LAMPIRAN

- Monitoring.pro

```
# Add files and directories to ship with the application
# by adapting the examples below.
# file1.source = myfile
# dir1.source = mydir
DEPLOYMENTFOLDERS = # file1 dir1
symbian:TARGET.UID3 = 0xE7708E6B
# Allow network access on Symbian
symbian:TARGET.CAPABILITY += NetworkServices
# If your application uses the Qt Mobility libraries, uncomment
# the following lines and add the respective components to the
# MOBILITY variable.
# CONFIG += mobility
# MOBILITY +=
QT += webkit
SOURCES += main.cpp mainwindow.cpp
HEADERS += mainwindow.h
FORMS += mainwindow.ui
# Please do not modify the following two lines. Required for
deployment.
include(deployment.pri)
qtcAddDeployment()
```

- Deployment.pri

```
# checksum 0xfb0c version 0x30001
# This file was generated by an application wizard of Qt Creator.
# The code below handles deployment to Symbian and Maemo, aswell as
copying
# of the application data to shadow build directories on desktop.
# It is recommended not to modify this file, since newer versions of
Qt Creator
# may offer an updated version of it.
defineTest(qtcAddDeployment) {
for(deploymentfolder, DEPLOYMENTFOLDERS) {
    item = item$${deploymentfolder}
    itemsources = $${item}.sources
    $$itemsources = $$eval($${deploymentfolder}.source)
    itempath = $${item}.path
    $$itempath= $$eval($${deploymentfolder}.target)
    export($$itemsources)
    export($$itempath)
    DEPLOYMENT += $$item
}
MAINPROFILEPWD = $$PWD
symbian {
    isEmpty(ICON):exists($${TARGET}.svg):ICON = $${TARGET}.svg
```

```
    isEmpty(TARGET.EPOCHEAPSIZE):TARGET.EPOCHEAPSIZE = 0x20000
0x2000000
} else:win32 {
    copyCommand =
    for(deploymentfolder, DEPLOYMENTFOLDERS) {
        source =
$$MAINPROFILEPWD/$$eval($${deploymentfolder}.source)
        source = $$replace(source, /, \\)
        sourcePathSegments = $$split(source, \\)
        target =
$$OUT_PWD/$$eval($${deploymentfolder}.target)/$$last(sourcePathSegme
nts)
        target = $$replace(target, /, \\)
        !isEqual(source,$$target) {
            !isEmpty(copyCommand):copyCommand += &&
            copyCommand += $(COPY_DIR) \"$$source\" \"$$target\"
        }
    }
    !isEmpty(copyCommand) {
        copyCommand = @echo Copying application data... &&
$$copyCommand
        copydeploymentfolders.commands = $$copyCommand
        first.depends = $(first) copydeploymentfolders
        export(first.depends)
        export(copydeploymentfolders.commands)
        QMAKE_EXTRA_TARGETS += first copydeploymentfolders
    }
} else:unix {
    maemo5 {
        installPrefix = /opt/usr
        desktopfile.path = /usr/share/applications/hildon
    } else {
        installPrefix = /usr/local
        desktopfile.path = /usr/share/applications
        copyCommand =
        for(deploymentfolder, DEPLOYMENTFOLDERS) {
            source =
$$MAINPROFILEPWD/$$eval($${deploymentfolder}.source)
            source = $$replace(source, \\, /)
            macx {
                target =
$$OUT_PWD/$${TARGET}.app/Contents/Resources/$$eval($${deploymentfold
er}.target)
            } else {
                target =
$$OUT_PWD/$$eval($${deploymentfolder}.target)
            }
            target = $$replace(target, \\, /)
            sourcePathSegments = $$split(source, /)
            targetFullPath = $$target/$$last(sourcePathSegments)
            !isEqual(source,$$targetFullPath) {
                !isEmpty(copyCommand):copyCommand += &&
```

```
                copyCommand += $(MKDIR) \"$$target\"
                copyCommand += && $(COPY_DIR) \"$$source\"
\"$$target\"
            }
        }
        !isEmpty(copyCommand) {
            copyCommand = @echo Copying application data... &&
$$copyCommand
            copydeploymentfolders.commands = $$copyCommand
            first.depends = $(first) copydeploymentfolders
            export(first.depends)
            export(copydeploymentfolders.commands)
            QMAKE_EXTRA_TARGETS += first copydeploymentfolders
        }
    }
    for(deploymentfolder, DEPLOYMENTFOLDERS) {
        item = item$${deploymentfolder}
        itemfiles = $${item}.files
        $$itemfiles = $$eval($${deploymentfolder}.source)
        itempath = $${item}.path
        $$itempath =
$${installPrefix}/share/$${TARGET}/$$eval($${deploymentfolder}.targe
t)
        export($$itemfiles)
        export($$itempath)
        INSTALLS += $$item
    }
    icon.files = $${TARGET}.png
    icon.path = /usr/share/icons/hicolor/64x64/apps
    desktopfile.files = $${TARGET}.desktop
    target.path = $${installPrefix}/bin
    export(icon.files)
    export(icon.path)
    export(desktopfile.files)
    export(desktopfile.path)
    export(target.path)
    INSTALLS += desktopfile icon target
}
export (ICON)
export (INSTALLS)
export (DEPLOYMENT)
export (TARGET.EPOCHEAPSIZE)
export (TARGET.CAPABILITY)
export (LIBS)
export (QMAKE_EXTRA_TARGETS)
}
```

- Mainwindow.h

```
// checksum 0x9a77 version 0x30001
/*
  This file was generated by the Mobile Qt Application wizard of Qt
Creator.
  MainWindow is a convenience class containing mobile device
specific code
  such as screen orientation handling.
  It is recommended not to modify this file, since newer versions of
Qt Creator
  may offer an updated version of it.
*/
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QtGui/QMainWindow>
namespace Ui {
    class MainWindow;
}
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    enum ScreenOrientation {
        ScreenOrientationLockPortrait,
        ScreenOrientationLockLandscape,
        ScreenOrientationAuto
    };
    explicit MainWindow(QWidget *parent = 0);
    virtual ~MainWindow();
    void setOrientation(ScreenOrientation orientation);
    void showExpanded();
private slots:
    void on_pushButton_clicked();
private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

- Main.cpp

```
#include "mainwindow.h"
#include <QtGui/QApplication>
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    MainWindow mainWindow;
    mainWindow.setOrientation(MainWindow::ScreenOrientationAuto);
```

```
    mainWindow.showExpanded();
    return app.exec();
}
```

- Mainwindow.cpp

```
// checksum 0xa193 version 0x30001
/*
  This file was generated by the Mobile Qt Application wizard of Qt
Creator.
  MainWindow is a convenience class containing mobile device
specific code
  such as screen orientation handling.
  It is recommended not to modify this file, since newer versions of
Qt Creator
  may offer an updated version of it.
*/
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QtCore/QCoreApplication>
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent), ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}
MainWindow::~MainWindow()
{
    delete ui;
}
void MainWindow::setOrientation(ScreenOrientation orientation)
{
#if defined(Q_OS_SYMBIAN)
    // If the version of Qt on the device is < 4.7.2, that attribute
won't work
    if (orientation != ScreenOrientationAuto) {
        const QStringList v =
QString::fromAscii(qVersion()).split(QLatin1Char('.'));
        if (v.count() == 3 && (v.at(0).toInt() << 16 |
v.at(1).toInt() << 8 | v.at(2).toInt()) < 0x040702) {
            qWarning("Screen orientation locking only supported with
Qt 4.7.2 and above");
            return;
        }
    }
#endif // Q_OS_SYMBIAN
    Qt::WidgetAttribute attribute;
    switch (orientation) {
#if QT_VERSION < 0x040702
```

```
    // Qt < 4.7.2 does not yet have the Qt::WA_*Orientation
attributes
    case ScreenOrientationLockPortrait:
        attribute = static_cast<Qt::WidgetAttribute>(128);
        break;
    case ScreenOrientationLockLandscape:
        attribute = static_cast<Qt::WidgetAttribute>(129);
        break;
    default:
    case ScreenOrientationAuto:
        attribute = static_cast<Qt::WidgetAttribute>(130);
        break;
#else // QT_VERSION < 0x040702
    case ScreenOrientationLockPortrait:
        attribute = Qt::WA_LockPortraitOrientation;
        break;
    case ScreenOrientationLockLandscape:
        attribute = Qt::WA_LockLandscapeOrientation;
        break;
    default:
    case ScreenOrientationAuto:
        attribute = Qt::WA_AutoOrientation;
        break;
#endif // QT_VERSION < 0x040702
    };
    setAttribute(attribute, true);
}
void MainWindow::showExpanded()
{
#ifdef Q_OS_SYMBIAN
    showFullScreen();
#elif defined(Q_WS_MAEMO_5) || defined(Q_WS_MAEMO_6)
    showMaximized();
#else
    show();
#endif
}
void MainWindow::on_pushButton_clicked()
{
    QString url;
    if(ui->radioButton->isChecked())
    {
        url = "http://192.168.173.1/TA/webcam1.html";
        ui->webView->load(QUrl(url));
    }
    else
    {
        url = "http://192.168.173.1/TA/webcam2.html";
        ui->webView->load(QUrl(url));
    }
}
```

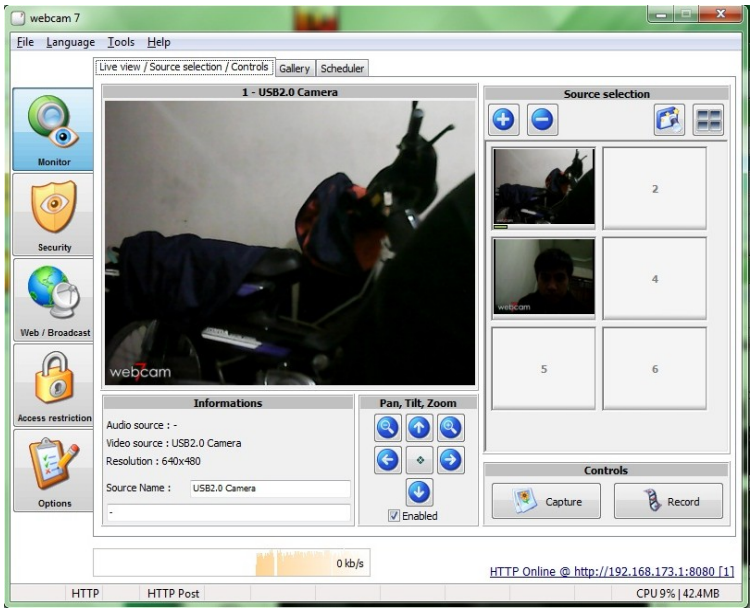
- Webcam1.html

```
<script type="text/javascript">
        var height_array = new Array();
        var width_array = new Array();
        width_array[1] = 300;
        height_array[1] = 200;
</script>
        <img src="http://192.168.173.1:8080/loading.jpg" class="webcam"
        id="webcam1" width="300" height="200" />
<script type="text/javascript">
        currentCamera1= 1;
        document.images.webcam1.onload = LoadImage1;
        function LoadImage1()
        {
            uniq1 = Math.random();
            document.images.webcam1.src = "http://192.168.173.1:8080/cam_" +
            currentCamera1 + ".jpg?uniq="+uniq1;
        }
</script>
```

- Webcam2.html

```
<script type="text/javascript">
        var height_array = new Array();
        var width_array = new Array();
        width_array[1] = 300;
        height_array[1] = 200;
</script>
      <img src="http://192.168.173.1:8080/loading.jpg" class="webcam"
      id="webcam2" width="300" height="200" />
<script type="text/javascript">
      currentCamera2= 2;
      document.images.webcam2.onload = LoadImage2;
      function LoadImage2()
      {
        uniq2 = Math.random();
        document.images.webcam2.src = "http://192.168.173.1:8080/cam_" +
        currentCamera2 + ".jpg?uniq="+uniq2;
      }
</script>
```

Tampilan webcam7



Memantau webcam 1

Memantau webcam2

Tampilan aplikasi monitoring