake doesnt build below the..“

Beim Kompilieren kommt es nach dem verschieben bzw. kopieren des Quellcodeverzeichnisses gelegentlich zu verschiedenen Fehlern. Die häufigsten sind z.B. Fehler in den „moc“-Dateien, welche beim bauen von Qt angelegt werden. Bzw. wenn man das Projekt bereinigt hat, die Meldung „Qmake doesnt build below the ..“.

Lösung des Problems ist das ändern des „Build-Pfades“ im Qt Creator unter dem Button „Projekt“. Hier muss der neue Pfad des verschobenen Projekts angepasst werden, da dieser unter Umständen immer noch auf den alten Pfad gesetzt ist.

Keine Regel für „Dateiname“ vorhanden

„Keine Regel für [Dateiname] vorhanden..“ ist eine Fehlermeldung, welche mehrere Gründe haben kann.

Eine Mögliche Ursache kann sein, dass beim Benutzen einer Versionsverwaltung vergessen wurde eine oder mehrere Dateien, dem Depot oder ähnlichem, hinzuzufügen. Zum beheben dieses Problems muss derjenige welcher das Projekt bzw. die Dateien hochgeladen hat, die fehlenden im Workspace die fehlenden Dateien dem Depot hinzufügen. Bei der Verwendung von Perforce im Visual Client, die Workspaceansicht wählen, rechts-klick auf die Dateien, welche hinzugefügt werden sollen, und „Mark for add“ wählen. Anschließend Submitten.

Eine andere Mögliche Ursache ist, dass die Projektname.pro Datei, aus welcher das Makefile generiert wird, noch einen Eintrag einer nicht mehr vorhandenen Datei beinhaltet. Zum Beheben dieses Problems, muss in der Projektname.pro Datei, im Abschnitt „SOURCES“ bzw „HEADERS“ der Name der nicht mehr vorhandenen Datei entfernt werden.

Multithreading

GUI-Relevante Operationen außerhalb des Mainthreads

Beim benutzen von Threads in Qt kommt es beim benutzen von Threads zu einer Fehlermeldung die folgender ähnlich ist: „Es ist nicht sicher GUI-relevante Operationen außerhalb des Main-“

Threads vorzunehmen“.

Grund für diese Fehlermeldung ist die, es keinem Thread, außer dem Mainthread gestattet ist, Veränderungen an der GUI vorzunehmen. Einzig mögliche Lösung des Problems ist die, es zu vermeiden Operationen im Bezug auf die GUI innerhalb von Threads vornehmen zu wollen.

Sauberes Beenden von Threads

„Qthread: Destroyed while thread is still running“ ist ein Hinweis darauf, dass ein Thread nicht sauber beendet wurde. Mögliche Ursache dafür ist, dass in einer Funktion ein Objekt erstellt wird, welcher den Thread starten soll. Problematisch dabei ist, dass die lokale Variable nach verlassen der Funktion vernichtet wird und damit der Thread unsauber beendet wird.

Einfache Lösung dafür ist die Objekte, welche solche Threads erzeugen, in der Klasse als Private zu deklarieren und die run() methode des Threads, welche automatisch aufgerufen wird nachdem der Thread mit start() ins leben gerufen wird, sauber zu beenden, indem man die Funktion einfach durch laufen lässt. Am besten dafür eignet sich eine Variable, welche von außen gesetzt werden kann, so dass im Fall der erfüllten Aufgabe des Threads, dieser sauber beendet werden kann.

Beim Dokumentieren des Quelltextes ist uns aufgefallen, das wir unseren Thread welcher die Datenaufzeichnung abwickelt nie richtig beenden, sondern nur anhalten. Dadurch existiert der Thread noch im Hintergrund, wird aber nie wieder gestartet. Das führt auch zu Problemen mit dem Netzwerk Client. Bei der nächsten Aufnahme wird ein neuer Thread mit neuem Client etc. gestartet. Dadurch das noch ein anderes Clientobjekt existiert, kann sich der neue Client nicht verbinden. Das liegt daran, das ein Client erst bei Aufruf des Destruktors die Verbindung trennt.

Versionsverwaltung Perforce

Build funktioniert vor dem hochladen ins Depot, beim herunterladen nicht mehr

Beim Umgang mit Perforce trat immer wieder das Phänomen auf, dass eine Version an einem Rechner funktioniert, ins Depot hochgeladen, an anderer Stelle heruntergeladen wird und nicht mehr funktioniert.

Grund dafür ist, dass derjenige welcher Das Projekt hochgeladen hat, es nicht vor dem hochladen bereinigt hat.

Es kommt häufiger vor, dass man z.B. Buttons, welche man nicht mehr braucht, löscht, aber deren automatisch generierten Funktionen nicht. Qt Creator ändert dann beim bauen diese Änderungen nicht in den bereits von ihm erstellten Dateien und das Programm funktioniert augenscheinlich Fehlerfrei, bis es von einem anderen, welcher sich das Projekt über ein Versionierungsprogramm herunterlädt, neu baut.

Wie bereits erwähnt hilft es das Projekt vor dem hochladen zu bereinigen. Bei fehlerhaften Überresten gelöschter Objekte wird der Compiler nun aufmerksam und meldet diese als Fehler. Problematisch dabei ist aber, dass die Fehler in den „ui_dateiname.h“ angezeigt werden. Diese sind aber vom Compiler generiert und können nicht verändert werden bzw werden sie beim bauen neu erzeugt und machen damit Änderungen zwecklos.

Grund für diese Fehlermeldung ist, dass in der „dateiname.h“ noch eine Funktion für das nicht mehr vorhandene Element deklariert ist. Die Lösung dieses Problems ist die deklaration der Funktion aus der „dateiname.h“ zu löschen.

Protokollierung von live Daten

Beim protokollieren von live Daten (aus dem Fahrzeug) ergeben sich gleich mehrere Probleme.

Direkte Anbindung

Das Programm stürzt bei der Aufzeichnung ab. Mit oder ohne Meldung über die Ursache.

Wie bereits im Kapitel Qt/Exceptionhandling erwähnt, kann Qt Exceptions nicht ohne weiteres behandeln. Da innerhalb des Interfaces der Fahrzeuganbindung aber Exceptions geworfen werden, sobald ein Wert nicht abgefragt werden kann oder nicht zur Verfügung steht, stürzt das Programm beim werfen einer solchen Exception ab.

Lösung des Problems ist im besagten Kapitel beschrieben.

WLAN Anbindung

Bei der WLAN Anbindung stürzt das Programm beim starten bzw. beim aufzeichnen ab. Grund dafür ist entweder der fehlgeschlagene Verbindungsaufbau zum Server, bei welchem innerhalb der „Boost-Library“, welche der Client nutzt, geworfen werden bzw. einem Timeout oder aber dass der Client im Hintergrund noch läuft, was zur Meldung „...Adress already in use..“ führt. Im letzteren Fall ist die einfachste Lösung den noch laufenden Prozess über das Terminal zu beenden.

LAN Anbindung

Bei der Verbindung über LAN füllt sich die Log-Tabelle so schnell dass das Programm nicht mehr bedienbar ist. Grund dafür ist, dass trotz Thread im Hintergrund, so viele Daten abgefragt, in die Datenbank und in die Tabelle geschrieben werden, dass das Programm nicht mehr bedienbar ist.

Das Auftreten des Problems unterscheidet sich von Rechner zu Rechner und ob das Betriebssystem auf 32- oder 64-Bit basiert.

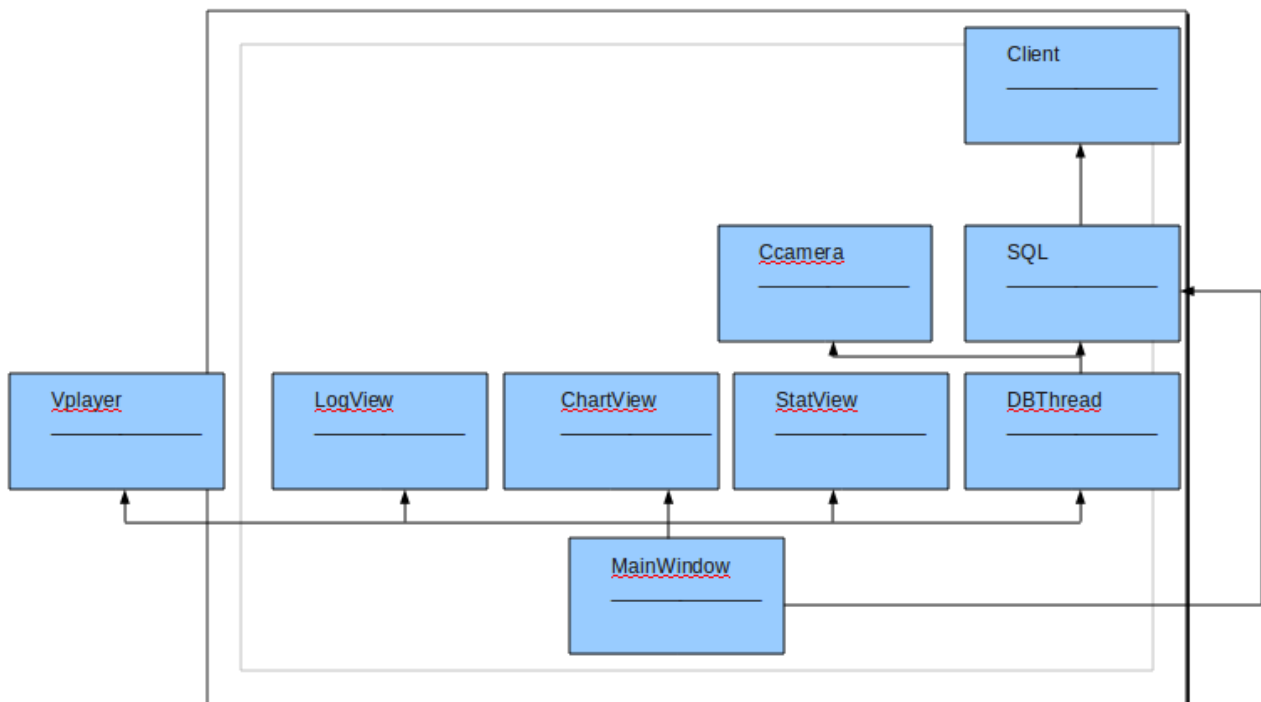
Lösung ist das einbauen einer bremse in das Aufrufen der Funktion, welche die Daten anfordert und weiter verarbeitet. Dazu muss in der „dbthread.cpp“ in der Funktion run() ein „sleep(x)“ nach dem Aufruf der „getData()“ Funktion der Datenbank. In der Regel reicht es, wenn man x so wählt, dass die Funktion pro Sekunde 100 mal aufgerufen wird.

Aufzeichnen eines Videos

Da keine Zeit da war um eine eigene Lösung zu entwickeln, welche Videos aufzeichnen kann, haben wir ein externes Programm Namens „Mencoder“ verwendet. Über einen System Aufruf starten wir dieses Programm entsprechend parametrisiert um ein Video im Home Verzeichnis aufzunehmen. Bis dieser Prozess dann wirklich aufzeichnet, vergeht unbestimmt viel Zeit. Diese muss man schätzen / messen, um die Datenaufzeichnung zu Synchronisieren. Das ist eine Notlösung, und sollte möglichst anders gelöst werden, evtl mit der Video For Linux API.

Programmbeschreibung

Klassendiagramm

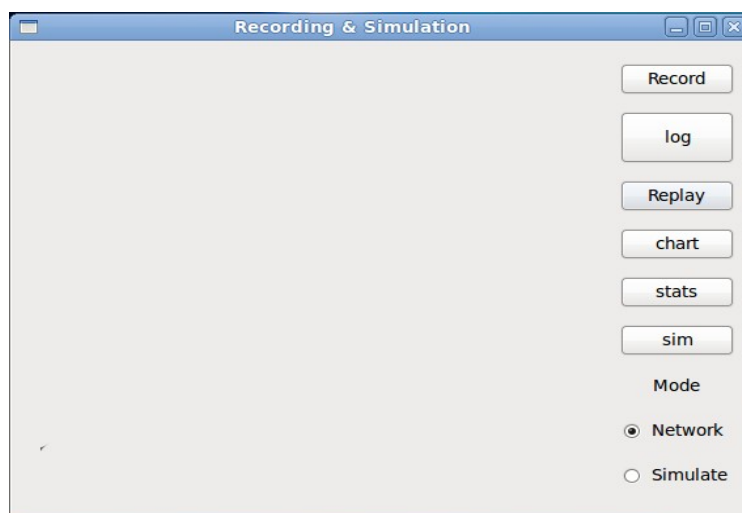


Screens

MainWindow

Mainwindow.cpp | Mainwindow.h | mainwindow.ui

Die Mainwindow-Klasse dient als Container für die übrigen GUI-Elemente. Im linken Teil des Fensters, welcher hier leer ist, befindet sich ein StackedWidget, welches die übrigen GUI-Elemente, Log View, Chart View und Stat View, beinhaltet. Diese können über die entsprechenden Buttons angesteuert werden, indem die entsprechende Funktion des Buttons den Index des StackedWidget setzt.



Log-View

Logview.cpp | logview.h | logview. Ui

Das Log View stellt die erfassten Daten in Tabellenform dar, welche durch den „Record“- bzw. den „Simulation“-Button aus der Netzwerkverbindung bzw. der Datenbank bezieht. Die Tabelle selbst ist als ein TableWidget realisiert und dient außerdem als ein Zwischenspeicher, um die Datenbankzugriffe zu minimieren, da aus ihr die Daten für die Charts und Statistiken bezogen werden. Die Tabelle kann über die Funktionen setItem() befüllt und durch Item() ausgelesen werden.

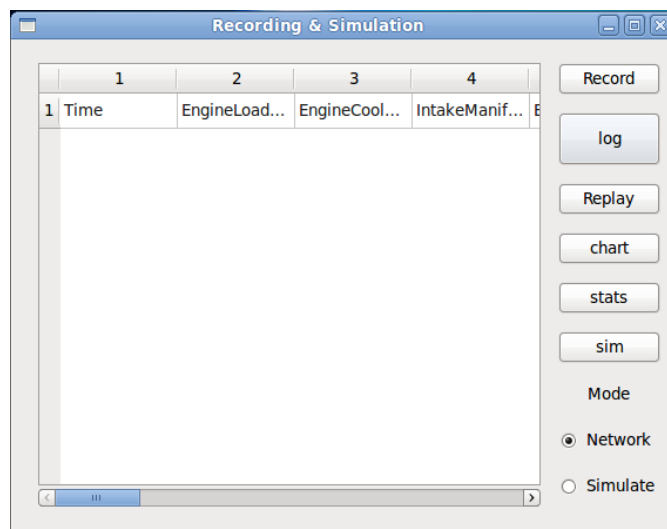
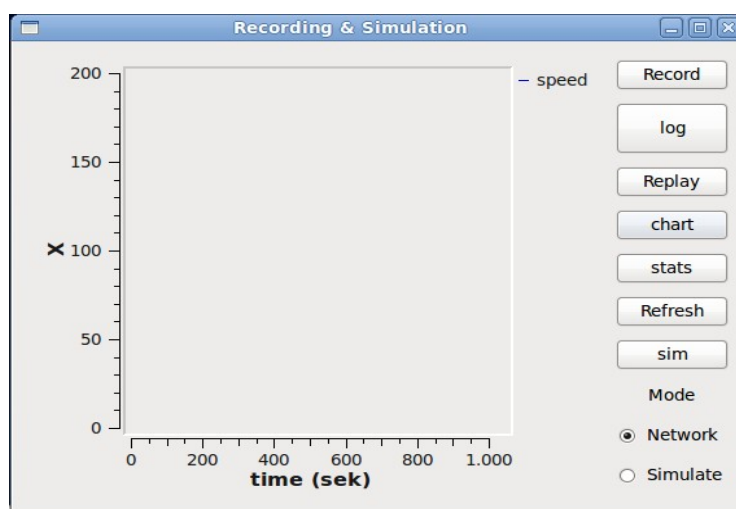


Chart-View

Chartview.cpp | chartview.h | chartview.ui

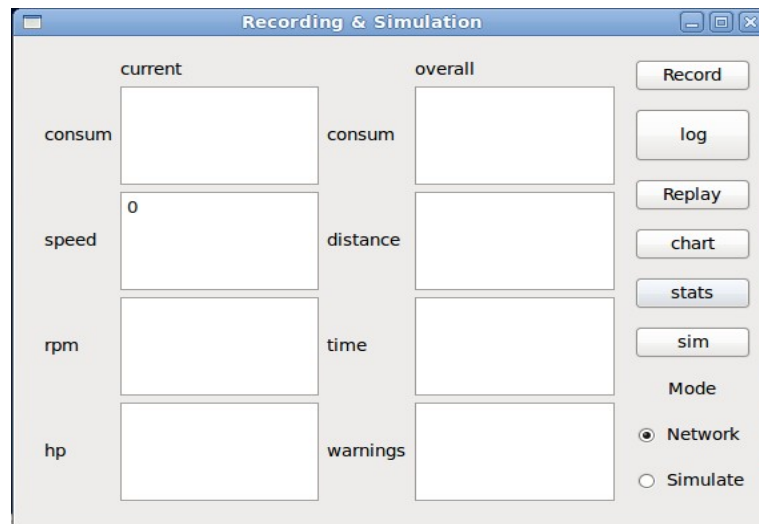
Das ChartView visualisiert die geloggtten Daten durch Verschiedene Kurven in einem Diagramm. Das Diagramm ist vom Typ QwtPlot, für dass die Library Qwt benötigt wird. Beim auswählen des ChartView Buttons werden die nötigen Informationen aus der Log-Tabelle geladen und über Arrays an QwtPlotCurve-Objekte über die Funktion setData() übergeben und anschließend durch die replot() Methode des QwtPlot neu gezeichnet.



Stat-View

Statview.cpp | Statview.h | Statview.ui

Das StatView visualisiert die Daten anhand durch verschiedene Berechnungen, die angestoßen werden, sobald man in diese Ansicht wechselt. Dazu werden die entsprechenden Werte aus der Log-Tabelle verwendet und in dem zugehörigen TextBrowser ausgegeben.



Vplayer (Replay)

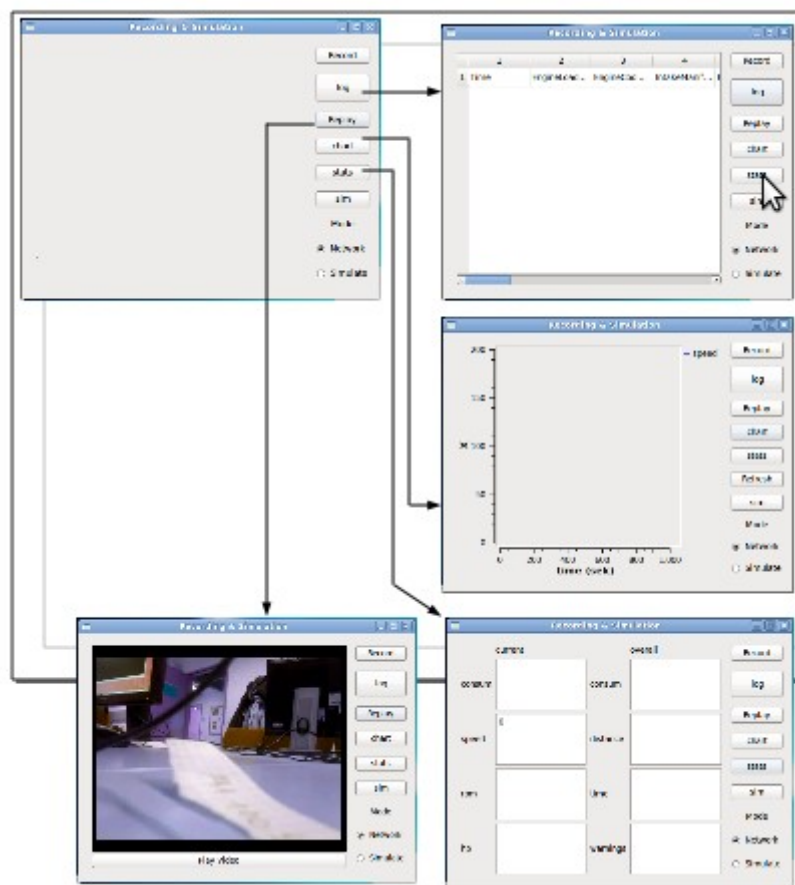
Vplayer.cpp | Vplayer.h | Vplayer.ui

Im Videoplayer Screen kann das Aufgezeichnete Video Synchron zu den Daten wiedergegeben werden (Funktioniert noch nicht richtig). Dazu wird ein Video in das Qvideo Widget geladen, und abgespielt sobald der Play Button gedrückt wird. Um die Daten zum Video Synchron einzuspielen wird die Differenz der Timestamps zwischen 2 Datensätzen ausgerechnet, und anschließend solange gewartet um den nächsten Datensatz einzuspielen.



Navigation

Wie bereits erwähnt dient das MainWindow als Container für die übrigen Screens. Über den Record-Button wird die Aufzeichnung des Programms angestoßen. Dazu wird ein Thread gestartet, welcher Daten vom Server holt bzw Testdaten in die Datenbank und die LogTabelle schreibt. Gleichzeitig wird die angeschlossene Kamera angesteuert, welche ab diesem Zeitpunkt bis zum erneuten Drücken des Buttons, welcher die Aufzeichnung beendet, das Geschehen aufzeichnet. Beim auswählen des Simulation-Buttons öffnet sich ein Dialog, welcher den Dateinamen einer Datenbank, die die aufgezeichneten Daten enthält, erfragen soll. Nach dem auswählen einer Datenbank startet ein Thread, welcher über einen Konsolenbefehl den VLC Player startet, der die Aufzeichnung der Kamera zeigt. Der Wechsel in die entsprechende View ist im unteren Bild dargestellt.



Tests

Datenaufzeichnungsgeschwindigkeit

Wir konnten 60 Vollständige Datensätze pro Sekunde vom Testserver der Verbindungsgruppe aufzeichnen.

Bei Tests mit dem Client der Simulationsgruppe konnten wir ca. 10 Pro Sekunde aufzeichnen, allerdings wurden Dabei auch Fehlerhafte Datensätze aufgezeichnet.

Dadurch das Exceptions unser Programm Beendet haben konnten wir nicht Testen wie viele wir direkt vom Auto empfangen können.

Da der Server der Übertragungsgruppe im Auto nur sporadisch funktioniert hatte, konnten wir auch dort nicht richtig Testen.

Beenden des Programms während einer Aufzeichnung

Wenn man das Programm während der Aufzeichnung beendet, wird der Mencoder Hintergrundprozess, welcher das Videobild aufzeichnet, nicht beendet. Dieser muss dann per Hand abgeschossen werden.

Schnelles drücken des Record-Buttons

Mencoder Hintergrundprozess wird nicht beendet, Videoaufzeichnung läuft weiter.

Das Clientobjekt wird nicht entfernt, dadurch kann sich bei erneuter Aufzeichnung nichtmehr mit dem Server verbinden.

Codedokumentation

Siehe Doxygen-Dokumentation.

Installationshandbuch

Beschreibung der Zielplattform

Zielplattform ist ein Linuxsystem. Über Cross-Compiler sollte man das Programm auf allen von QT unterstützten Plattformen benutzen können. Auch das Programm Mencoder was für die Videoaufzeichnung verwendet wird, sollte von Ubuntu bis Angstrom überall laufen.

Kurze Einführung in die Entwicklungsumgebung und notwendige Pakete

Alle benötigten Programme können über das Ubuntu Paketmanagement per apt-get, Aptitude und das SoftwareCenter installiert werden.

Folgende Pakete werden benötigt_

- **Qt SDK**

Basis-Entwicklungsumgebung.

- **Qwt**

Notwendig für die Visualisierung von Diagrammen.

Die Library lässt sich wie eben beschrieben installieren oder von der Anbieterseite <http://qwt.sourceforge.net/> herunterladen. Um die Library zu installieren reicht es den Anweisungen in der Readme zu folgen. Um sie auch zu benutzen und in den Qt Creator einzubinden müssen folgende zusätzliche Schritte unternommen werden:

- Idconfig benutzen um die Library hinzuzufügen:
Erstellen sie unter Ubuntu im Pfad /etc/ld.so.conf.d/ eine Datei z.B qwt.conf, und schreiben Sie in dieses File den Pfad zur Qwt-Library. Z.B. „/usr/local/qwt-XXXX/lib“. Starten die anschließend in der Konsole „ldconfig“.
- Ausführen von Programmen, die die so installierten Qwt libs nutzen, erfordert die Angabe des Verzeichnisses, in dem sich die libs befinden, in der Umgebungsvariable LD_LIBRARY_PATH:
export LD_LIBRARY_PATH=\$HOME/qwtverzeichnis/lib
- sudo cp /usr/lib/qt4/plugins/designer/libqwt_designer_plugin.so /usr/lib/qtcreator/plugins/ Achtung: Die Ordner können anders heißen.
- In das QT .pro File reinschreiben (das muss dann in der IDE gemacht werden):
INCLUDEPATH+=/usr/include/qwt-qt4
LIBS += -lqwt-qt4

- **SQLite3**

Library für Objektrelationales Datenbanksystem, welches aus Qt angesteuert werden kann.

- **SQLiteman**

Grafische Oberfläche zum Erstellen und Verwalten von SQLite3 Datenbankfiles.

- **Phonon**

Videowiedergabe innerhalb von Qt

- **VLC**

Videowiedergabe außerhalb von Qt, da Phonon ohne ersichtlichen Grund mal funktioniert und mal nicht.

- **Mancoder**

Benötigt für die Videoaufzeichnung

Anwenderhandbuch

Starten des Programms

a) Konsole

- Öffnen Sie die Konsole
- Wechseln sie mit „cd /Verzeichnispfad“ in das Verzeichnis des Programms
- Geben sie ./mehrwindow ein, um das Programm zu starten

b) Direkt

- Starten Sie das Programm durch einen Doppelklick auf die Datei namens „mehrwindow“

Datenverbindungen in Netzwerken

Für das Empfangen von Daten wird eine Netzwerkverbindung empfohlen. Am besten eignet sich eine LAN-Verbindung, da diese die maximale ausbeute des Datendurchsatzes sicherstellt. Ebenfalls eignet sich eine WLAN-Verbindung. Diese ist nicht ganz so schnell wie die über LAN, trotzdem ist mit der WLAN-Verbindung immer noch eine detaillierte Aufzeichnung der Fahrt möglich.

Beenden des Programms

Um das Programm sauber zu beenden, drücken Sie in der oberen rechten Ecke des Fensters auf das „X“.

Tipps und To Do für Nachfolger

- Perforce mit Vorsicht genießen. Am besten niemals das komplette Projekt aus-checken, sondern immer nur die Teile, die bearbeitet werden müssen.
- Früh integrieren um aufkommende Komplikationen auch früh festzustellen und entsprechende Maßnahmen zu unternehmen. Das verhindert, dass gegen Ende des Projekts Komplikationen auftauchen, die evtl. grundlegende Entscheidungen negativ beeinflussen oder hohen Aufwand verursachen.
- Bauen der Entwicklungsumgebung MIT gestelltem Exceptionflag.

Hilfreiche Quellen

Qt

- Online Reference Documentation von Nokia: <http://doc.qt.nokia.com/>
- Qt Reference Documentation von Trolltech: <http://doc.trolltech.com/3.3/index.html>

Qwt

- Beispiele und Hilfen auf: <http://qwt.sourceforge.net/>

SQLite in Qt

- <http://wiki.forum.nokia.com/>