```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QMessageBox>
#include "manudialog.h"
#include "fejlenum.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    mD = new manuDialog;
    SMConnection = NULL;
    serverConnection = NULL;
    levelAlarm = NULL;
    styrbordVandtanksNiveauAlarm = NULL;
    bagbordVandtanksNiveauAlarm = NULL;

    last_DATA_Update = time(NULL);
    last_GUI_Update = time(NULL);


    //Opsætter connections
    QObject::connect(mD, SIGNAL(manuSet(Level, int, double)), this,
SLOT(manuChangedHandle(Level, int, double)));
    QObject::connect(ui->btn_manu, SIGNAL(clicked()), mD,
SLOT(displayWindow()));

    QObject::connect(ui->btn_close, SIGNAL(clicked()), this,
SLOT(btn_CloseApplication_clicked()));

    //QTimer til opdatering af gui-clocken
    clockTimer = new QTimer(this);
    QObject::connect(clockTimer, SIGNAL(timeout()), this,
SLOT(showTimeSinceLastUpdate()));
    clockTimer->start(1000); //Starter timer

    //Aktivering af automatisk regulering
    emit this->activateAutoClicked();
    setLight(AUTO, true);
    displayManuText(false);

    //Indikation af at programmet er opstartet
    ui->list_aktivitet->insertItem(0,"[PROG] PROGRAM STARTET");
}

MainWindow::~MainWindow()
{
    delete ui;
}


void MainWindow::updateGuiSlot(status temp)
{
    if (SMConnection)
    {
        alarmChangerCheck("HÆLDNING", LEVEL_CRITICAL, temp.level, levelAlarm);
        alarmChangerCheck("STYRBORDSNIVEAU", STYRBORD_CRITICAL,
temp.styrbordNiveau, styrbordVandtanksNiveauAlarm);
        alarmChangerCheck("BAGBORDSNIVEAU", BAGBORD_CRITICAL,
temp.bagbordNiveau, bagbordVandtanksNiveauAlarm);

        //Opdaterer gui'en til de modtagede værdier
        QString vbteStatus[] = {"LUKKET", "IND", "UD", "UKENDT"};

        if (temp.styrbordStatus < 3)
            ui->lblStyrbordStatus->setText(vbteStatus[temp.styrbordStatus]);
        else
```

```
            ui->lblStyrbordStatus->setText(vbteStatus[3]);

        if (temp.bagbordStatus < 3)
            ui->lblBagbordStatus->setText(vbteStatus[temp.bagbordStatus]);
        else
            ui->lblBagbordStatus->setText(vbteStatus[3]);

        ui->lblBagbord->setNum(temp.bagbordNiveau);
        ui->lblStyrbord->setNum(temp.styrbordNiveau);
        levelEnumConverterAndSetter(temp.level);
        //ui->list_aktivitet->insertItem(0, "[DATA] OPDATERING FULDFØRT");
        //ui->list_aktivitet->item(0)->setTextColor(Qt::cyan);
        last_GUI_Update = time(NULL);
    }
}


void MainWindow::on_btn_manu_clicked()
{
    qDebug() << "Clicked";
    mD->show();
}

void MainWindow::on_btn_auto_clicked()
{
    qDebug() << "Auto state: " << autoActivated;
    if (!autoActivated)
    {
        qDebug() << "Auto-IF";
        int reply = QMessageBox::question(this, "Automatisk regulering", "Ønsker
du at aktivere automatisk regulering? Dette vil få systemet væk fra den manuelt
indstillede vinkel.", QMessageBox::Yes | QMessageBox::No );

        if (reply == QMessageBox::Yes)
        {

            emit this->activateAutoClicked();

            if (SMConnection){
                setLight(AUTO, true);
                displayManuText(false);
            }

        }
    }
    else
        QMessageBox::information(this, "Automatisk regulering", "Automatisk
regulering er allerede aktiveret");
}

void MainWindow::setLight(lys ID, bool state)
{
    switch(ID)
    {
    case DATA:
    {
        if (state)
        {
            ui->grondata->show();
            ui->roddata->hide();
        }
        else
        {
            ui->grondata->hide();
            ui->roddata->show();
        }
        break;
    }
    case SM:
```

```
    {
        if (state)
        {
            ui->gronsm->show();
            ui->rodsm->hide();
        }
        else
        {
            ui->gronsm->hide();
            ui->rodsm->show();
        }
        break;
    }
    case AUTO:
        if (state)
        {
            ui->gronauto->show();
            ui->rodauto->hide();

            //Hvis auto er til, er manu fra:
            ui->gronmanu->hide();
            ui->rodmanu->show();
        }
        else
        {
            ui->gronauto->hide();
            ui->rodauto->show();

            //Hvis auto er til, er manu fra:
            ui->gronmanu->show();
            ui->rodmanu->hide();
        }
        break;
    case MANU:
        if (state)
        {
            ui->gronmanu->show();
            ui->rodmanu->hide();

            //Hvis manu er til, er auto fra:
            ui->gronauto->hide();
            ui->rodauto->show();

        }
        else
        {
            ui->gronmanu->hide();
            ui->rodmanu->show();

            //Hvis manu er fra, er auto til:
            ui->gronauto->show();
            ui->rodauto->hide();

        }
        break;
    }
}

void MainWindow::showSMConnectionChange(bool state)
{
    if(state != SMConnection)
    {
        SMConnection = state;
        if(state)
        {
            ui->list_aktivitet->insertItem(0, "[FORB]FORBINDELSE TIL
STYRINGSMODUL OPRETTET");
            ui->list_aktivitet->item(0)->setTextColor(Qt::green);
        }
```

```
        if(!state)
        {
            ui->list_aktivitet->insertItem(0, "[ALARM]FORBINDELSE TIL
STYRINGSMODUL MISTET");
            ui->list_aktivitet->item(0)->setTextColor(Qt::red);
        }
        setLight(SM, SMConnection);
    }
}

void MainWindow::showServerUpdateStatus(bool state)
{
    if(state != serverConnection)
    {
        serverConnection = state;
        if(state)
        {
            ui->list_aktivitet->insertItem(0, "[FORB]FORBINDELSE TIL SERVEREN
OPRETTET");
            ui->list_aktivitet->item(0)->setTextColor(Qt::green);
        }
        else
        {
            ui->list_aktivitet->insertItem(0, "[ALARM] FORBINDELSE TIL SERVER
MISTET");
            ui->list_aktivitet->item(0)->setTextColor(Qt::red);
        }

        setLight(DATA, serverConnection);
    }
    if (state)
        last_DATA_Update = time(NULL);
}

void MainWindow::manuChangedHandle(Level temp, int side, double level)
{
    qDebug() << "#########\n" << "TESTCASE ID: MANUEL HÆLDNINGSREGULERING";
    qDebug() << "START";

    manuelLevel = level;
    qDebug () << "HÆLDNING: " << level;
    if(side == 0)
    {
        manuelSide = "STYRBORD";
        qDebug() << "SIDE: STYRBORD";
    }
    else
    {
        manuelSide = "BAGBORD";
        qDebug() << "SIDE: BAGBORD";
    }

    //Deaktiverer automatisk regulering i gui'en og sætter samtidig manuelt
    autoActivated = false;

    //Viser teksten omhandlende manuel hældningsregulering:
    displayManuText(true);
    emit this->manuChanged(temp);
}

void MainWindow::displayManuText(bool state)
{
    if (state)
    {
        qDebug() << "YES!";
        ui->lblManuGrader->show();
        ui->lblManuText->show();
        ui->lblManuLevel->show();
        ui->lblManuSide->show();
```

```
        setLight(AUTO, false);
        ui->lblManuLevel->setNum(manuelLevel);
        ui->lblManuSide->setText(manuelSide);
    }
    else {
        qDebug() << "NO!";
        ui->lblManuGrader->hide();
        ui->lblManuText->hide();
        ui->lblManuLevel->hide();
        ui->lblManuSide->hide();
        setLight(AUTO, true);
    }
    //Hvis manuel text skal vises er automatisk regulering deaktiveret og
omvendt.
    autoActivated = !state;
}

void MainWindow::showTimeSinceLastUpdate()
{
    int temp = time(NULL);
    ui->lbl_DATA_TimeSinceLastUpdate->setNum(temp-last_DATA_Update);
    ui->lbl_GUI_TimeSinceLastUpdate->setNum(temp-last_GUI_Update);

    if((UPDATE_TIME/1000)<(temp-last_DATA_Update))
        ui->lbl_DATA_TimeSinceLastUpdate->setStyleSheet("QLabel { color : red;
}");
    else
        ui->lbl_DATA_TimeSinceLastUpdate->setStyleSheet("QLabel { color : black;
}");

    if((UPDATE_TIME/1000)<(temp-last_GUI_Update))
        ui->lbl_GUI_TimeSinceLastUpdate->setStyleSheet("QLabel { color : red;
}");
    else
        ui->lbl_GUI_TimeSinceLastUpdate->setStyleSheet("QLabel { color : black;
}");
}

void MainWindow::alarmChangerCheck(QString sentence, int critical_point, int
value, bool &earlier_state)
{
    QString criticalString;
    criticalString.setNum(critical_point);
    bool current_state = (value > critical_point);

    if(earlier_state != current_state) //Check if state changed
    {
        earlier_state = current_state; //Update state to new state
        if(current_state)
        {
            if (sentence == "HÆLDNING")
                ui->list_aktivitet->insertItem(0, sentence + " >= " +
criticalString + " GRADER");
            else
                ui->list_aktivitet->insertItem(0, sentence + " >= " +
criticalString + "%");
            ui->list_aktivitet->item(0)->setTextColor(Qt::red);
        }
        else
        {
            ui->list_aktivitet->insertItem(0, sentence + " UNDER " +
criticalString + "%");
            ui->list_aktivitet->item(0)->setTextColor(Qt::green);
        }
    }
}

void MainWindow::btn_CloseApplication_clicked()
```

```
{
    int reply = QMessageBox::question(this, "Terminering", "Er du sikker på at
du ønsker at lukke BROS? Hvis du lukker BROS vil du ikke længere kunne aflæse
hældning og vandtanksniveauerne gennem dette program.\nSystemet vil:\nLukke
ventilerne\nStoppe enhver regulering\nInformere server om programterminering\n",
QMessageBox::Yes | QMessageBox::No );

    if (reply == QMessageBox::Yes)
    {
        qDebug() << "Emitted: programTerminated";
        emit this->programTerminated();
    }
}

void MainWindow::levelEnumConverterAndSetter(int level)
{
    if(level == NUL)
    {
        ui->lblLevel->setNum(0);
        ui->lblSide->setText("STYRBORD");
    }
    for(int i = SBSTART; i < SBSLUT; i++)
    {
        if( i == level )
        {
            ui->lblLevel->setNum((level-SBSTART)*0.5);
            ui->lblSide->setText("STYRBORD");
            qDebug() << ((level-SBSTART)*0.5);
            qDebug() << "i: " << i << "level: " << level;
            return;
        }
    }
    qDebug() << "Vi hælder til BAGBORD";
    for(int j = BBSTART; j < BBSLUT; j++)
    {
        if( j == level )
        {
            ui->lblLevel->setNum((level-BBSTART)*0.5);
            ui->lblSide->setText("BAGBORD");
            qDebug() << ((level-BBSTART)*0.5);
            qDebug() << "j: " << j << "level: " << level;
            return;
        }
    }
    qDebug() << "Level: " << level;

}

int MainWindow::getSecondsSinceLastGuiUpdate()
{
    return (time(NULL)-last_GUI_Update);
}

void MainWindow::displayCloseFailure()
{
    qDebug() << "Informationsbox";
    QMessageBox::information(this, "Nedlukningsfejl", "Ingen kontakt til
Styringsmodulet. Af sikkerhedsmæssige årsager kan programmet ikke lukkes.");
}
```