muPlantPython

Release 0.1

Lennart Schink

May 27, 2023

TABLE OF CONTENTS

This Software is part of muPlant Project of University of Kassel. It implements basic functions for WareHouse Management.

CHAPTER

ONE

COMMUNICATION STANDARDS

- TCP/IP for communication with ABB Robot
- OPC UA with other muPlant stations
- RFID to communicate with turtle bots
- uEye Camera with openCV and arUco markers for automated storage detection

CHAPTER

TWO

LEARNINGS FROM EARLIER STUDIES

Sebastian Hübler has published hie practical studies in 2019. He evaluated methods to detect cups in muPlant storage by using two different cameras. Regarding the detection with arUco markers he made some helpful analysis:

- low resolution leads to better detection results
- minimal/ maximal distance uEye tp marker: 17mm/ 745mm
- ambient light has significant influence
- auto focus maybe a big issue
- detection best works while robotic arm is without movement

2.1 Further thoughts:

- if ambient light and other surcumstances are not good:
- reduce image size to criticl area only
- calibrate location of camera to maximize image size reduction
- better detection with custom filter which smoothes and increases contrast?
- other camera which has no autofocus

2.2 Getting started

This project uses a requirements.txt which can be used to set up the project running.

```
pip install -r requirements.txt
```

Once the program is running the requirements.txt can be updated with following command

```
pip freeze > requirements.txt
```

2.2.1 Read Me

```
This is the python implementation for originally c# warehouse manager and warehouse␣
↪controller
```

2.2.2 Modules and Scripts

This list contains all created Modules and scripts created for this software.

Main python file

```
'''
This is the entrancee file of Python-Implementation of muPlant Warehouse Manager.
Author: L.Schink
Date: 11.05.2023

'''
import sys
from pathlib import Path
from PySide6.QtGui import QGuiApplication
from PySide6.QtQml import QQmlApplicationEngine
from src.model import ProductListModel
from src.model.InventoryModel import InventoryModel, createTableModel
from src.model.ProductSummaryListModel import ProductSummaryListModel, ␣
↪InventoryFilterProxyModel, createSummaryModel
from src.controller.InventoryController import InventoryController
from src.controller.EventlogController import EventlogController
from src.controller import websocketController
from src.cameraApplication import cameraProcessing


if __name__ == "__main__":
    '''Create Basic Application Class and QMLEngine'''
    app = QGuiApplication(sys.argv)
    engine = QQmlApplicationEngine()

    '''Define pathes for saved data, load the data in QML-usable data models and set mata␣
↪model as RootContext'''
    PRODUCTLIST = Path(__file__).resolve().parent / "src" / "data" / "Produkte.db"
    STORAGEDATA = Path(__file__).resolve().parent / "src" / "data" / "StorageData.db"
    CAMAPP_QML = "../cameraApplication/qml/CameraAppMain.qml"

    # simple Productlist
    productListModel = ProductListModel.ProductListModel(ProductListModel.
↪getProducts(PRODUCTLIST))
    engine.rootContext().setContextProperty("productListModel", productListModel)

    # tableModel for storage-visualization
    inventoryModel = InventoryModel(createTableModel(STORAGEDATA, PRODUCTLIST))
    engine.rootContext().setContextProperty("inventoryModel", inventoryModel)
```

(continues on next page)

(continued from previous page)

```
    # model combined from productlist and storage. Provides list with storage data and␣
↪quantity
    productSummaryModel = ProductSummaryListModel(createSummaryModel(STORAGEDATA,␣
↪PRODUCTLIST))
    engine.rootContext().setContextProperty("productSummaryModel", productSummaryModel)

    # model based on productSummaryModel but can be filtered dependding on quantity
    inventoryFilterModel = InventoryFilterProxyModel(model= productSummaryModel)
    engine.rootContext().setContextProperty("inventoryFilterModel", inventoryFilterModel)

    # create EventlogController instance
    eventlogController = EventlogController()
    engine.rootContext().setContextProperty("eventLogController", eventlogController)

    # create InventoryController instance
    inventoryController = InventoryController(model=inventoryModel, eventcontroller =␣
↪eventlogController, productlist = productListModel)
    engine.rootContext().setContextProperty("inventoryController", inventoryController)

    # set inventoryModel as property of productSummaryModel
    productSummaryModel.setStorageModel(inventoryModel)

    # register controller to make them availlable in qml files.
    wsController = websocketController.WebsocketController(eventlogController)
    engine.rootContext().setContextProperty("wsController", wsController)

    # add camApp to engine
    camApp = cameraProcessing.VideoPlayer()
    engine.rootContext().setContextProperty("camApp", camApp)
    engine.addImageProvider("camApp", camApp)

    # Connect idSwapped signal from inventoryModel to productSummaryModel
    inventoryController.idSwapped.connect(productSummaryModel.update)

    # set main qml of camera App as rootContext
    engine.rootContext().setContextProperty("camAppPath", CAMAPP_QML)

    # define load main.qml file to start application
    qml_file = Path(__file__).resolve().parent / "src" / "qml" / "main.qml"
    engine.load(qml_file)

    if not engine.rootObjects():
        sys.exit(-1)

    sys.exit(app.exec())
```

Image Processing Application

This Python File implements the logic to recognize arUco markers. class VideoThread inherits from QThread class. So image capture and image processing code is in seperated thread. Processed images are provided to qml by using class videoPlayer which inherits from QQuickImageProvider.

`class src.cameraApplication.cameraProcessing.`**`VideoPlayer`**

`requestImage`(*id*, *size*, *requestedSize*)

This function overrides requestImage from inherited class. :param id: necessary identifier to switch between images. Can be any value. Implemented as boolean value which is toggled everytime when imageChanged is emitted form a JavaScript - function in CameraApplicationMain.qml :param size: :param requestedSize: :return: returns QImage object in RGBA color format

`start`()

Overrides start method of inherited class QQuickImageProvider. It is a Slot and called from QML Button of CameraAppMain.qml :return: this method returns nothing.

`stop`()

Overrides stop method of inherited class QQuickImageProvider. It is a Slot and called from QML Button of CameraAppMain.qml :return: this method returns nothing

`toggleDetection`()

Toggles detection field of VideoThread object. Enables / disables feature detection in VideoThread's run method. It is a Slot and called from QML Button of CameraAppMain.qml :return: This method returns nothing

`updateImage`(*frame*)

Implements connection between VideoThread and VideoPlayer. If VideoThread emits a new image this Slot is called. stores emitted image in self.image and emits image to QQmlEngine :param frame: QImage which is emitted from run-method in VideoThread object. :return: this method returns nothing but emits signal to QQmlEngine

`class src.cameraApplication.cameraProcessing.`**`VideoThread`**(*parent=None*)

`capture`

initializes the first camera device.

`detect`()

enables/disables detection in run-method

`detecting`

enables/disables feature detection

`faceCascade`

initialize haar cascade face detection.. just that there is some image processing

`frameChanged`

Signal which is emitted when a new image is ready for QQuickImageProvider

`quit`()

Necessary Implementation of inherited class to quit existing thread.

`run`()

This Method reads the camera sensor and performs necessary image processing. Converts processed image to Qt's QImage class and emits Signal with QImage `running`

run variable for while loop in run() function

`start()`

Necessary Implementation of inherited class to quit existing thread.

Controllers

class `src.controller.EventlogController.`**`EventlogController`**

class `src.controller.InventoryController.`**`InventoryController`**(*model: InventoryModel = None*, *parent=None*, *eventcontroller: EventlogController = None*, *productlist: ProductListModel = None*)

`changeStorage`(*storage*, *slot*, *cupID*, *productID*)

Takes Data from Override Storage Dialog from Storage.qml Decodes Storage ID ‘L1’ to L’18’ in row / col and checks for ValueErrors. changes InventoryModel Data depending on entries.

`loadStorage`(*storage: str*, *slot: str*)

Takes Data from Override Storage Dialog from Storage.qml Decodes Storage ID ‘L1’ to L’18’ in row / col and checks for ValueErrors. returns productslot, cup ID and productListindex.

class `src.controller.websocketController.`**`WebsocketController`**(*controller: EventlogController*, *parent=None*)

DataModels

class `src.model.InventoryModel.`**`InventoryModel`**(*storageData*, *parent=None*)

`columnCount`(*self*, *parent: PySide6.QtCore.QModelIndex | PySide6.QtCore.QPersistentModelIndex = Invalid(PySide6.QtCore.QModelIndex)*) → int

`data`(*self*, *index: PySide6.QtCore.QModelIndex | PySide6.QtCore.QPersistentModelIndex*, *role: int = Instance(Qt.DisplayRole)*) → Any

`roleNames`(*self*) → Dict[int, PySide6.QtCore.QByteArray]

`rowCount`(*self*, *parent: PySide6.QtCore.QModelIndex | PySide6.QtCore.QPersistentModelIndex = Invalid(PySide6.QtCore.QModelIndex)*) → int

`setData`(*self*, *index: PySide6.QtCore.QModelIndex | PySide6.QtCore.QPersistentModelIndex*, *value: Any*, *role: int = Instance(Qt.EditRole)*) → bool

class `src.model.ProductListModel.`**`ProductListModel`**(*products*, *parent=None*)

`data`(*index*, *role*)

Returns an appropriate value for the requested data. If the view requests an invalid index, an invalid variant is returned. Any valid index that corresponds to a string in the list causes that string to be returned :param index: :param role: :return:

`headerData`(*section*, *orientation*, *role=ItemDataRole.DisplayRole*)

Returns the appropriate header string depending on the orientation of the header and the section. If anything other than the display role is requested, we return an invalid variant.

roleNames(*self*) → Dict[int, PySide6.QtCore.QByteArray]

rowCount(*self*, *parent: PySide6.QtCore.QModelIndex | PySide6.QtCore.QPersistentModelIndex = Invalid(PySide6.QtCore.QModelIndex)*) → int

class src.model.ProductSummaryListModel.**InventoryFilterProxyModel**(*model*, *parent=None*)

filterAcceptsRow(*self*, *source_row: int*, *source_parent: PySide6.QtCore.QModelIndex | PySide6.QtCore.QPersistentModelIndex*) → bool

class src.model.ProductSummaryListModel.**ProductSummaryListModel**(*products*, *parent=None*)

data(*index*, *role*)

Returns an appropriate value for the requested data. If the view requests an invalid index, an invalid variant is returned. Any valid index that corresponds to a string in the list causes that string to be returned. :param index: :param role: :return:

headerData(*section*, *orientation*, *role=ItemDataRole.DisplayRole*)

Returns the appropriate header string depending on the orientation of the header and the section. If anything other than the display role is requested, we return an invalid variant :param section: :param orientation: :param role: :return:

roleNames(*self*) → Dict[int, PySide6.QtCore.QByteArray]

rowCount(*self*, *parent: PySide6.QtCore.QModelIndex | PySide6.QtCore.QPersistentModelIndex = Invalid(PySide6.QtCore.QModelIndex)*) → int

OPC UA Client

Websocket Client

2.2.3 QML Files

QML Files are shown here as literal include. Sphinx doesnt handle qml language by default. Pleas note, that

qml.main.qml

```
import QtQuick
import QtQuick.Window
import QtQuick.Controls 2.15
import QtQuick.Controls.Material 2.15

/*
  Create ApplicationWindow object as base which stores all other elements
  */

ApplicationWindow {

    property bool init: false
    id: mainWindow
    width: Screen.width
    minimumWidth : 480
    height: Screen.height
```

(continues on next page)

(continued from previous page)

```
    minimumHeight: 200
    visible: true
    title: qsTr("Warehouse Management")
    color: "#BDBDBD"

    // custom QML Item which holds logo and welcome String
    HeaderLine {
        id: headerLine
    }

    // custom QML Item which stores all GUI elements to perform manual operations in␣
↪storage
    ManualController {
        id: manCon01
        anchors.left : parent.left
        anchors.top : headerLine.bottom
    }

    // custom QML Item where users can eprform all necessary configurations regarding␣
↪the ABB robottic arm
    ABBRobotConfig {
        id: abbConfig
        anchors.top: manCon01.bottom
        anchors.left: parent.left
    }

    // custom QML Item which shows all possible products (product id and name)
    ProductList {
        id: productlist
        anchors.top: abbConfig.bottom
        anchors.bottom: parent.bottom
    }

    // cutsom QML Item which stores a symbolic picture of the robotic arm and the␣
↪storage of turtle bot and workbench
    ABBRobot {
        id: roboShow
        x: manCon01.width
        anchors.top: headerLine.bottom
    }

    // custom QML Item which shows events happening in the program.
    Eventlogger {
        id: eventlogger
        anchors {
            right: parent.right
            bottom: parent.bottom
            left: roboShow.right
            top: roboShow.bottom
        }
    }
```

(continues on next page)

(continued from previous page)

```
    // custom QML Item which shows a list with product id, name and quantitiy in storage
    Inventory {
        width: roboShow.width
        height: parent.height - roboShow.height - headerLine.height
        anchors{
            top: roboShow.bottom
            left: roboShow.left
            right: roboShow.right
        }
    }

    // custom QML Item which visualizes the rows, columns and pallet content in storage
    Storage {
        id: storage
        anchors {
            left: roboShow.right
            top:  headerLine.bottom
            right: parent.right
            bottom: roboShow.bottom

        }
    }

    // if the GUI has rendered every Item, pass an event to eventlogger
    onAfterRendering: {
        if (!init){
            eventLogController.writeEvent("QML", "Program GUI fully rendered")
            init = true
        }
    }
}
```

qml.ABBRobot.qml

```
import QtQuick 2.15
import QtQuick.Controls 2.15
import QtQuick.Controls.Material 2.3
/*
  This QML Item shows a symbolic imgage of the robotic arm and visualizes product␣
↪information in workbench and turtle bot.
  A Rectangle is basic parent for evry qml item.
  */
Rectangle{
    width: parent.width /3
    height: parent.height /2
    color: "white"
    // The startButton is meant to start the Application.
    Button{
        id: startButton
        text: "START"
```

(continues on next page)

(continued from previous page)

```
        width:  parent.width/4
        height: 60 < parent.height/10 ? 60: parent.height/10
        font.pixelSize: 12
        font.bold: true
        anchors {
            left: parent.left
            top: parent.top
            margins: 5
        }
    }
    // This displays the symbolic picture of the robotic arm.
    Image {
        id: roboImage
        source: "../assets/robot_ref_img.png"
        anchors{
            top: parent.top
            horizontalCenter: parent.horizontalCenter
            margins: 10
        }

        height: 0.5*parent.height
        fillMode: Image.PreserveAspectFit
    }
    // Custom QML Item which shows product id's, cup id'S and if there is a pallet or not.
↪ Enables the user to manually override the storage in turtle bot.
    ProductSlot {
        id : mobileRobot
        width: parent.width/3
        height: parent.height/3 > 210? parent.height/3 : 210
        name: "Mobile Robot"
        anchors{
            bottom: parent.bottom
            left:  parent.left
            margins: 20
        }
    }
    // Custom QML Item which shows product id's, cup id's and if there is a pallet or not.
↪ Enables the user to manually override the storage in workbench.
    ProductSlot {
        id : workBench
        width: parent.width/3
        height: parent.height/3 > 210? parent.height/3 : 210
        name: "Work Bench"
        anchors{
            bottom: parent.bottom
            right:  parent.right
            margins: 20
        }

    }
}
```

qml.ABBRobotArmConfig.qml

```
import QtQuick 2.15
import QtQuick.Controls 2.5
import QtQuick.Controls.Material
import QtQuick.Layouts 1.15
/*
This QML Item enables the user to configure The ModBus settings.
Rectangle is the parent which stores all other elements
*/
Rectangle{
    width: parent.width-20
    height: 90
    // Rowlayouts store each a label and a TextField or Combobox
    RowLayout{
        height: 30
        width: parent.width
        id: row1
        anchors.top: parent.top
        anchors.left: parent.left
        anchors.right: parent.right
        Text {
            id: label_ip
            text: qsTr("IP Adress")
            verticalAlignment: Text.AlignVCenter
            Layout.preferredWidth: parent.width/3
            Layout.preferredHeight: parent.height
        }

        TextField {
            id: abbIp
            placeholderText:  "Enter ModBus IP"
            Layout.preferredWidth: 2*parent.width/3
            Layout.preferredHeight: parent.height
        }
    }

    RowLayout{
        id: row2
        height:30
        width:parent.width
        anchors.top: row1.bottom
        anchors.left: parent.left
        anchors.right: parent.right
        Text {
            id: labelTries
            text: qsTr("Max. Tries")
            verticalAlignment: Text.AlignVCenter
            Layout.preferredWidth: parent.width/3
            Layout.preferredHeight: parent.height
        }

        TextField {
```

(continues on next page)

(continued from previous page)

```
            id: maxTriesField
            placeholderText:  "Enter max. Tries"
            verticalAlignment: Text.AlignVCenter
            Layout.preferredWidth: 2*parent.width/3
            Layout.preferredHeight: parent.height
        }
    }


    RowLayout{
        id: row3
        height: 30
        width: parent.width
        anchors.top: row2.bottom
        anchors.left: parent.left
        anchors.right: parent.right
        Button {
            id: startButton
            text: "Start"
            Layout.preferredWidth: parent.width/2
            Layout.preferredHeight: parent.height
        }
        Button {
            id: modifyButton
            text: "Modify"
            Layout.preferredWidth: parent.width/2
            Layout.preferredHeight: parent.height
        }
    }

}
```

qml.ABBRobotConfig.qml

```
import QtQuick 2.9
import QtQuick.Controls 2.5
import QtQuick.Controls.Material
/*
This QML Item enables the user to configure The ModBus settings.
Rectangle is the parent which stores all other elements
*/
Rectangle {
        id: window
        radius: 10
        color : "white"

        border.color: "#546E7A"
        border.width: 2

        property bool expanded : true
```

(continues on next page)

(continued from previous page)

```
        property int ex_height : 125
        width: 400
        height: 140
        // Image Item contains a small icon picture and a Mouse Area.
        Image {
            id: arrow
            source: "../assets/angle-small-up.png"
            height: 15
            fillMode: Image.PreserveAspectFit
            anchors.left: parent.left
            anchors.top: parent.top
            anchors.margins: 5
            // When the User clicks on the Image image file is changed and visbility is␣
↪changed
            MouseArea {
                anchors.fill: parent
                onClicked: {
                    arrow.source = window.expanded ? "../assets/angle-small-down.png" :
↪"../assets/angle-small-up.png"
                    window.expanded = window.expanded ? false : true
                    window.height = window.expanded? window.ex_height : 25
                    seperator01.visible = seperator01.visible ? false: true
                    seperator02.visible = seperator02.visible ? false : true
                    abbArmConfig.visible = abbArmConfig.visible ? false : true
                }
            }
        }
        // Basic Text Item
        Text {
            id: title
            height: 15
            anchors.left: arrow. right
            anchors.top : parent.top
            anchors.right: parent.right
            anchors.margins : 5
            text: "ABB Robot Arm"
            horizontalAlignment: Text.AlignHCenter
            verticalAlignment: Text.AlignVCenter
        }
        // As QML has no Line Item this Rectangle's height is just 1 pt small
        Rectangle {
            id: seperator01
            visible: true
            width : window.width - 10
            height: 1
            color: "#546E7A"
            anchors.top : arrow.bottom
            anchors.left: window.left
            anchors.margins : 5

            Behavior on visible { PropertyAnimation{} }
        }
```

(continues on next page)

(continued from previous page)

```
        // Custom QML Item which enables User to perform Robot ModBus settings
        ABBRobotArmConfig{
            id: abbArmConfig
            anchors.top : seperator01.bottom
            anchors.left: parent.left
            anchors.margins: 10
            Behavior on visible { PropertyAnimation{ duration: 50; easing.type: Easing.
↪OutCubic}}
        }
        // As QML has no Line Item this Rectangle's height is just 1 pt small
        Rectangle {
            id: seperator02
            visible: true
            width : window.width - 10
            height: 1
            color: "#546E7A"
            anchors.top : abbArmConfig.bottom
            anchors.left: window.left
            anchors.margins : 5

            Behavior on visible { PropertyAnimation{ duration: 50; easing.type: Easing.
↪OutCubic} }
        }
}
```

qml.EditDialog.qml

```
import QtQuick 2.15
import QtQuick.Dialogs
import QtQuick.Controls 2.15
import QtQuick.Controls.Material 2.15
import QtQuick.Layouts 1.3
/*
  This QML File shows a Dialog which enables the user to manually override the storage␣
↪data
  */
Dialog {
    id: editDialog
    title: "Override Storage"
    // ColumnLayout helps to organize Items in vertical order.
    ColumnLayout{
        Layout.fillHeight: true
        Layout.fillWidth: true
        // This Row enables user to allocate the storage location
        Row{
            Text {
                id: location
                text: qsTr("Location: ")
                width: parent.width/2
                height: setLocation.height
```

(continues on next page)

(continued from previous page)

```
                Layout.fillHeight: true
                Layout.fillWidth: true
                verticalAlignment: Text.AlignVCenter
            }
            ComboBox{
                // Comobobox has List of all possible hardcoded storage locations
                id: setLocation
                model: ['L1', 'L2', 'L3', 'L4', 'L5', 'L6', 'L7', 'L8', 'L9', 'L10', 'L11
↪', 'L12', 'L13', 'L14', 'L15', 'L16', 'L17', 'L18']
                Layout.fillHeight: true
                Layout.fillWidth: true
                onCurrentValueChanged: {
                    if(setLocation.currentValue !==''){
                        inventoryController.loadStorage(setLocation.currentValue, setAB.
↪currentValue)
                    }
                }
            }
            Layout.fillHeight: true
            Layout.fillWidth: true
        }
        // This Row enables the user to select either he wants to override the cup in␣
↪front or at the backside.
        Row{
            Text {
                id: slotText
                width: parent.width/2
                text: qsTr("Product a or b: ")
                Layout.fillHeight: true
                Layout.fillWidth: true
                verticalAlignment: Text.AlignVCenter
            }
            ComboBox{
                // a = front, b = back
                id: setAB
                model: ["a","b"]
                Layout.fillHeight: true
                Layout.fillWidth: true
                // load actual storage values if storage location is changed and not␣
↪empty
                onCurrentValueChanged: {
                    inventoryController.loadStorage(setLocation.currentValue, setAB.
↪currentValue)
                }
            }
            Layout.fillHeight: true
            Layout.fillWidth: true
        }
        // This row has a textlabel and textfield which enables the user to override Cup␣
↪ID
        Row{
            Text {
```

(continues on next page)

(continued from previous page)

```
                id: cupText
                width: parent.width/2
                text: qsTr("Set Cup ID: ")
                verticalAlignment: Text.AlignVCenter
            }
            TextField{
                id: setCup
                // limit the cup ID to positive integer between 0 and 9999
                validator: IntValidator{
                    bottom: 0
                    top: 9999
                }

            }
        }
        // This row enables the user to override product id in storage
        Row{
            Text {
                id: setProd
                width: parent.width/2
                text: qsTr("Set Product ID:")
                Layout.fillHeight: true
                Layout.fillWidth: true
                verticalAlignment: Text.AlignVCenter
            }
            ComboBox{
                id:setProduct
                model: productListModel
                textRole: 'id'
                Layout.fillHeight: true
                Layout.fillWidth: true

            }
        }
        // clearbutton enables the user to set values for cup and product which␣
↪implicate that the storage is empty
        DialogButtonBox{
            Button {
                id: clearButton
                text: "Clear"
                onClicked: {
                    console.log("Clear Clicked")
                    setProduct.currentIndex = 0
                    setCup.text = "0"
                }
            }

        }
    }
    // standardbuttons are buttons which perform standard tasks.
    standardButtons: Dialog.Ok | Dialog.Cancel
    // signal which is emitted when Dialog.OK is clicked. It calls changeStorage()␣
```

(continues on next page)

(continued from previous page)

```
↪function of InventoryController
    onAccepted: {
        console.log("location: "+ setLocation.currentText)
        console.log("slot: " +setAB.currentText)
        console.log("cup: " + setCup.text)
        console.log("product: " + setProduct.currentText)
        inventoryController.changeStorage(setLocation.currentText, setAB.currentText,␣
↪setCup.text, setProduct.currentText)
        console.log("Ok clicked")
    }
    onRejected: console.log("Cancel clicked")
    // Connect InventoryController's transmitData Signal to this qml file. If storage is␣
↪set and InventoryController's loadStorage() function is called
    // data will be transmitted by this signal
    Connections{
        target: inventoryController
        function onTransmitData(slot, cup, product){
            setCup.text = cup
            setProduct.editText = product
        }
    }
}
```

qml.Eventlogger.qml

```
import QtQuick 2.15
import QtQuick.Controls 2.15
import QtQuick.Layouts 1.15
import QtQuick.Controls.Material 2.15
/*
  This Qml file implements a basic eventlogger textarea.
  It uses the EventLogController.
  */
Rectangle{
    id: eventWindow
    anchors {
        right: parent.right
        bottom: parent.bottom
        left: parent.right
        top: parent.bottom
    }


    Rectangle {
        id: pane
        radius: 10
        border.color: "#546E7A"
        border.width: 2
        anchors.fill: parent
        property string dateTimeFormat: "yyyy-MM-dd hh:mm:ss"
```

(continues on next page)

(continued from previous page)

```
        Text {
            id: eventLogTitle
            text: qsTr("Event Log:")
            horizontalAlignment: Text.AlignHCenter
            anchors.left: parent.left
            anchors.right: parent.right
            anchors.top: parent.top
            height: 30
            verticalAlignment: Text.AlignVCenter
        }

        ScrollView {
            id: eventScrollView
            width: parent.width
            anchors.top: eventLogTitle.bottom
            anchors.bottom: parent.bottom
            anchors.left: parent.left
            anchors.right: parent.right
            anchors.margins: 10

            TextArea {
                id: eventLogTextArea
                width: parent.width
                height: parent.height
                anchors.fill: parent
                readOnly: true

            }
        }
    }

    Button {
        id:clearButton
        width: 100
        height: 30
        text: "clear"
        anchors {
            top: parent.top
            right: parent.right
            margins: 10
        }
        onClicked: {
            eventLogTextArea.text = ""
        }

    }
    Connections{
        target: eventLogController
        function onNewSignal(message){
            eventLogTextArea.text = message+ "\n"+ eventLogTextArea.text
        }
```

(continues on next page)

(continued from previous page)

```
    }
}
```

qml.HeaderLine.qml

```
import QtQuick 2.15


Rectangle {
        id: headerLine
        width: parent.width
        height: 100
        color : "white"
        anchors.top : parent.top
        anchors.left : parent.left

        Image {
            id: uniKassel
            source: "../assets/logo_unikassel.jpg"
            antialiasing: true
            height: parent.height / 2 -10
            fillMode: Image.PreserveAspectFit
            anchors.left : parent.left
            anchors.top : parent.top
            anchors.margins: 5
        }

        Image {
            id: mrt
            source: "../assets/logo_mrt.png"
            antialiasing: true
            height: parent.height / 2 -10
            fillMode: Image.PreserveAspectFit
            anchors.horizontalCenter : uniKassel.horizontalCenter
            anchors.top : uniKassel.bottom
            anchors.margins: 5
        }

        Text {
            id: titleText
            width: headerLine.width / 2
            height: headerLine.height
            color: "#607d8b"
            text: "Plant Model Factory: Warehouse"
            anchors.left : uniKassel.right
            anchors.top : headerLine.top
            horizontalAlignment: Text.AlignHCenter
            verticalAlignment: Text.AlignVCenter
            minimumPointSize: 9
            minimumPixelSize: 6
```

(continues on next page)

(continued from previous page)

```
            font.pointSize: 20
            textFormat: Text.AutoText
            fontSizeMode: Text.HorizontalFit
            font.kerning: true
            style: Text.Raised
            styleColor: "#607d8b"
        }

        Image {
            id: muPlant
            source: "../assets/logo_uPlant.png"
            antialiasing: true
            height: headerLine.height -10
            fillMode: Image.PreserveAspectFit
            anchors.right : headerLine.right
            anchors.top : headerLine.top
            anchors.margins: 5
        }

    }
```

qml.Inventory.qml

```
import QtQuick 2.15
import QtQuick.Layouts 1.15
import QtQuick.Controls 2.15
import QtQuick.Controls.Material 2.15

//Uses InventoryModel.py DataModel consisting of Produkte.db and StorageData to render␣
↪ListView
//with id, name and quantity

Rectangle{
    Rectangle{
        id: window
        color: "white"
        radius: 10
        border.color: "#546E7A"
        border.width: 2
        anchors{
            top: parent.top
            left: parent.left
            right: parent.right
            bottom: parent.bottom
            margins: 5
        }
        Text {
            id: label
            text: "Inventory"
            font.pixelSize: 12
```

(continues on next page)

(continued from previous page)

```
            font.bold: true
            anchors{
                top: parent.top
                left: parent.left
                margins: 15
            }
        }

        CheckBox {
            id: showEmpty
            checked:  true
            text: "Show empty Entries"
            font.pixelSize: 12
            anchors{
                top: parent.top
                right: parent.right
            }
            onCheckedChanged: {
                    console.log("checkstatechanged!")
                    if (checked) {
                        inventoryFilterModel.setShowZero(true)
                    } else {
                        inventoryFilterModel.setShowZero(false)
                    }
                }
        }

        Rectangle{
            color: "white"
            radius: 5

            border.color: "#546E7A"
            border.width: 1
            anchors {
                top: parent.top
                bottom: parent.bottom
                left: parent.left
                right: parent.right

                topMargin: 40
                bottomMargin: 10
                leftMargin: 10
                rightMargin: 10
            }
            ListView {
                id: inventoryList
                model: inventoryFilterModel
                anchors.fill: parent
                anchors.margins: 10
                clip: true
                spacing: 5
                Layout.fillWidth: true
```

(continues on next page)

(continued from previous page)

```
                delegate:Rectangle{
                    id: rect1
                    width: ListView.view.width
                    height: 50
                    property bool selected: false
                    color: selected ? "#4FC3F7": "white"

                    RowLayout{
                        id: row
                        anchors.fill: parent
                        Text {
                            id: id
                            text: model.id
                            font.pixelSize: 20
                            verticalAlignment: Text.AlignVCenter
                            Layout.fillHeight: true
                            Layout.fillWidth: true
                            Layout.preferredWidth: 50
                        }
                        Text {
                            id: name
                            text: model.name
                            font.pixelSize: 20
                            verticalAlignment: Text.AlignVCenter
                            Layout.fillHeight: true
                            Layout.fillWidth: true
                            Layout.preferredWidth: 400
                        }
                        Text {
                            id: quantity
                            text: model.quantity
                            font.pixelSize: 20
                            verticalAlignment: Text.AlignVCenter
                            Layout.fillHeight: true
                            Layout.fillWidth: true
                            Layout.preferredWidth: 100
                        }
                    }
                    MouseArea {
                        anchors.fill: parent
                        onClicked: {
                            if(!rect1.selected) {
                                inventoryController.selectRow(model.id)
                                rect1.selected= true
                            }
                        }
                    }
                    Connections {
                        target: inventoryController
                        function onRowClicked(message) {
                            if (model.id !== message) {
                                rect1.selected = false
```

(continues on next page)

(continued from previous page)

```
                            }
                            if(parseInt(model.id) === parseInt(message)) {
                                rect1.selected = true
                            }
                        }
                    }
                }
            }
        }
    }
}
```

qml.ManualController.qml

```
import QtQuick 2.9
import QtQuick.Controls 2.5
import QtQuick.Controls.Material


Rectangle {
        id: window
        radius: 10
        color : "white"

        border.color: "#546E7A"
        border.width: 2

        property bool expanded : true
        property int ex_height : 800
        width: 400
        height: 800



        Image {
            id: arrow
            source: "../assets/angle-small-up.png"
            height: 15
            fillMode: Image.PreserveAspectFit
            anchors.left: parent.left
            anchors.top: parent.top
            anchors.margins: 5
            MouseArea {
                anchors.fill: parent
                onClicked: {
                    arrow.source = window.expanded ? "../assets/angle-small-down.png" :
↪"../assets/angle-small-up.png"
                    window.expanded = window.expanded ? false : true
                    window.height = window.expanded? window.ex_height : 25
                    seperator01.visible = seperator01.visible ? false: true
```

(continues on next page)

(continued from previous page)

```
                    modBusConfig.visible = modBusConfig.visible ? false : true
                    seperator02.visible = seperator02.visible ? false : true
                    manOrder.visible = manOrder.visible ? false : true
                    manRFID.visible = manRFID.visible ? false : true
                    manualEvent.visible = manualEvent.visible ? false : true
                }
            }
        }

        Text {
            id: title
            height: 15
            anchors.left: arrow. right
            anchors.top : parent.top
            anchors.right: parent.right
            anchors.margins : 5
            text: "Manual handling"
            horizontalAlignment: Text.AlignHCenter
            verticalAlignment: Text.AlignVCenter
        }

        Rectangle {
            id: seperator01
            visible: true
            width : window.width - 10
            height: 1
            color: "#546E7A"
            anchors.top : arrow.bottom
            anchors.left: window.left
            anchors.margins : 5

            Behavior on visible { PropertyAnimation{} }
        }

        ModBusConfig{
            id: modBusConfig
            anchors.top : seperator01.bottom
            anchors.left: parent.left
            anchors.margins: 5
            Behavior on visible { PropertyAnimation{ duration: 50; easing.type: Easing.
↪OutCubic}}
        }

        Rectangle {
            id: seperator02
            visible: true
            width : window.width - 10
            height: 1
            color: "#546E7A"
            anchors.top : modBusConfig.bottom
            anchors.left: window.left
            anchors.margins : 5
```

(continues on next page)

(continued from previous page)

```
            Behavior on visible { PropertyAnimation{ duration: 50; easing.type: Easing.
↪OutCubic} }
        }

        ManualOrder{
            id: manOrder
            width : window.width - 1
            anchors.top: seperator02.bottom
            anchors.left: window.left
            anchors.right: window.right
            anchors.margins: 5
            Behavior on visible { PropertyAnimation{ duration: 50; easing.type: Easing.
↪OutCubic} }
        }

        ManualRFIDServer{
            id: manRFID
            anchors.top:manOrder.bottom
            anchors.left: parent.left
            anchors.right: parent.right
            anchors.margins: 5
            Behavior on visible { PropertyAnimation{ duration: 50; easing.type: Easing.
↪OutCubic} }
        }

        ManualEventLog {
            id: manualEvent
            anchors.top: manRFID.bottom
            anchors.bottom: parent.bottom
            anchors.left: parent.left
            anchors.right: parent.right
            anchors.margins: 5
            Behavior on visible { PropertyAnimation{ duration: 50; easing.type: Easing.
↪OutCubic} }

        }
}
```

qml.ManualEventLog.qml

```
import QtQuick 2.9
import QtQuick.Controls 2.5
import QtQuick.Controls.Material
import QtQuick.Layouts 1.3

Rectangle {
    id: pane
    width: parent.width
    height: 200
```

(continues on next page)

(continued from previous page)

```
    radius: 10
    border.color: "#546E7A"
    border.width: 2

    Text {
        id: manEventLogTitle
        text: qsTr("Event Log:")
        horizontalAlignment: Text.AlignHCenter
        anchors.left: parent.left
        anchors.right: parent.right
        anchors.top: parent.top
        height: 30
        verticalAlignment: Text.AlignVCenter
    }

    ScrollView {
        id: scrollView
        width: parent.width
        anchors.top: manEventLogTitle.bottom
        anchors.bottom: parent.bottom
        anchors.left: parent.left
        anchors.right: parent.right
        anchors.margins: 10

        TextArea {
            id: manEventLog
            width: parent.width
            height: parent.height
            anchors.fill: parent
            text: "hier steht in wahrheit kein Text"
        }
    }


}
```

qml.ManualOrder.qml

```
import QtQuick 2.9
import QtQuick.Controls 2.5
import QtQuick.Controls.Material
import QtQuick.Layouts 1.3

Rectangle {
    width: parent.width
    height: 330

    TabBar {
        id: tabBar
        width: parent.width
```

(continues on next page)

(continued from previous page)

```
        TabButton {
            text: "Basic"
            onClicked: stackLayout.currentIndex = 0
        }

        TabButton {
            text: "Palette"
            onClicked: stackLayout.currentIndex = 1
        }

        TabButton {
            text: "Cup"
            onClicked: stackLayout.currentIndex = 2
        }
    }

    StackLayout {
        id: stackLayout
        anchors.top: tabBar.bottom
        anchors.left: parent.left
        anchors.right: parent.right
        anchors.bottom: parent.bottom
        anchors.margins: 5
        width: parent.width
        height: parent.height
        Item {
            id: basicItem
            width: parent.width
            height: parent.height
            Rectangle {
                width: parent.width
                height: parent.height

                RowLayout{
                    id: row1
                    height: 40
                    anchors.top: parent.top
                    layer.enabled: false
                    layoutDirection: Qt.LeftToRight
                    anchors.left: parent.left
                    anchors.right: parent.right
                    Text {
                        id: operationLabel
                        height: 30
                        width: row1.width / 2
                        text: qsTr("Operation: ")
                        minimumPixelSize: 6
                        Layout.preferredWidth: parent.width/3
                        Layout.preferredHeight: parent.height
                        verticalAlignment: Text.AlignVCenter
```

(continues on next page)

(continued from previous page)

```
                }
                ComboBox{
                    id: operationComboBox
                    font.family: "Arial"
                    Layout.preferredWidth: 2* parent.width/3
                    Layout.preferredHeight: parent.height
                }
            }

            RowLayout{
                id: row2
                height: 40
                anchors.top: row1.bottom
                anchors.left: parent.left
                anchors.right: parent.right
                Text {
                    id: requestLabel
                    height: 30
                    width: parent.width / 2
                    text: qsTr("Request Type: ")
                    Layout.preferredWidth: parent.width/3
                    Layout.preferredHeight: parent.height
                    verticalAlignment: Text.AlignVCenter
                }
                ComboBox{
                    id: requestComboBox
                    font.family: "Arial"
                    Layout.preferredWidth: 2* parent.width/3
                    Layout.preferredHeight: parent.height
                }
            }

            RowLayout{
                id: row3
                height: 40
                anchors.top: row2.bottom
                anchors.left: parent.left
                anchors.right: parent.right
                Text {
                    id: cupLabel
                    height: 30
                    width: parent.width / 2
                    text: qsTr("Cup ID: ")
                    font.italic: true
                    Layout.preferredWidth: parent.width/3
                    Layout.preferredHeight: parent.height
                    verticalAlignment: Text.AlignVCenter
                }
                TextField{
                    id: cupField
                    width: parent.width / 2
                    text: "0"
```

(continues on next page)

(continued from previous page)

```
                horizontalAlignment: Text.AlignHRight
                font.family: "Arial"
                validator: IntValidator {bottom: 0; top: 100000}
                Layout.preferredWidth: 2* parent.width/3
                Layout.preferredHeight: parent.height
            }
        }

        RowLayout{
            id: row4
            height: 40
            width: parent.width / 2
            anchors.top: row3.bottom
            anchors.left: parent.left
            anchors.right: parent.right
            Text {
                id: productLabel
                height: 30
                width: parent.width / 2
                text: qsTr("Product ID: ")
                font.italic: true
                Layout.preferredWidth: parent.width/3
                Layout.preferredHeight: parent.height
                verticalAlignment: Text.AlignVCenter
            }
            TextField{
                id: productField
                text: "0"
                horizontalAlignment: Text.AlignHRight
                font.family: "Arial"
                validator: IntValidator {bottom: 0; top: 100000}
                Layout.preferredWidth: 2* parent.width/3
                Layout.preferredHeight: parent.height
            }
        }

        RowLayout{
            id: row5
            height: 40
            anchors.top: row4.bottom
            anchors.left: parent.left
            anchors.right: parent.right
            Text {
                id: storageLabel
                height: 30
                width: parent.width /3 -10
                text: qsTr("Storage Position (optional): ")
                font.italic: true
                fontSizeMode: Text.HorizontalFit
                Layout.preferredWidth: parent.width/2
                Layout.preferredHeight: parent.height
                verticalAlignment: Text.AlignVCenter
```

(continues on next page)

(continued from previous page)

```
                    }
                    ComboBox{
                        id: storageCol
                        font.family: "Arial"
                        displayText: "column"
                        Layout.preferredWidth: parent.width/4
                        Layout.preferredHeight: parent.height
                    }
                    ComboBox{
                        id: storageRow
                        font.family: "Arial"
                        displayText: "row"
                        Layout.preferredWidth: parent.width/4
                        Layout.preferredHeight: parent.height
                    }
                }

                RowLayout{
                    id: row6
                    height: 40
                    anchors.top: row5.bottom
                    anchors.left: parent.left
                    anchors.right: parent.right

                    Text {
                        id: cupPositionLabel
                        height: 30
                        width: row1.width / 2
                        text: qsTr("Cup Position (optional): ")
                        Layout.preferredWidth: parent.width/2
                        Layout.preferredHeight: parent.height
                        verticalAlignment: Text.AlignVCenter
                    }
                    ComboBox{
                        id: cupPositionComboBox
                        font.family: "Arial"
                        Layout.preferredWidth: parent.width/2
                        Layout.preferredHeight: parent.height                    }
                }

                RowLayout{
                    id: row7
                    height: 40
                    anchors.top: row6.bottom
                    anchors.left: parent.left
                    anchors.right: parent.right
                    Text {
                        id: noteLabel
                        text: qsTr("Note: Cup ID or Product ID must be set.")
                        fontSizeMode: Text.HorizontalFit
                        Layout.preferredWidth: 2*parent.width/3
```

(continues on next page)

(continued from previous page)

```
                        Layout.preferredHeight: parent.height
                        verticalAlignment: Text.AlignVCenter

                    }
                    Button {
                        id: sendButton
                        text: "send"
                        font.family: "Arial"
                        enabled: false
                        Layout.preferredWidth: parent.width/3
                        Layout.preferredHeight: parent.height
                    }
                }
            }
        }

        Item {
            id: paletteItem
            width: parent.width
            height: parent.height
            Rectangle {
                width: parent.width
                height: parent.height
                RowLayout{
                    id: paletteRow1
                    height: 40
                    anchors.top: parent.top
                    anchors.left: parent.left
                    anchors.right: parent.right
                    Text {
                        id: paletteOperationLabel
                        text: qsTr("Operation: ")
                        minimumPixelSize: 6
                        Layout.preferredWidth: parent.width/3
                        Layout.preferredHeight: parent.height
                        verticalAlignment: Text.AlignVCenter                    }
                    ComboBox{
                        id: paletteOperationComboBox
                        Layout.preferredWidth: 2*parent.width/3
                        Layout.preferredHeight: parent.height
                    }
                }

                RowLayout{
                    id: paletteRow2
                    height: 40
                    anchors.top: paletteRow1.bottom
                    anchors.left: parent.left
                    anchors.right: parent.right
                    Text {
                        id: paletteRequestLabel
                        width: parent.width / 2
```

(continues on next page)

(continued from previous page)

```
                text: qsTr("Request Type: ")
                Layout.preferredWidth: 2*parent.width/3
                Layout.preferredHeight: parent.height
                verticalAlignment: Text.AlignVCenter
            }

        }

        RowLayout{
            id: paletteRow3
            height: 40
            anchors.top: paletteRow2.bottom
            anchors.left: parent.left
            anchors.right: parent.right
            Text {
                id: paletteCupLabel
                width: parent.width / 2
                text: qsTr("Cup ID: ")
                font.italic: true
                Layout.preferredWidth: 2*parent.width/3
                Layout.preferredHeight: parent.height
                verticalAlignment: Text.AlignVCenter                    }

        }

        RowLayout{
            id: paletteRow4
            height: 40
            width: parent.width / 2
            anchors.top: paletteRow3.bottom
            anchors.left: parent.left
            anchors.right: parent.right
            Text {
                id: paletteProductLabel
                text: qsTr("Product ID: ")
                font.italic: true
                Layout.preferredWidth: 2*parent.width/3
                Layout.preferredHeight: parent.height
                verticalAlignment: Text.AlignVCenter                    }

        }

        RowLayout{
            id: paletteRow5
            height: 40
            anchors.top: paletteRow4.bottom
            anchors.left: parent.left
            anchors.right: parent.right
            Text {
                id: paletteStorageLabel
                text: qsTr("Storage Position (optional): ")
                font.italic: true
```

(continues on next page)

(continued from previous page)

```
                    fontSizeMode: Text.HorizontalFit
                    Layout.preferredWidth: parent.width/2
                    Layout.preferredHeight: parent.height
                    verticalAlignment: Text.AlignVCenter
                }
                ComboBox{
                    id: paletteStorageCol
                    Layout.preferredWidth: parent.width/4
                    Layout.preferredHeight: parent.height
                    displayText: "column"
                }
                ComboBox{
                    id: paletteStorageRow
                    Layout.preferredWidth: parent.width/4
                    Layout.preferredHeight: parent.height
                    displayText: "row"
                }
            }

            RowLayout{
                id: paletteRow6
                height: 40
                anchors.top: paletteRow5.bottom
                anchors.left: parent.left
                anchors.right: parent.right

                Text {
                    id: paletteCupPositionLabel
                    text: qsTr("Cup Position (optional): ")
                    Layout.preferredWidth: parent.width/2
                    Layout.preferredHeight: parent.height
                    verticalAlignment: Text.AlignVCenter
                }
                ComboBox{
                    id: paletteCupPositionComboBox
                    Layout.preferredWidth: parent.width/2
                    Layout.preferredHeight: parent.height
                }
            }

            RowLayout{
                id: paletteRow7
                height: 40
                anchors.top: paletteRow6.bottom
                anchors.left: parent.left
                anchors.right: parent.right
                Text {
                    id: paletteNoteLabel
                    text: qsTr("")
                    fontSizeMode: Text.HorizontalFit
                    Layout.preferredWidth: 2*parent.width/3
                    Layout.preferredHeight: parent.height
```

(continues on next page)

(continued from previous page)

```
                        verticalAlignment: Text.AlignVCenter

                    }
                    Button {
                        id: paletteSendButton
                        text: "send"
                        enabled: false
                        Layout.preferredWidth: parent.width/3
                        Layout.preferredHeight: parent.height
                    }
                }
            }
        }

        Item {
            id: cupItem
            width: parent.width
            height: parent.height
            Rectangle {
                width: parent.width
                height: parent.height
                RowLayout{
                    id: cupRow1
                    height: 40
                    anchors.top: parent.top
                    anchors.left: parent.left
                    anchors.right: parent.right
                    Text {
                        id: cupOperationLabel
                        Layout.preferredWidth: parent.width/3
                        Layout.preferredHeight: parent.height
                        verticalAlignment: Text.AlignVCenter
                        text: qsTr("Operation: ")
                        minimumPixelSize: 6
                    }
                    ComboBox{
                        id: cupOperationComboBox
                        Layout.preferredWidth: 2*parent.width/3
                        Layout.preferredHeight: parent.height                    }
                }

                RowLayout{
                    id: cupRow2
                    height: 40
                    anchors.top: cupRow1.bottom
                    anchors.left: parent.left
                    anchors.right: parent.right
                    Text {
                        id: cupRequestLabel
                        Layout.preferredWidth: 2*parent.width/3
                        Layout.preferredHeight: parent.height
                        verticalAlignment: Text.AlignVCenter
```

(continues on next page)

(continued from previous page)

```
                    text: qsTr("Request Type: ")
                }

            }

            RowLayout{
                id: cupRow3
                height: 40
                anchors.top: cupRow2.bottom
                anchors.left: parent.left
                anchors.right: parent.right
                Text {
                    id: cupCupLabel
                    Layout.preferredWidth: 2*parent.width/3
                    Layout.preferredHeight: parent.height
                    verticalAlignment: Text.AlignVCenter
                    text: qsTr("Cup ID: ")
                    font.italic: true
                }

            }

            RowLayout{
                id: cupRow4
                height: 40
                width: parent.width / 2
                anchors.top: cupRow3.bottom
                anchors.left: parent.left
                anchors.right: parent.right
                Text {
                    id: cupProductLabel
                    Layout.preferredWidth: 2*parent.width/3
                    Layout.preferredHeight: parent.height
                    verticalAlignment: Text.AlignVCenter
                    text: qsTr("Product ID: ")
                    font.italic: true
                }

            }

            RowLayout{
                id: cupRow5
                height: 40
                anchors.top: cupRow4.bottom
                anchors.left: parent.left
                anchors.right: parent.right
                Text {
                    id: cupStorageLabel
                    Layout.preferredWidth: parent.width/2
                    Layout.preferredHeight: parent.height
                    verticalAlignment: Text.AlignVCenter
                    text: qsTr("Storage Position (optional): ")
```

(continues on next page)

(continued from previous page)

```
                font.italic: true
                fontSizeMode: Text.HorizontalFit

            }
            ComboBox{
                id: cupStorageCol
                Layout.preferredWidth: parent.width/4
                Layout.preferredHeight: parent.height
                displayText: "column"
            }
            ComboBox{
                id: cupStorageRow
                Layout.preferredWidth: parent.width/4
                Layout.preferredHeight: parent.height
                displayText: "row"
            }
        }

        RowLayout{
            id: cupRow6
            height: 40
            anchors.top: cupRow5.bottom
            anchors.left: parent.left
            anchors.right: parent.right

            Text {
                id: cupCupPositionLabel
                Layout.preferredWidth: parent.width/2
                Layout.preferredHeight: parent.height
                verticalAlignment: Text.AlignVCenter
                text: qsTr("Cup Position (optional): ")
            }
            ComboBox{
                id: cupCupPositionComboBox
                Layout.preferredWidth: parent.width/2
                Layout.preferredHeight: parent.height
            }
        }

        RowLayout{
            id: cupRow7
            height: 40
            anchors.top: cupRow6.bottom
            anchors.left: parent.left
            anchors.right: parent.right
            Text {
                id: cupNoteLabel
                Layout.preferredWidth: 2*parent.width/3
                Layout.preferredHeight: parent.height
                verticalAlignment: Text.AlignVCenter
                text: qsTr("")
                fontSizeMode: Text.HorizontalFit
```

(continues on next page)

(continued from previous page)

```
                    }
                    Button {
                        id: cupSendButton
                        text: "send"
                        enabled: false
                        Layout.preferredWidth: parent.width/3
                        Layout.preferredHeight: parent.height
                    }
                }
            }
        }
    }


}
```

qml.ManualRFIDServer.qml

```
import QtQuick 2.9
import QtQuick.Controls 2.5
import QtQuick.Controls.Material
import QtQuick.Layouts 1.3

Rectangle {
    id: manRfidServer
    width: parent.width
    height: 200
    radius: 5
    border.color: "#546E7A"
    border.width: 2
    Text {
        id: title
        text: "RFID Server"
        width: parent.width
        height: 15
        horizontalAlignment: Text.AlignHCenter
        anchors.left: parent.left
        anchors.top: parent.top    }
        anchors.margins: 5
    RowLayout{
        id: row1
        width:parent.width
        height: 40
        anchors.top: title.bottom
        anchors.left:parent.left
        anchors.right: parent.right
        anchors.margins: 5
```

(continues on next page)

(continued from previous page)

```
        Text {
            id: cupLabel
            text: "Cup ID"
            verticalAlignment: Text.AlignVCenter
            Layout.preferredHeight: parent.height
            Layout.preferredWidth:  parent.width/2
        }

        TextField {
            id: cupIdField
            text: "0"
            horizontalAlignment: Text.AlignHRight
            validator: IntValidator{bottom: 0; top: 100000}
            Layout.preferredHeight: parent.height
            Layout.preferredWidth:  parent.width/2
        }
    }

    RowLayout{
        id: row2
        width:parent.width
        height: 40
        anchors.top: row1.bottom
        anchors.left:parent.left
        anchors.right: parent.right
        anchors.margins: 5


        Text {
            id: cupSizeLabel
            text: "Cup Size (optional)"
            verticalAlignment: Text.AlignVCenter
            Layout.preferredHeight: parent.height
            Layout.preferredWidth:  parent.width/2
        }

        ComboBox {
            id: cupSizeField
            displayText: "Any"
            Layout.preferredHeight: parent.height
            Layout.preferredWidth:  parent.width/2
        }
    }

    RowLayout{
        id: row3
        width:parent.width
        height: 40
        anchors.top: row2.bottom
        anchors.left:parent.left
        anchors.right: parent.right
        anchors.margins: 5
```

(continues on next page)

(continued from previous page)

```
        Text {
            id: produktIDLabel
            text: "Product ID (optional)"
            verticalAlignment: Text.AlignVCenter
            Layout.preferredHeight: parent.height
            Layout.preferredWidth:  parent.width/2
        }

        TextField {
            id: productIDField
            text: "0"
            horizontalAlignment: Text.AlignHRight
            validator: IntValidator{bottom: 0; top: 100000}
            Layout.preferredHeight: parent.height
            Layout.preferredWidth:  parent.width/2
        }
    }
    Button {
        id: sendButton
        anchors.bottom: parent.bottom
        anchors.right: parent.right
        enabled: false
        text: "Send"
        anchors.margins: 5

    }
}
```

qml.ModBusConfig.qml

```
import QtQuick 2.15
import QtQuick.Controls
import QtQuick.Controls.Material


Rectangle{
    width: parent.width -10
    height: 80
    Text {
        id: label_ModBusConfiguration
        width: window.width
        height: 15
        text: qsTr("ModBus Configuration")
        verticalAlignment: Text.AlignVCenter
        horizontalAlignment: Text.AlignHCenter
        font.bold: true
        anchors.left: parent.left
        anchors.top: parent.top
```

(continues on next page)

(continued from previous page)

```
    }

    Text {
        id: label_ip
        text: qsTr("IP Adress")
        width : parent.width /3 -5
        height: 30
        verticalAlignment: Text.AlignVCenter
        anchors.left: parent.left
        anchors.top: label_ModBusConfiguration.bottom
        horizontalAlignment: Text.AlignHCenter
    }

    Text {
        id: label_port
        text: qsTr("Port:")
        width : parent.width /5-5
        height: 30
        verticalAlignment: Text.AlignVCenter
        horizontalAlignment: Text.AlignHCenter
        anchors.left: label_ip.right
        anchors.top: label_ModBusConfiguration.bottom
    }

    Text {
        id: label_connect
        height: 30
        text: qsTr("Disconnected")
        verticalAlignment: Text.AlignVCenter
        font.styleName: "Semibold"
        horizontalAlignment: Text.AlignHCenter
        anchors.right:parent.right
        anchors.rightMargin: 10
        anchors.left: label_port.right
        anchors.leftMargin: 10
        anchors.top: label_ModBusConfiguration.bottom
        font.bold: true
        font.weight: 10
    }
    TextField {
        id: ipField
        width: label_ip.width
        height: label_ip.height
        anchors.left: label_ip.left
        anchors.top: label_ip.bottom
        placeholderText:  "Enter ModBus IP"
        verticalAlignment: Text.AlignVCenter
        color: "black"
    }
    TextField {
        id: portField
        width: label_port.width
```

(continues on next page)

(continued from previous page)

```
        height: label_port.height
        anchors.left: label_port.left
        anchors.top: label_port.bottom
        placeholderText: "Enter Port"
        verticalAlignment: Text.AlignVCenter
        color: "black"
    }

    Button {
        text: "Start"
        width: label_connect.width -5
        height: label_port.height
        anchors.right: label_connect.right
        anchors.top : label_ip.bottom
        checked: false
        checkable: false
        anchors.rightMargin: 10
        onClicked: {
            wsController.startWebSocket(ipField.text, portField.text)
        }
    }
}
```

qml.ProductSlot.qml

```
import QtQuick 2.15
import QtQuick.Controls 2.15
import QtQuick.Controls.Material 2.15
import QtQuick.Layouts 1.15



Rectangle {
    id: productSlot
    width: 200
    height: 400
    radius: 10
    border.width: 2
    border.color: "#546E7A"

    property string name: "ProductSlot"
    property string cupA: "0"
    property string prodA: "0"
    property string nameA: "Kein Becher"
    property string cupB: "0"
    property string prodB: "0"
    property string nameB: "Kein Becher"


    WorkbenchDialog{
```

(continues on next page)

(continued from previous page)

```
        id: editDialog
        title: "Override " + name
    }


    Rectangle{
        id: titleRect
        height: 30
        anchors{
            left: parent.left
            right: parent.right
            top: parent.top
            leftMargin: 20
            rightMargin: 20
            topMargin: 5
        }
        RowLayout{
            anchors.fill: parent

            Text {
                id: titleT
                height: 40
                text: name
                verticalAlignment: Text.AlignVCenter
                Layout.fillHeight: true
                Layout.fillWidth: true
                font.pixelSize: 12
                font.bold: true
            }
            Image {
                id: setImage
                source: "../assets/gear.png"
                fillMode: Image.PreserveAspectFit
                height: title.height
                width: Image.PreserveAspectFit
                Layout.fillHeight: true
                MouseArea{
                    anchors.fill: parent
                    onClicked: {
                        editDialog.source = name
                        editDialog.open()
                    }
                }
            }
        }
    }


    Rectangle {
        id: greySpace
        anchors {
            top: titleRect.bottom
```

(continues on next page)

(continued from previous page)

```
            left: parent.left
            bottom: parent.bottom
            right: parent.right
            margins: 5
        }
        radius: 5
        color: "#607D8B"
        antialiasing: true
        border.width: 1
        border.color: "#263238"
        ColumnLayout{
            anchors.fill: parent
            spacing: 2
            Rectangle{
                id: rect11
                implicitHeight: parent.height/2-10
                implicitWidth: parent.width
                property bool selected: false
                color: rect11.selected ? "#4FC3F7": "white"
                border.color: "#546E7A"
                border.width: 2
                Layout.fillHeight: true
                Layout.fillWidth: true
                radius: 5
                ColumnLayout{
                   anchors.fill: parent
                   Text{
                       text:"Cup ID: "+cupA
                       horizontalAlignment: Text.AlignHCenter
                       verticalAlignment: Text.AlignVCenter
                       Layout.alignment: Qt.AlignHCenter | Qt.AlignVCenter
                       Layout.fillHeight: true
                       Layout.fillWidth: true
                   }
                   Text{
                       text:"Produkt ID: "+prodA
                       horizontalAlignment: Text.AlignHCenter
                       verticalAlignment: Text.AlignVCenter
                       Layout.fillHeight: true
                       Layout.fillWidth: true
                   }
                   Text{
                       text:nameA
                       horizontalAlignment: Text.AlignHCenter
                       verticalAlignment: Text.AlignVCenter
                       Layout.fillHeight: true
                       Layout.fillWidth: true
                   }

                }

            }
```

(continues on next page)

(continued from previous page)

```
            Rectangle{
                id: rect12
                implicitHeight: parent.height/2-10
                implicitWidth: parent.width
                color: rect12.selected ? "#4FC3F7": "white"
                border.color: "#546E7A"
                border.width: 2
                Layout.fillHeight: true
                Layout.fillWidth: true
                radius: 5
                property bool selected: false

                ColumnLayout{
                   anchors.fill: parent
                   Text{
                       width: parent.width
                       text:"Cup ID: "+cupB
                       horizontalAlignment: Text.AlignHCenter
                       verticalAlignment: Text.AlignVCenter
                       Layout.alignment: Qt.AlignHCenter | Qt.AlignVCenter
                       Layout.fillHeight: true
                       Layout.fillWidth: true
                   }
                   Text{
                       width: parent.width
                       text:"Produkt ID: "+prodB
                       horizontalAlignment: Text.AlignHCenter
                       verticalAlignment: Text.AlignVCenter
                       Layout.fillHeight: true
                       Layout.fillWidth: true
                   }
                   Text{
                       width: parent.width
                       text:nameB
                       horizontalAlignment: Text.AlignHCenter
                       verticalAlignment: Text.AlignVCenter
                       Layout.fillHeight: true
                       Layout.fillWidth: true
                   }

                }

            }

        }
    }
    Connections{
        target: inventoryController
        function onRowClicked(message){
            if (parseInt(message)=== parseInt(prodA)){
                rect11.selected = true
            } else{
```

(continues on next page)

(continued from previous page)

```
                rect11.selected = false
            }
            if (parseInt(message)=== parseInt(prodB)){
                rect12.selected = true
            } else{
                rect12.selected = false
            }
        }
    }
}
```

qml.SmallProductSlot.qml

```
import QtQuick 2.15
import QtQuick.Controls 2.15
import QtQuick.Controls.Material 2.15
import QtQuick.Layouts 1.15



Rectangle {
    id: productSlot
    width: 200
    height: 400
    radius: 10
    border.width: 2
    border.color: "#546E7A"

    property string name: "ProductSlot"
    property string cupA: ""
    property string prodA: ""
    property string nameA: ""
    property string cupB: ""
    property string prodB: ""
    property string nameB: ""


    Text {
        id: title
        text: name
        width: parent.width
        height: 20
        verticalAlignment: Text.AlignVCenter
        minimumPixelSize: 6
        horizontalAlignment: Text.AlignHCenter
        anchors{
            top: parent.top
            left: parent.left
            right: parent.right
            leftMargin: 10
```

(continues on next page)

(continued from previous page)

```
            topMargin: 5
        }

        fontSizeMode: Text.HorizontalFit
        font.bold: true
    }

    Rectangle {
        id: greySpace
        height: parent.height
        width: parent.width
        anchors {
            top: title.bottom
            left: parent.left
            bottom: parent.bottom
            right: parent.right
            margins: 5
        }
        radius: 5
        color: "#607D8B"
        border.width: 1
        border.color: "#263238"
        ColumnLayout{
            anchors.fill: parent
            spacing: 2
            // Rectangle holding Product A
            Rectangle{
                id: productSlotA
                implicitHeight: parent.height/2-10
                implicitWidth: parent.width
                color: selected ? "#4FC3F7": "white"
                border.color: "#546E7A"
                border.width: 2
                Layout.fillWidth: true
                Layout.fillHeight: true
                activeFocusOnTab: true
                radius: 5
                property bool selected: false
                ColumnLayout{
                   anchors.fill: parent
                   Text{
                       text:"Cup ID: "+cupA
                       horizontalAlignment: Text.AlignHCenter
                       verticalAlignment: Text.AlignVCenter
                       Layout.fillHeight: true
                       Layout.fillWidth: true
                   }
                   Text{
                       text:nameA
                       horizontalAlignment: Text.AlignHCenter
                       verticalAlignment: Text.AlignVCenter
                       Layout.fillHeight: true
```

(continues on next page)

(continued from previous page)

```
                   Layout.fillWidth: true
               }

            }
            MouseArea {
                anchors.fill: parent
                onClicked: {
                    if (!productSlotA.selected){
                       inventoryController.selectRow(prodA)
                    }

                }

            }
        }
        // Rectangle holds Product B
        Rectangle{
            id: productSlotB
            implicitHeight: parent.height/2-10
            implicitWidth: parent.width
            color: selected ? "#4FC3F7": "white"
            border.color: "#546E7A"
            border.width: 2
            Layout.fillHeight: true
            Layout.fillWidth: true
            radius: 5
            property bool selected: false
            ColumnLayout{
               anchors.fill: parent
               Text{
                   text:"Cup ID: "+cupB
                   horizontalAlignment: Text.AlignHCenter
                   verticalAlignment: Text.AlignVCenter
                   Layout.fillHeight: true
                   Layout.fillWidth: true
               }

               Text{
                   text:nameB
                   horizontalAlignment: Text.AlignHCenter
                   verticalAlignment: Text.AlignVCenter
                   Layout.fillHeight: true
                   Layout.fillWidth: true
               }

            }
            MouseArea {
                anchors.fill: parent
                onClicked: {
                    if (!productSlotB.selected){
                       inventoryController.selectRow(prodB)
                    }
```

(continues on next page)

(continued from previous page)

```
                    }

                }
            }

        }
        //Connect to InventoryController.py's InventoryController and change color of␣
↪selected Product.
        Connections{
            target: inventoryController
            function onRowClicked(message){
                if (prodA === message){
                    productSlotA.selected = true
                }else{
                    productSlotA.selected = false
                }
                if (prodB === message){
                    productSlotB.selected = true
                }else{
                    productSlotB.selected = false
                }
            }
        }
    }
}
```

qml.Storage.qml

```
import QtQuick 2.15
import QtQuick.Layouts 1.15
import QtQuick.Controls 2.15
import QtQuick.Controls.Material 2.15
import QtQuick.Dialogs

Rectangle{
    id: storageRect

    color: "white"
    border.color: "#546E7A"
    border.width: 2
    radius: 10

    EditDialog{
        //Dialog to edit storage data
        id: editDialog
    }

    Loader{
        //Loader to load camera Application QML file
```

(continues on next page)

(continued from previous page)

```
        id: camAppLoader
    }


    Rectangle{
        id: titleRect
        height: 25
        anchors{
            left: parent.left
            right: parent.right
            top: parent.top
            leftMargin: 20
            rightMargin: 20
            topMargin: 5
        }
        RowLayout{
            anchors.fill: parent

            Text {
                id: title
                height: 40
                text: qsTr("Storage")
                verticalAlignment: Text.AlignVCenter
                Layout.fillHeight: true
                Layout.fillWidth: true
                font.pixelSize: 12
                font.bold: true
            }

            Button {


                id: cameraButton
                text: "camApp"
                height: title.height -5


                onClicked: {
                    console.log("Clicked")
                    camAppLoader.source = camAppPath
                }

            }

            Image {
                id: setImage
                source: "../assets/gear.png"
                fillMode: Image.PreserveAspectFit
                height: title.height
                width: Image.PreserveAspectFit
                Layout.fillHeight: true
                MouseArea{
```

(continues on next page)

(continued from previous page)

```
                    anchors.fill: parent
                    onClicked: {
                        editDialog.open()
                    }
                }
            }
        }
    }

    // TableView holds objects of StorageData.db which is read in InventoryModel
    TableView {
        model: inventoryModel
        anchors{
            top: titleRect.bottom
            left: parent.left
            right: parent.right
            bottom: parent.bottom
        }
        anchors.margins: 10
        columnSpacing: 10
        rowSpacing: 5
        clip: true
        delegate: SmallProductSlot{
            cupA: model.a_CupID
            prodA: model.a_ProductID
            nameA: model.a_Name
            cupB: model.b_CupID
            prodB: model.b_ProductID
            nameB: model.b_Name
            name: "L"+ (model.col+1 +model.row*6)
            implicitHeight: 150
            implicitWidth: 150
            Layout.fillWidth: true
            Layout.fillHeight: true
        }
        Layout.fillWidth: true
        Layout.fillHeight: true
    }
}
```

qml.WorkbenchDialog.qml

```
import QtQuick 2.15
import QtQuick.Dialogs
import QtQuick.Controls 2.15
import QtQuick.Controls.Material 2.15
import QtQuick.Layouts 1.3

Dialog {
    id: editDialog
```

(continues on next page)

(continued from previous page)

```
    title: "Override Workbench"
    property string source: ""
    ColumnLayout{
        Layout.fillHeight: true
        Layout.fillWidth: true

        Row{
            Text {
                id: slotText
                width: parent.width/2
                text: qsTr("Product a or b: ")
                Layout.fillHeight: true
                Layout.fillWidth: true
                verticalAlignment: Text.AlignVCenter

            }
            ComboBox{
                id: setAB
                model: ["a","b"]
                Layout.fillHeight: true
                Layout.fillWidth: true
                onCurrentValueChanged: {
                    inventoryController.loadStorage(source, setAB.currentValue)
                }

            }
            Layout.fillHeight: true
            Layout.fillWidth: true
        }
        Row{
            Text {
                id: cupText
                width: parent.width/2
                text: qsTr("Set Cup ID: ")
                verticalAlignment: Text.AlignVCenter
            }
            TextField{
                id: setCup
                validator: IntValidator{
                    bottom: 0
                    top: 9999
                }
            }
        }
        Row{
            Text {
                id: setProd
                width: parent.width/2
                text: qsTr("Set Product ID:")
                Layout.fillHeight: true
                Layout.fillWidth: true
                verticalAlignment: Text.AlignVCenter
```

(continues on next page)

(continued from previous page)

```
            }
            ComboBox{
                id:setProduct
                model: productListModel
                textRole: 'id'
                Layout.fillHeight: true
                Layout.fillWidth: true
            }

        }
        DialogButtonBox{

            Button {
                id: clearButton
                text: "Clear"
                onClicked: {
                    setProduct.currentIndex = 0
                    setCup.text = "0"
                }
            }

        }
    }

    standardButtons: Dialog.Ok | Dialog.Cancel
    onAccepted: {

        inventoryController.changeStorage(source, setAB.currentText, setCup.text,␣
↪setProduct.currentText)
    }
    onRejected: console.log("Cancel clicked")

    Connections{
        target: inventoryController
        function onTransmitData(slot, cup, product){
            setCup.text = cup
            setProduct.editText = product
        }
    }
    onOpened: {
        function setValues(){
            console.log("function called")
            if (name === "Workbench"){
                console.log("recognized Workbench ")
                if(setAB ==='a'){
                    console.log("recognized a ")
                    setCup = workBench.cupA
                    setProduct = workBench.prodA
                } else {
                    console.log("recognized b ")
                    setCup = workBench.cupB
                    setProduct = workBench.prodB
```

(continues on next page)

(continued from previous page)

```
                }
            }
            if (name === "Mobile Robot"){
                if(setAB ==='a'){
                    setCup.text = mobileRobot.cupA
                    setProduct.editText = mobileRobot.prodA
                } else {
                    setCup.text = mobileRobot.cupB
                    setProduct.editText = mobileRobot.prodB
                }

            }
        }
    }
}
```

src.cameraApplication.qml.CameraMain.qml

```
import QtQuick 2.15
import QtQuick.Controls 2.15
import QtQuick.Controls.Material 2.15
import QtQuick.Layouts 1.15

/*
  Create Window object as parent. Mustn't be ApplicationWindow because QMLEngine ans␣
↪QGuiApllication instance already exist
  */
Window {
    id: window
    title: "Warehouse Management - Camera Application"
    color: "white"
    width: 800
    height: 800
    visibility: "Maximized"
    visible: true

    // Draw a Rectangle with colored border and radius as basic screen element
    Rectangle {
        id: baseRect
        visible: true
        color: "white"
        anchors.fill: parent
        anchors.margins: 10
        border.color: "#546E7A"
        border.width: 2
        radius: 10

        // Draw a Rectangle as Container for Image
        Rectangle{
            id: imRim
```

(continues on next page)

(continued from previous page)

```
            width: baseRect.width
            height: baseRect.height * 0.7
            anchors{
                top: baseRect.top
                left: baseRect.left
                right: baseRect.right
            }
            border.color: "#546E7A"
            border.width: 2
            radius: 10
            // Image shows VideoPlayer's captured and processed images
            Image {
                id: camImage
                height: parent.height
                width: height* 1.5
                anchors.centerIn: parent
                source: "image://camApp/img"
                property bool counter: false

            }
            // Connect VideoPlayer's imageChanged Signal with Image item in qml
            Connections{
                target: camApp
                // this function toggles the counter property to constantly alternate␣
↪the id value to get another picture as before.
                // Sets the new incoming picture as Content of Image item camImage
                function onImageChanged(image){
                    console.log("new image emitted")
                    camImage.counter = !camImage.counter
                    camImage.source = "image://camApp/img?id="+camImage.counter
                }
            }

        }
        // Just a describing Text
        Text {
            text: "arUco Camera Application"
            font.pixelSize: 24
            font.bold: true
            anchors.left: baseRect.left
            anchors.right: baseRect.right
            anchors.top: baseRect.top
            anchors.topMargin: 20
            horizontalAlignment: Text.AlignHCenter
        }
        // This Rowlayout stores the buttons to controll the CameraApplication
        RowLayout{
            id: buttonBar
            anchors.left: imRim.left
            anchors.right: imRim.right
            anchors.top: imRim.bottom
            height: 100
```

(continues on next page)

(continued from previous page)

```
            anchors.rightMargin: 100
            anchors.leftMargin: 100
            anchors.topMargin: 10

            Row{
                // Startbutton calls VideoPlayers start - function
                Button{
                    id: startButton
                    text: "Camera Start"
                    width: 200
                    height: 50
                    Layout.fillWidth: true
                    Layout.fillHeight: true
                    onClicked: {
                        camApp.start()
                    }
                }
                // This button calls VideoPlayers toggleDetection - function
                Button{
                    id: toggleButton
                    text: "Detection Start"
                    width: 200
                    height: 50
                    Layout.fillWidth: true
                    Layout.fillHeight: true
                    property bool toggle : false
                    onClicked: {
                        camApp.toggleDetection()
                        toggle = !toggle
                        if(toggle){
                            text = "Detection Stop"
                        } else {
                            text = "Detection Start"
                        }
                    }

                }
                // This Button stops the actual Video feed
                Button{
                    id: stopButton
                    text: "Camera Stop"
                    width: 200
                    height: 50
                    Layout.fillWidth: true
                    Layout.fillHeight: true

                    onClicked: {
                        camApp.stop()
                    }
                }
                Layout.alignment: Qt.AlignHCenter
            }
```

(continues on next page)

(continued from previous page)

```
        }

    }
    // If someone closes the Window the VideoPlayer instance has to destroy VideoThread␣
↪instance.
    onClosing: {
        camAppLoader.source = ""
        camApp.stop()
    }
}
```

2.3 Contributions

- Qt Project

PYTHON MODULE INDEX

INDEX