Case Study: Driver Terminal for Forage Harvester

Burkhard Stubert
Solopreneur & Chief Engineer
Embedded Use (DBA)
www.embeddeduse.com

About Me

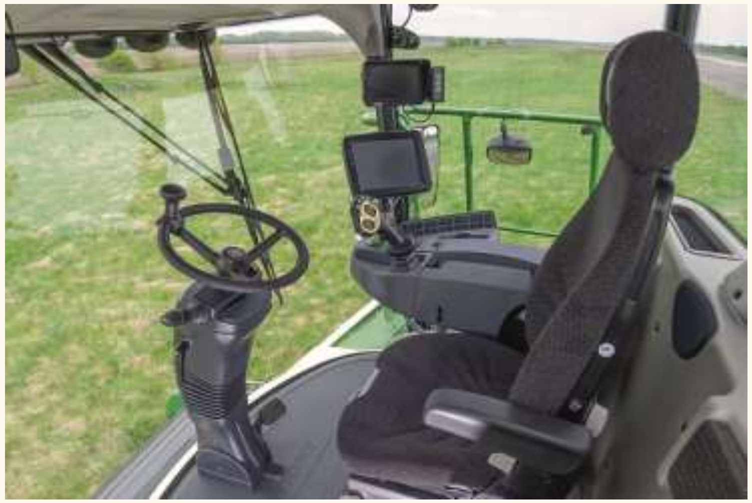
Burkhard Stubert

Solopreneur & Chief Engineer
Embedded Use (DBA)

Mail: burkhard.stubert@embeddeduse.com
Web: www.embeddeduse.com
Mobile: +49 176 721 433 16

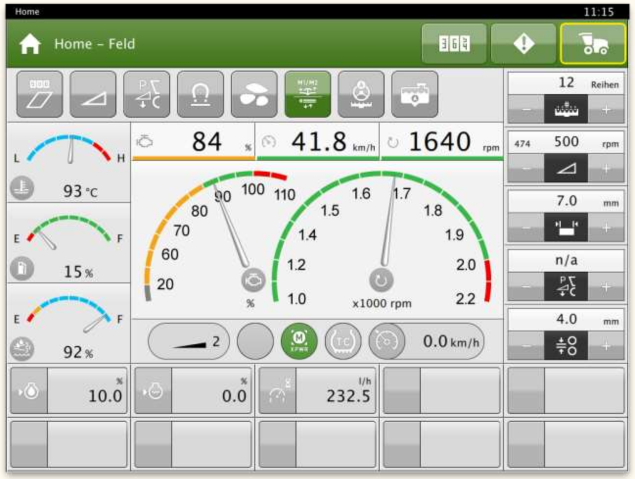
- Interesting Projects
  - In-flight entertainment system for US company
  - In-vehicle infotainment system for German tier-1 supplier
  - Driver terminal for Krone harvester
  - Internet radio for CSR
  - VoIP handset for xG
- 15+ years developing embedded and desktop systems
  - Especially with Qt and C++
  - Architecture, development, test, coaching, project lead
- Previous companies:
  - Nokia, Cambridge Consultants, Infineon, Siemens

Forage Harvester: Krone BigX 480/580



Project Schedule

Old

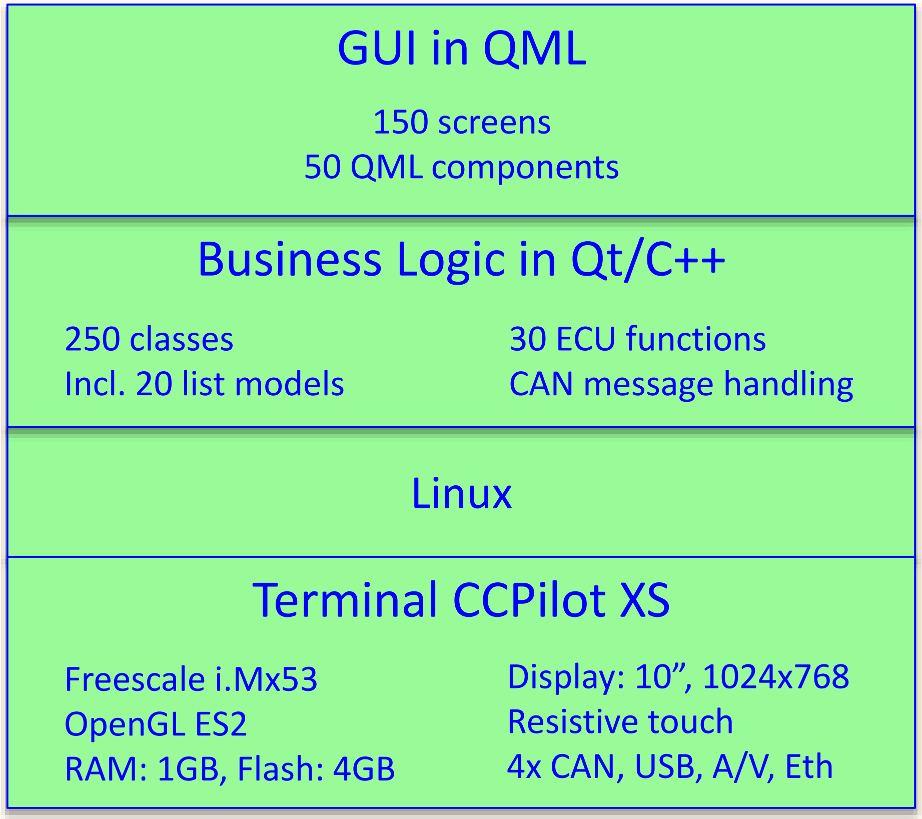


New

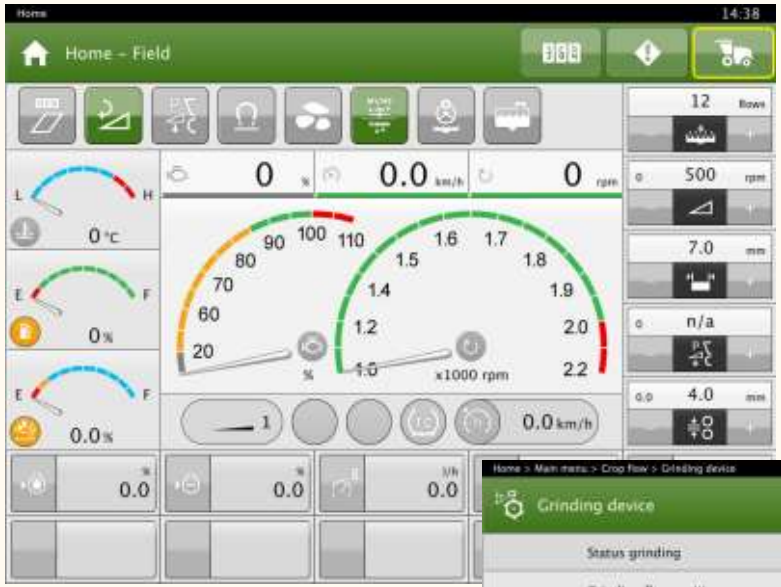
- **06/2012**: Project start
- **08/2012**: First prototype in corn harvest
  - Qt 4.8 on Windows XP and Intel Atom
- **02/2013**: Usability test with drivers
  - Qt 5.1 on Linux and ARM Cortex-A8
- **05/2013**: Alpha in grass harvest
- **08/2013**: Beta in corn harvest
- **11/2013**: Shown at Agritechnica
- **04/2014**: Product release

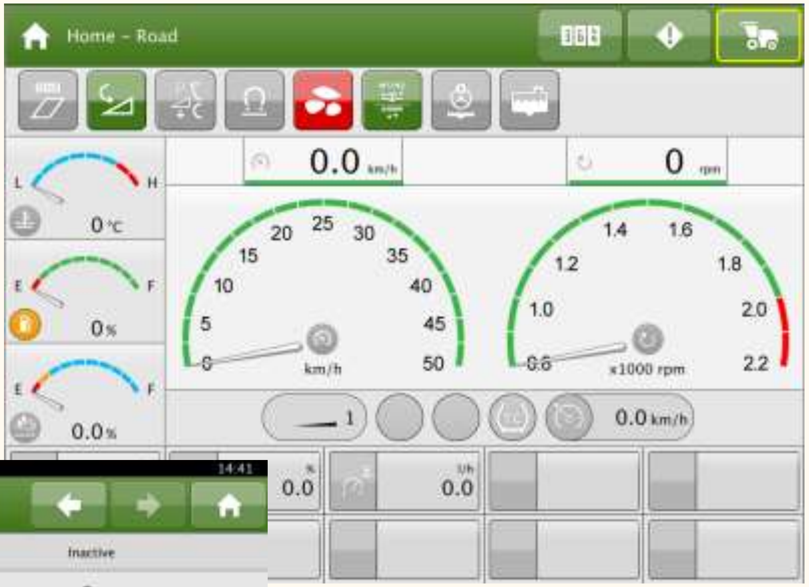
All done with 3 SW developers and 1 UI designer!

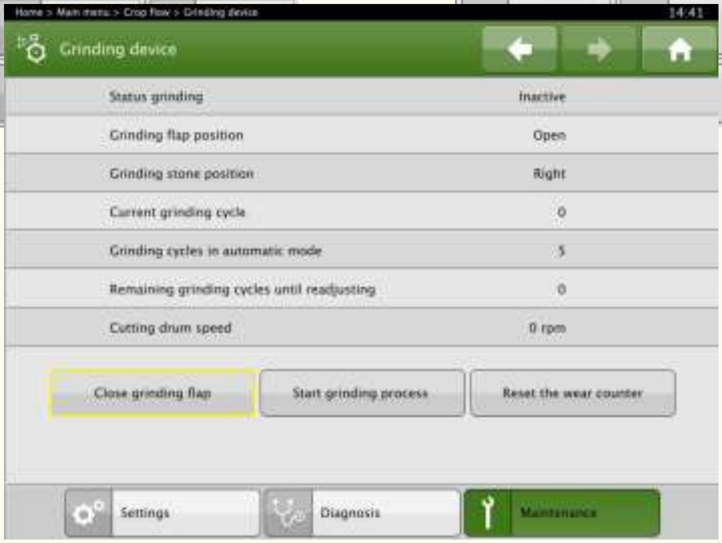
System Architecture



Modes: Field, Road, Maintenance

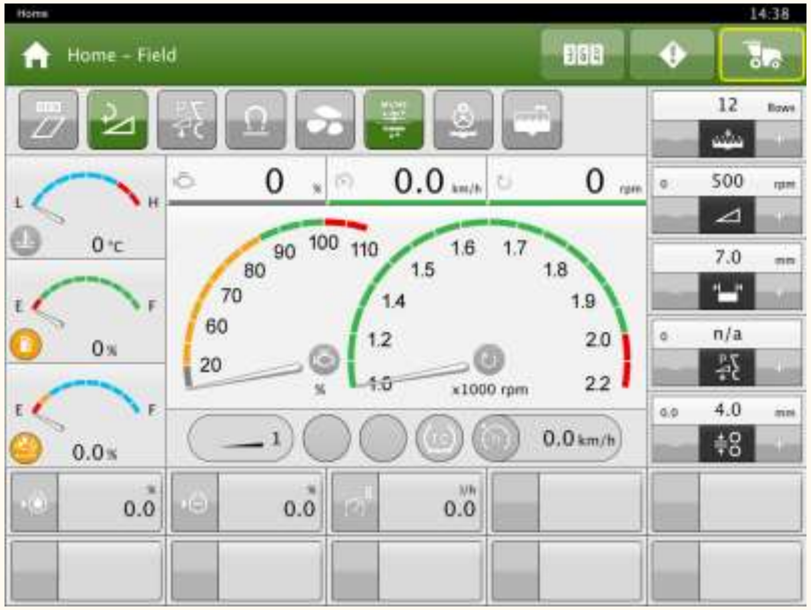


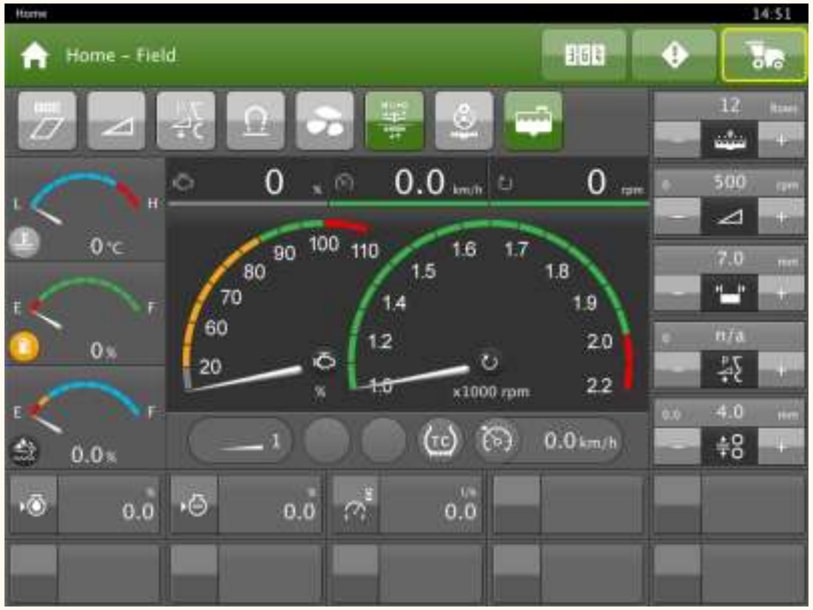




Status grinding	Inactive
Grinding flap position	Open
Grinding stone position	Right
Current grinding cycle	0
Grinding cycles in automatic mode	5
Remaining grinding cycles until readjusting	0
Cutting drum speed	0 rpm

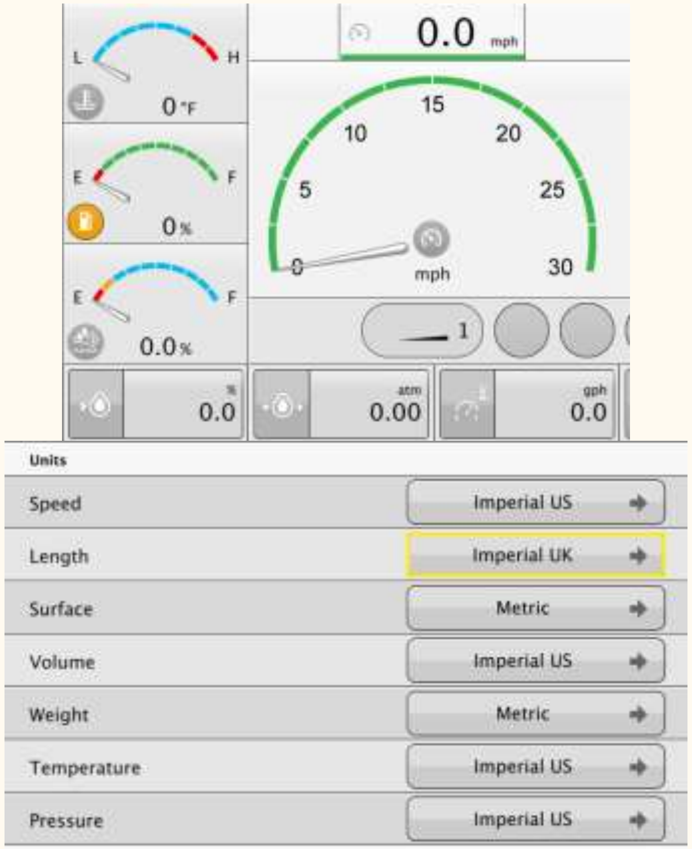
Modes: Day and Night





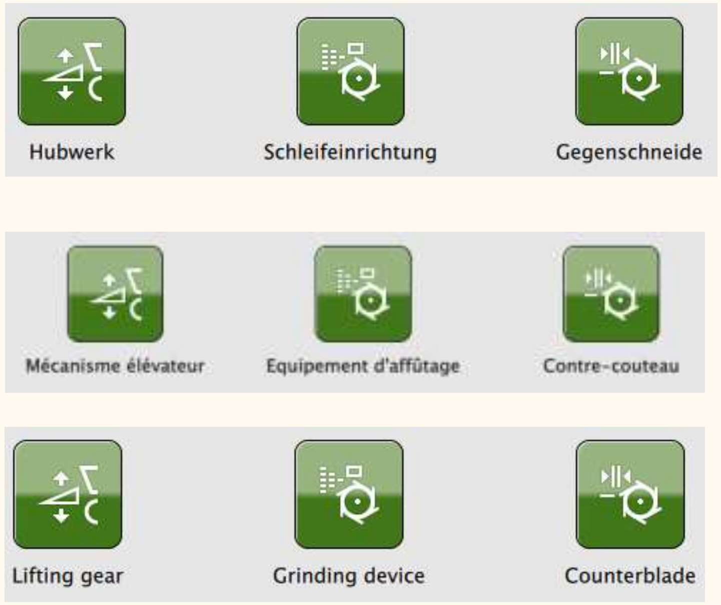
Internationalization

Metric vs. Imperial Units

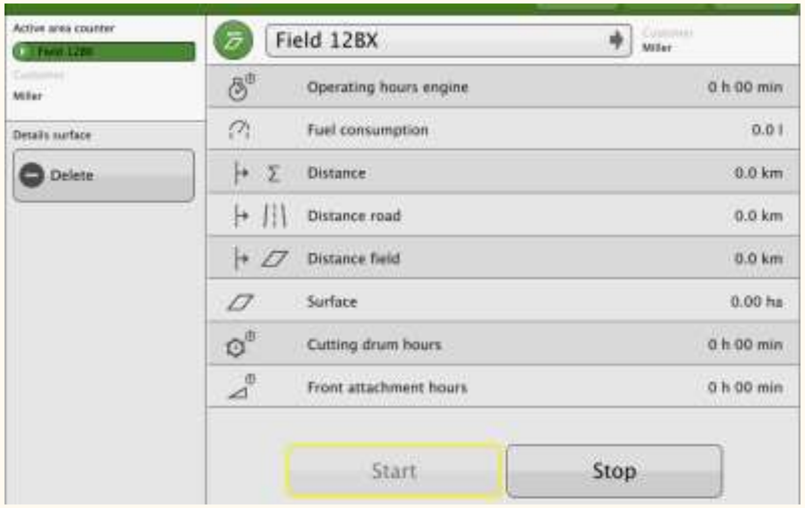


Units	
Speed	Imperial US
Length	Imperial UK
Surface	Metric
Volume	Imperial US
Weight	Metric
Temperature	Imperial US
Pressure	Imperial US

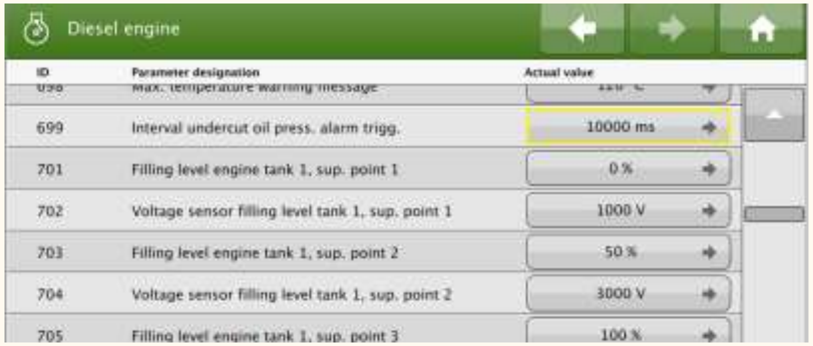
Multiple Languages



More Features



Operating hours engine	0 h 00 min
Fuel consumption	0.0 l
Distance	0.0 km
Distance road	0.0 km
Distance field	0.0 km
Surface	0.00 ha
Cutting drum hours	0 h 00 min
Front attachment hours	0 h 00 min



ID	Parameter designation	Actual value
698	Max. temperature warning message	[illegible] °C
699	Interval undercut oil press. alarm trigg.	10000 ms
701	Filling level engine tank 1, sup. point 1	0 %
702	Voltage sensor filling level tank 1, sup. point 1	1000 V
703	Filling level engine tank 1, sup. point 2	50 %
704	Voltage sensor filling level tank 1, sup. point 2	3000 V
705	Filling level engine tank 1, sup. point 3	100 %

- User input with both touch and rotary/push knob
- Crop area management
- Access to machine parameters for fine tuning
- On-Board Diagnosis

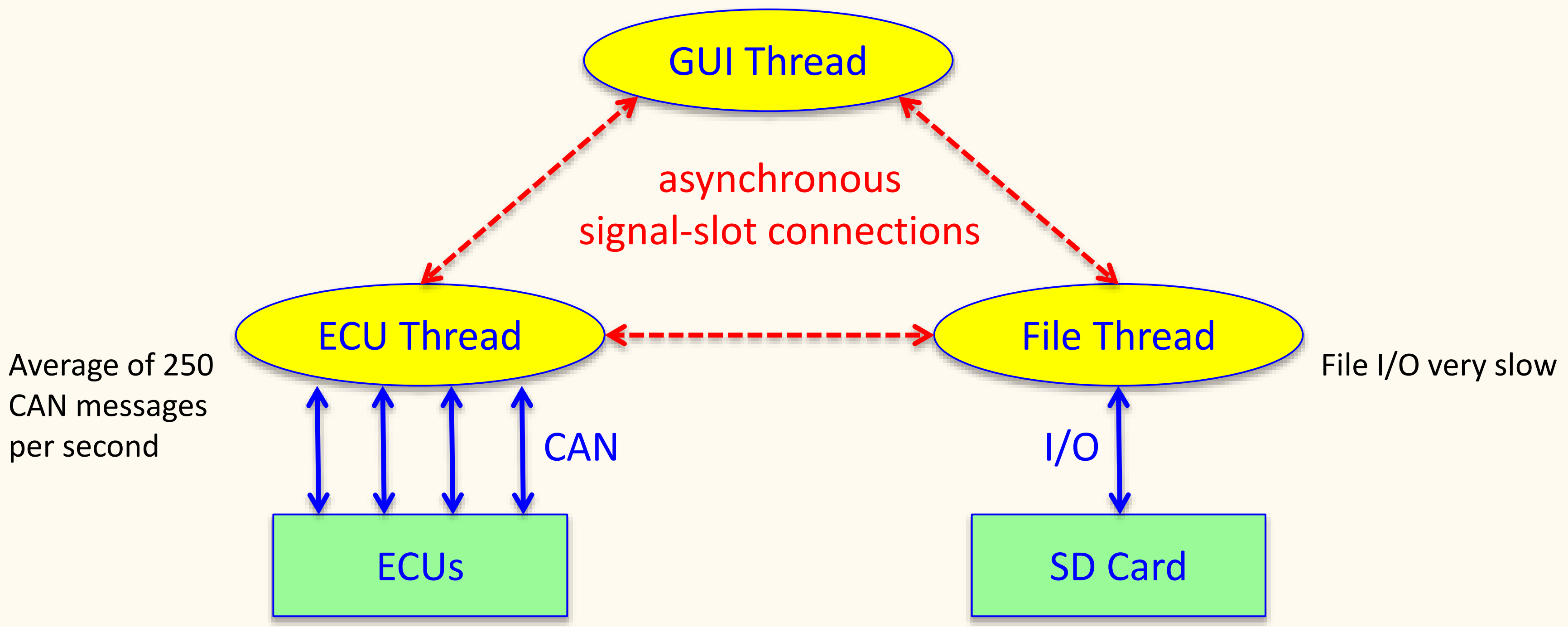
Agenda

- Multi-Threaded Architecture
- Know Your ListViews Well
- An Efficient Page Stack
- Themes for Day and Night Mode
- Internationalisation

Agenda

- **Multi-Threaded Architecture**
- Know Your ListViews Well
- An Efficient Page Stack
- Themes for Day and Night Mode
- Internationalisation

Dividing Application into Threads



© Burkhard Stubert, 2014

Setting up the Threads

```
int main(int argc, char *argv[]) {
    QGuiApplication app(argc, argv);

    QThread fileThread;
    FileManager::instance()->moveToThread(&fileThread);

    QThread ecuThread;
    EcuManager::instance()->moveToThread(&ecuThread);

    fileThread.start();
    ecuThread.start();

    QQuickView view;
    view.setSource(“main.qml”);
    view.show();
    return app.exec();
}
```

FileManager and EcuManger are single entry point into fileThread and ecuThread

All singletons used in several threads must be created **before** threads started!

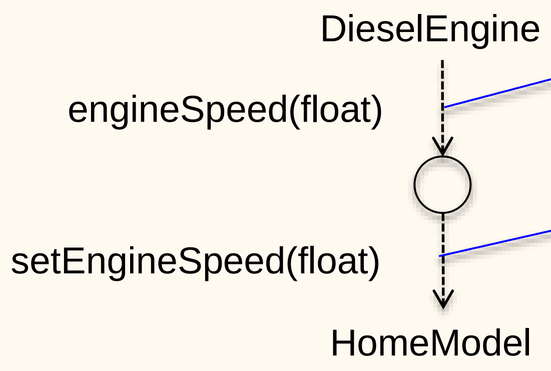
Starts event loop of QThread object. Needed for queued signal-slot connections

Inter-Thread Calls with Signals & Slots

```
// Set up inter-thread connections in HomeModel,
// which is in GUI thread
connect(EcuManager::instance()->getDieselEngine(),
        SIGNAL(engineSpeed(float)),
        this,
        SLOT(setEngineSpeed(float)));
```

Sender and receiver in different threads. Hence: Qt::QueuedConnection

Triggering the queued connection:



Meta object appends slot to event loop of GUI thread

When event loop of GUI thread executed next time, slot is called

Best of all: No thread synchronisation needed!

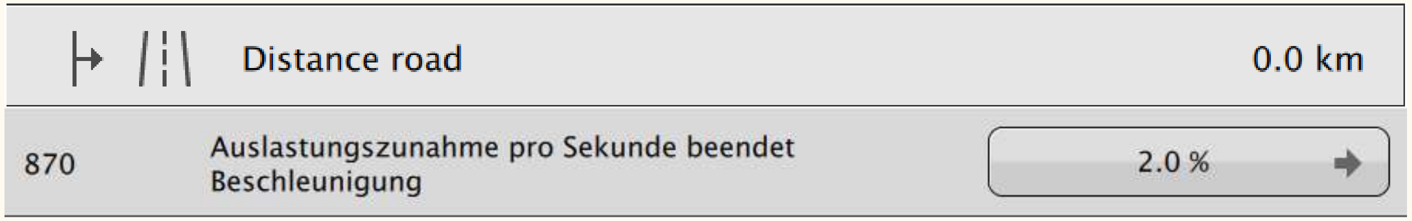
Caveats

- Arguments of queued connections must be builtins or have copy constructor
  - Examples: int, QString, const QList<float>&
- Don't pass pointers over queued connections
- Don't call into other thread directly
- Create singletons used in 2+ threads before threads started
  - **Before C++11**: No guarantee that synchronisation of singleton creation works on multi-core/processor machines
    - See "C++ and the Perils of Double-Checked Locking" by Scott Meyers and Andrei Alexandrescu, 2004, Dr. Dobbs Journal
  - **Since C++11**: Double-checked locking fixed
    - See "Double-Checked Locking is Fixed in C++11" by Jeff Preshing, 2013
    - But: instance() with synchronisation performs considerably worse than without

Agenda

- Multi-Threaded Architecture
- **Know Your ListViews Well**
- An Efficient Page Stack
- Themes for Day and Night Mode
- Internationalisation

Over-Generalized QML Delegates


Distance road 0.0 km

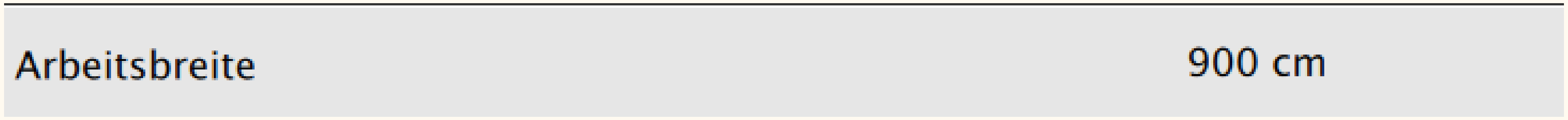
870 Auslastungszunahme pro Sekunde beendet Beschleunigung 2.0 %

- AllInOneDelegate instantiates 10 cells
  - For unused cells: visible: false
- AllInOneCell instantiates 5 types of cells
  - For unused types (4 out of 5): visible: false
- Problem:
  - For each visible row, ListView creates 50 objects
  - While scrolling, rows are deleted and created when becoming invisible and visible
- Result: Scrolling becomes erratic for 30+ rows

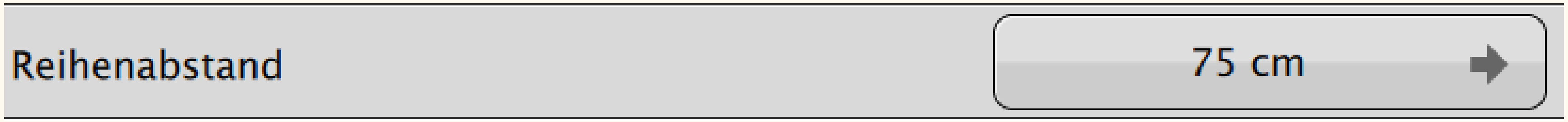
Over-Generalized QML Delegates (2)

- **Solution**:
  - Write a delegate for each type of ListView
  - Write a component for each type of cell
  - For each visible row, ListView creates 1 object for each cell
    - For the examples: 3 instead of 50 objects created

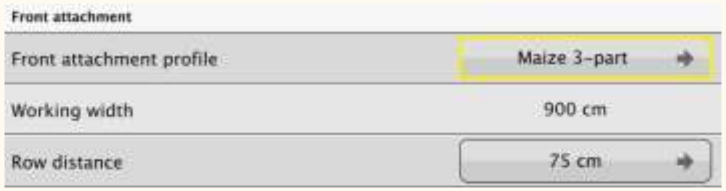
Bad Vibrations


Arbeitsbreite 900 cm

- Problem:
  - Due to harvester's vibrations and resistive touch, tapping always interpreted as flicking
- Solution:
  - Read-only cells used for flicking
  - Editable cells used for tapping (selection)

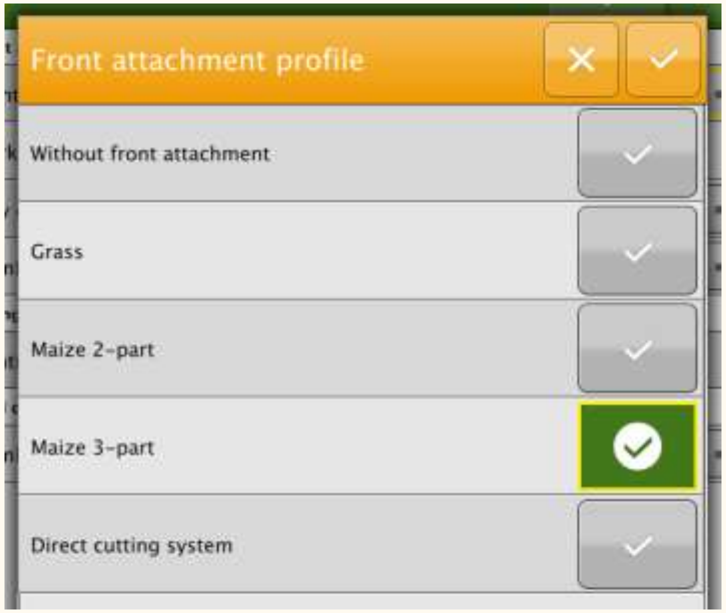

Reihenabstand 75 cm

Nested C++ List Models



Front attachment

Front attachment profile	Maize 3-part
Working width	900 cm
Row distance	75 cm



Front attachment profile

Without front attachment	✓
Grass	✓
Maize 2-part	✓
Maize 3-part	✓
Direct cutting system	✓

- Roles of SeasonView
  - pName: QString
  - pValue: EnumListModel*
- Roles of SingleChoiceDialog
  - eName: QString
  - eValue: int

Passing EnumListModel* to QML

```
// Delegate of SeasonView

Row {
    DisplayCell {
        text: pName
    }
    EditCell {
        text: pValue.eName
        onClicked: g_guiMgr.pushPage( “SingleChoiceDialog”, { “model”: pValue } );
    }
}
```

SingleChoiceDialog is ListView with pValue as its model

Passing EnumListModel* to QML (2)

```
// In SeasonModel.h

class SeasonModel : public QAbstractListModel {
    struct RowData {
        QString pName;
        EnumListModel *pValue;
    };

    QList<RowData *> m_data;

public slots:
    void receiveParameters(const QList<ParamData> &params);

    // More …
};
```

Slot (in GUI thread) triggered by signal from ECU thread

ParamData is copyable object providing the data of a row in SeasonView

© Burkhard Stubert, 2014

Passing EnumListModel* to QML (3)

```
// In SeasonModel.cpp

QVariant SeasonModel::data(const QModelIndex &index, int role) const {
    // Some checks ...
    QVariant v;
    switch (role) {
    case ROLE_PNAME:
        return m_data[index.row()]->pName;
    case ROLE_PVALUE:
         v.setValue(static_cast<QObject*>(m_data[index.row()]->pValue));
         return v;
    default:
         return v;
    }
}
```

Must be QObject*, because only pointer type registered with QVariant. Custom pointer types cannot be registered with QVariant.

Wrapping Enum in ListModel

```
class EnumListModel : public QAbstractListModel {
    Q_OBJECT

    Q_PROPERTY(int eIndex READ getEIndex WRITE setEIndex
                    NOTIFY eIndexChanged)
    Q_PROPERTY(int eName READ getEName
                    NOTIFY eIndexChanged)
    Q_PROPERTY(int eValue READ getEValue
                    NOTIFY eIndexChanged)

signals:
    void eIndexChanged();

public:
    EnumListModel(QMap<int, QPair<QString, int> >&(*func)(),
                        QObject *parent = 0)

    // More …
```

Index of the currently selected item in the ListView

eName and eValue depend fully on eIndex. Hence, no setter and same notification signal as eIndex

Function with mapping from index to pair of enum string and value

Getters, setters for properties. roleNames(), data(), rowCount() for QAbstractListModel

Wrapping Enum in ListModel (2)

```
// In ParameterEnums.h
class ParameterEnums : public QObject {
    Q_OBJECT
    Q_ENUMS(AttachmentProfile)
public:
    enum AttachmentProfile {
        AP_NONE = 0, AP_GRASS, AP_MAIZE_2 //…
    };

    static QMap<int, QPair<QString, int> > &getApEnum() {
        static QMap<int, QPair<QString, int> > m;
        if (m.isEmpty()) {
            m.insert(0, qMakePair(QString(“Without front attachment”),  AP_NONE));
            m.insert(1, qMakePair(QString(“Grass”), AP_GRASS));
            m.insert(2, qMakePair(QString(“Maize 2-part”), AP_MAIZE_2));
            // …
        }
        return m;
    }
```

Wrapping Enum in ListModel (3)

```
// In SeasonModel.cpp

// In constructor
m_mapper = new QSignalMapper(this);
connect(m_mapper, SIGNAL(mapped(int)), this, SLOT(emitDataChanged(int)));

// In slot receiving data from ECU thread
void SeasonModel::receiveParameters(const QList<ParamData> &params) {
    for (int i = 0; i < params.count(); ++i) {
        RowData *rd = new RowData;
        if (params[i].pValueType == ParamData::VT_ENUM) {
            rd->pName = params[i].pName;
            rd->pValue = new EnumListModel(params[i].enumFunc, this);
            connect(rd, SIGNAL(eIndexChanged()), m_mapper, SLOT(map()));
            m_mapper->setMapping(rd, i);
        }
        // other value types
```

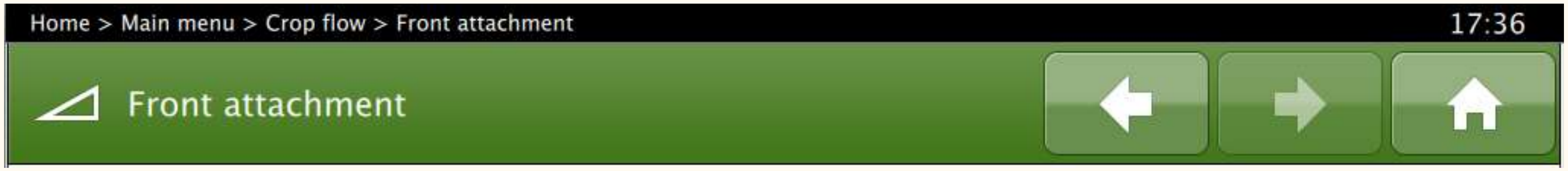
Emits dataChanged(QModelIndex, QModelIndex) of QAbstractListModel for row given as argument

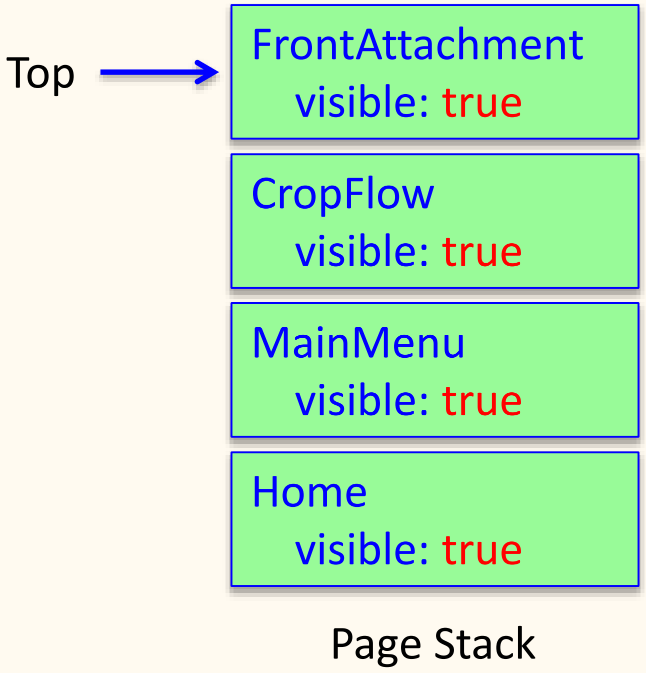
For example: ParameterEnums::getApEnum()

Agenda

- Multi-Threaded Architecture
- Know Your ListViews Well
- **An Efficient Page Stack**
- Themes for Day and Night Mode
- Internationalisation

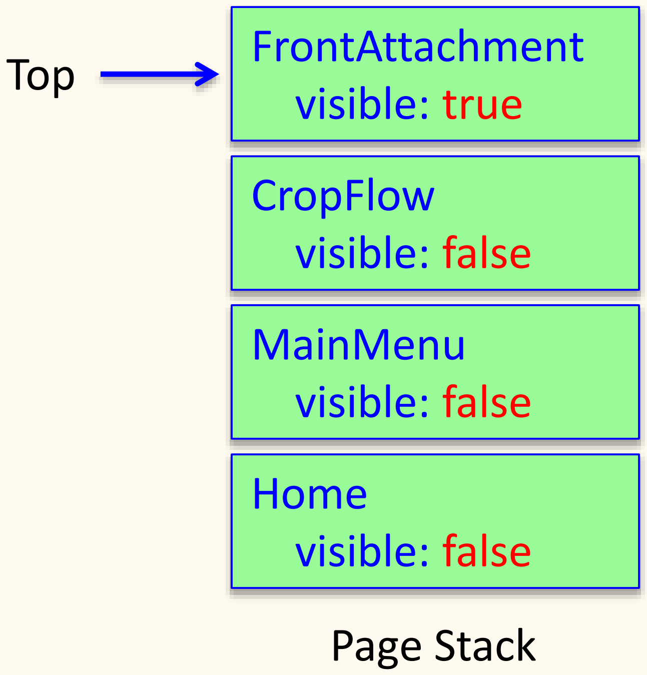
Page Stack: Bad Implementation





- **Problem**:
  - Every QML item with "visible: true" is rendered
  - 4 full screens of 1024x768 pixels rendered!
- GUI hardly responsive!
- Occurred in PageStack of Qt 4.8

Page Stack: Good Implementation



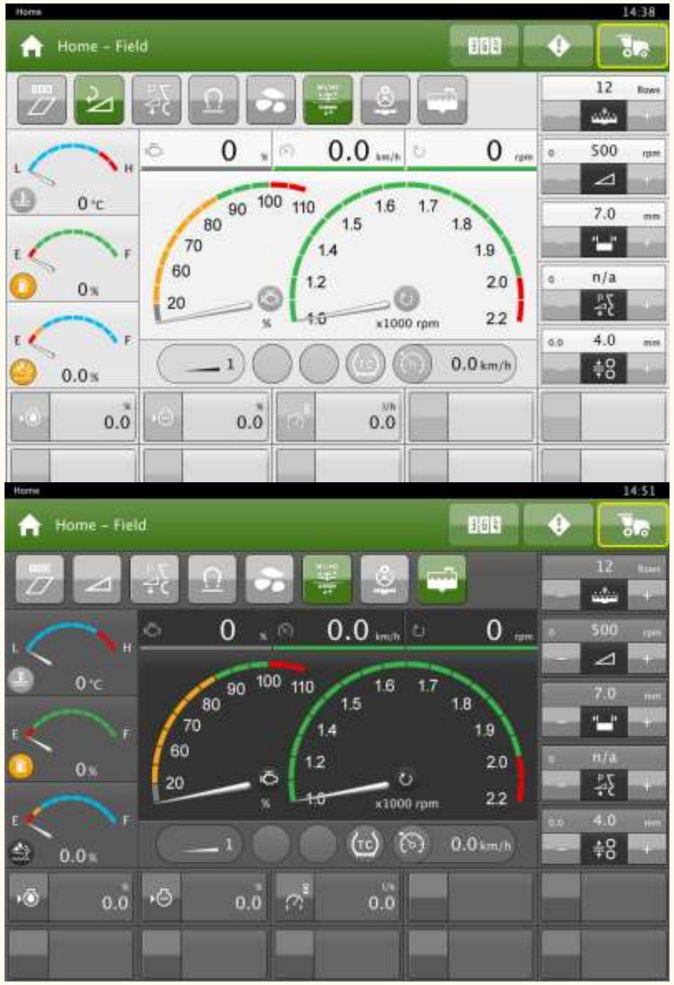
Page Stack

- **Solution**:
  - All pages except top page have "visible: false"
  - If transition animated, set "visible: false" when animation finished
- Correct in StackView since Qt 5.1
  - But: Too much Javascript!
- Note: Two full screens during animation may be too much for some GPUs

Agenda

- Multi-Threaded Architecture
- Know Your ListViews Well
- An Efficient Page Stack
- **Themes for Day and Night Mode**
- Internationalisation

Day and Night Mode



- Theme
  - GUI Structure unchanged
  - Look changes: text and background colours, images
- Change between day and night theme at runtime
- Solution: QQmlFileSelector

Changing Theme in C++

```
// In ThemeManager.cpp
// Exposed to QML as singleton

void ThemeManager::setTheme(const QString &theme) {
    if (m_theme == theme)
        return;

    m_theme = theme;
    QStringList xsel;
    if (m_theme == “night”) {
        xsel << m_theme;
    }

    QQmlFileSelector::get(m_qmlEngine)->setExtraSelectors(xsel);
    emit themeChanged();
}
```

Triggered in QML by ThemeManager.theme = “night”

Searches for …/+night/Theme.qml

Day theme is default with empty selector list
Searches for …/Theme.qml

Extra selectors searched before locale and platform selectors

Notify QML about theme change

Theme Selection in QML

```
// In Main.qml

property alias g_theme: themeLoader.item

Loader {
    id: themeLoader
    source: Qt.resolvedUrl(“Theme.qml”)
}

Connections {
    target: ThemeManager
    onThemeChanged: {
        themeLoader.source = Qt.resolvedUrl(“Theme.qml”)
    }
}
```

Name of theme file same for day and night mode

Singleton managing themes

Loads file variant for new theme
Binding for every theme variable changes

Theme Files

DayTheme.qml

```
Item {
    property color textColor: “black”
    property color lineColor1: “#333333”
    property color backgroundColor1: “#E6E6E6”

    property url bg_centralArea:
“images/bg_CentralArea.png”
    property url bg_tableRow1:
“images/bg_TableRow1.png”
    property url ic_accelerationRamp1:
“images/AccelerationRamp1.png”

    // Many more …
}
```

NightTheme.qml

```
Item {
    property color textColor: “white”
    property color lineColor1: “#000000”
    property color backgroundColor1: “#595959”

    property url bg_centralArea:
“images/bg_CentralArea.png”
    property url bg_tableRow1:
“images/bg_TableRow1.png”
    property url ic_accelerationRamp1:
“images/AccelerationRamp1.png”

    // Many more …
}
```

Property names used in QML code instead of actual values
source: g_theme.bg_centralArea

Organisation in File System

```
/path/to/my/app/qml
    +night/
        images/
            bg_CentralArea.png
            bg_TableRow1.png
            AccelerationRamp1.png
        Theme.qml
    images/
        bg_CentralArea.png
        bg_TableRow1.png
        AccelerationRamp1.png
    Theme.qml
```

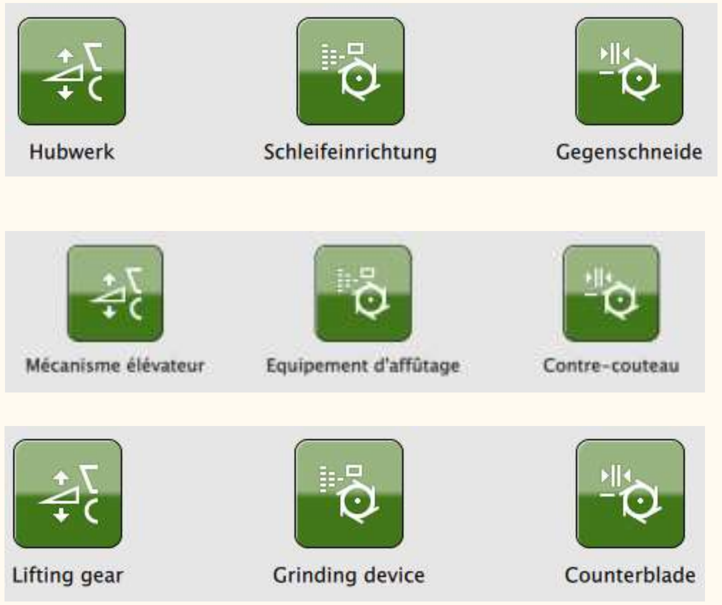
Files for night theme in variant directory

Files for day/default theme in plain directory

Agenda

- Multi-Threaded Architecture
- Know Your ListViews Well
- An Efficient Page Stack
- Themes for Day and Night Mode
- **Internationalisation**

Multiple Languages



- Change between languages at runtime
- Two solutions:
  - [S1] qsTr bound to languageChanged property
  - [S2] "Theme" for each language

[S1] QML Client Code

```
// In a QML file

Text {
    text: qsTr(“Areas”) + g_tr.languageChanged
    // …
}
```

When property *g_tr.languageChanged* changes, property *text* must be re-evaluated and qsTr(“Areas”) re-executed
Translation of “Areas” assigned to property *text*

[S1] Changing Language in C++

```
// In TranslationMgr.h
// Exposed to QML as singleton g_tr

class TranslationMgr : public QObject {
    Q_OBJECT

    Q_PROPERTY(QString languageChanged
                      READ getLanguageChanged
                      NOTIFY languageChanged)

    QString getLanguageChanged() const {
        return “”;
    }

signals:
    void languageChanged();
```

Must return empty string to keep translated string unchanged

[S1] Changing Language in C++

```
// In TranslationMgr.cpp

void TranslationMgr::setLanguage(const QString &lang) {
    if (m_lang == lang)
        return;

    // Install proper QTranslator for language
    // …

    QLocale::setDefault(lang);
    emit languageChanged();
}
```

Triggered in QML by g_tr.language = “de_DE”

Load qm file for *lang* (e.g., “de_DE”)

Set default locale to *lang* (e.g., “de_DE”)

Notify QML about language change

[S2] QML Client Code

```
// In a QML file

Text {
    text: g_tr.s_areas
    // …
}
```

Simpler than
qsTr("Areas") + g_tr.languageChanged

[S2] Changing Language in C++

```
// In TranslationMgr.cpp
// Exposed to QML as singleton TranslationMgr

void TranslationMgr::setLanguage(const QString &lang) {
    if (m_lang == lang)
        return;

    // Install proper QTranslator for language
    // …

    QLocale::setDefault(lang);
    emit languageChanged();
}
```

Triggered in QML by
TranslationMgr.language = “de_DE”

Load qm file for *lang* (e.g., “de_DE”)

File selector supports locales
Searches for …/+de_DE/Translations.qml
Use proper default (e.g., en_US)

Notify QML about
theme change

[S2] Translation Selection in QML

```
// In Main.qml

property alias g_tr: trLoader.item

Loader {
    id: trLoader
    source: Qt.resolvedUrl("Translations.qml")
}

Connections {
    target: TranslationMgr
    onLanguageChanged: {
        trLoader.source = Qt.resolvedUrl("Translations.qml")
    }
}
```

Name of translation file same for all languages

Singleton managing translations

Loads file variant for new language
Binding for every translated string changes

© Burkhard Stubert, 2014

[S2] Translation Files with qsTr

Translations.qml (en_US)

```
Item {
    property string s_arm_rest:
                qsTr(“Arm rest”)
    property string s_settings:
                qsTr(“Settings”)
    property string s_areas:
                qsTr(“Areas”)
```

Property names used in QML code instead of qsTr(“string”)
    text: g_tr.s_areas

Translations.qml (de_DE)

```
Item {
    property string s_arm_rest:
                qsTr(“Arm rest”)
    property string s_settings:
                qsTr(“Settings”)
    property string s_areas:
                qsTr(“Areas”)
```

qsTr() does the actual translation

© Burkhard Stubert, 2014

[S2] Translation Files without qsTr

Translations.qml (en_US)

```
Item {
    property string s_arm_rest: “Arm rest”
    property string s_settings: “Settings”
    property string s_areas: “Areas”
```

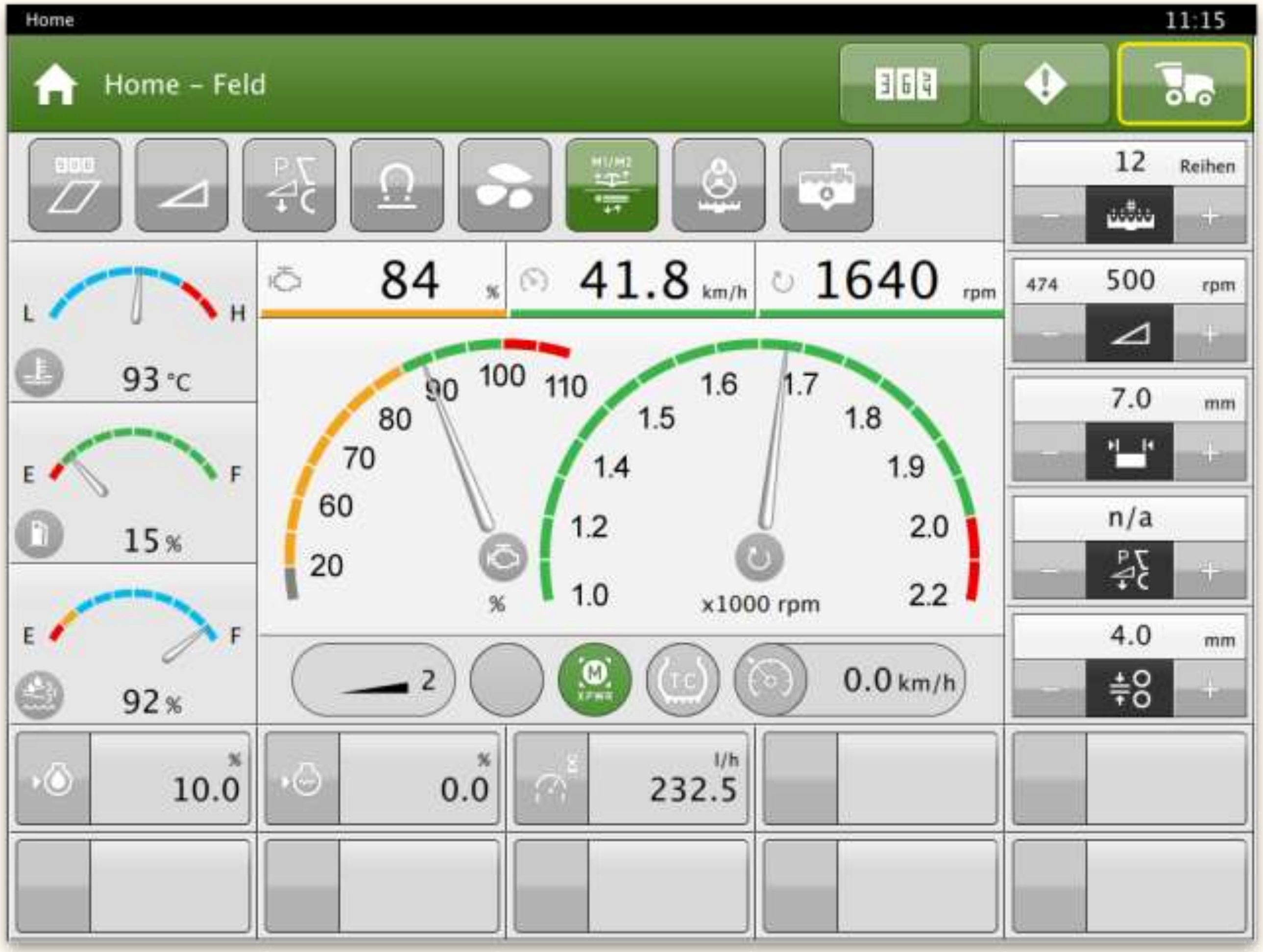
Translations.qml (de_DE)

```
Item {
    property string s_arm_rest: “Armlehne”
    property string s_settings: “Einstellungen”
    property string s_areas: “Flächen”
```

Translations written directly into QML “translation” files

© Burkhard Stubert, 2014

Terminal Krone Big X 480/580


Home
11:15
Home – Feld
12 Reihen
474
500 rpm
7.0 mm
n/a
4.0 mm
84 %
41.8 km/h
1640 rpm
L
H
93 °C
E
F
15 %
E
F
92 %
20
60
70
80
90
100
110
%
1.0
1.2
1.4
1.5
1.6
1.7
1.8
1.9
2.0
2.2
x1000 rpm
2
0.0 km/h
% 10.0
% 0.0
l/h 232.5