Debreceni Egyetem

Informatikai Kar

Robotautó világbajnokság

**Témavezető:** Dr. Bátfai Norbert

**Beosztása:** Adjunktus

**Készítette:** Fábián Kristóf – Szabolcs

**Szak megnevezése:** Programtervező Informatikus BSc

Tartalomjegyzék

[1. Bevezetés 4](#__RefHeading___Toc571_1166183222)

[2. Tárgyalás 5](#__RefHeading___Toc573_1166183222)

[2.1 Használt szoftver eszközök 5](#__RefHeading___Toc575_1166183222)

[2.1.1 Qt 5](#__RefHeading___Toc577_1166183222)

[2.1.1.1 qmake 5](#__RefHeading___Toc1100_612737926)

[2.1.1.2 Modulok 7](#__RefHeading___Toc1102_612737926)

[2.1.1.3 Slot-signal mechanizmus 10](#__RefHeading___Toc573_612737926)

[2.1.1.4 Qt Creator 10](#__RefHeading___Toc583_1166183222)

[2.1.2 OpenCV 10](#__RefHeading___Toc585_1166183222)

[2.1.2.1 Telepítés 10](#__RefHeading___Toc587_1166183222)

[2.1.3 BGSLib 10](#__RefHeading___Toc589_1166183222)

[2.2 Előzmények 10](#__RefHeading___Toc591_1166183222)

[2.2.1 <megelőző forgalomszámáló szoftverről> 10](#__RefHeading___Toc593_1166183222)

[2.3 A Robotautó Világbajnokság 10](#__RefHeading___Toc595_1166183222)

[2.3.1 A platform konceptiója 10](#__RefHeading___Toc597_1166183222)

[2.3.2 <> 10](#__RefHeading___Toc1391_1166183222)

[2.4 A szoftver elkészítésének folyamata 10](#__RefHeading___Toc599_1166183222)

[2.4.1 Tervezés 10](#__RefHeading___Toc601_1166183222)

[2.4.2 Fejlesztés 10](#__RefHeading___Toc603_1166183222)

[2.4.2.1 Videó források típusok 10](#__RefHeading___Toc605_1166183222)

[2.4.2.2 Adatbázis kapcsolatok kialakítása, kezelése 10](#__RefHeading___Toc607_1166183222)

[2.4.2.3 Szálkezelés 10](#__RefHeading___Toc609_1166183222)

[2.4.2.4 Videó feldolgozása 11](#__RefHeading___Toc611_1166183222)

[2.4.2.5 Járművek detektálása 11](#__RefHeading___Toc613_1166183222)

[2.4.3 Problémák a szoftver fejlesztése során 11](#__RefHeading___Toc615_1166183222)

[2.4.4 A szoftver tesztelése 11](#__RefHeading___Toc1607_1166183222)

[2.4.5 Eredmények 11](#__RefHeading___Toc1609_1166183222)

[2.5 A szoftver használata 11](#__RefHeading___Toc621_1166183222)

[2.5.1 Fordítása 11](#__RefHeading___Toc1611_1166183222)

[2.5.2 Indítása 11](#__RefHeading___Toc623_1166183222)

[2.5.3 A grafikus felhasználói felület használata 11](#__RefHeading___Toc625_1166183222)

[2.5.3.1 Új forrás kiválasztása 11](#__RefHeading___Toc627_1166183222)

[2.5.3.2 Adatbázisok kezelése 11](#__RefHeading___Toc629_1166183222)

[2.5.3.3 Fájl videó forrás információi 11](#__RefHeading___Toc631_1166183222)

[2.5.4 A parancssoros felhasználói felület használata 11](#__RefHeading___Toc633_1166183222)

[2.5.4.1 Használható parancsok 11](#__RefHeading___Toc635_1166183222)

[3. Összefoglalás 11](#__RefHeading___Toc637_1166183222)

[4. Köszönetnyilvánítás 11](#__RefHeading___Toc639_1166183222)

[5. Irodalomjegyzék 12](#__RefHeading___Toc641_1166183222)

# Bevezetés

# Tárgyalás

## Használt szoftver eszközök

### Qt

A Qt (kiejtése az angol “cute” szóval megegyező) egy számos platformra fordítható keretrendszer, aminek segítségével PC, mobil, illetve beágyazott rendszerekre is fejleszthetünk alkalmazásokat C++ programozási nyelven. A többi keretrendszerhez eltérően, a Qt rendelkezik egy MOC (Meta – Object Compiler) [MOC] nevű kód-generátorral. Ez az generátor feldolgozza a Qt- ban írt forrásfájlokat. Ha valamely osztály deklarációban a MOC megtalálja a Q\_OBJECT makrót akkor egy másik C++ forrásfájlt generál belőlük, ezt meta-objektum kódnak nevezzük. A Qt egyik sajátos funkciója, a slot-signal (magyarul?) a MOC- nak köszönhető. A slot-signal mechanizmusról bővebben a 2.1.1.3- as, “Slot-signal mechanizmus” nevű fejezetben írok. A MOC mellett a Qt még használ egy uic nevű kód-generátort. Az uic egy felhasználói felületet leíró XML fájlt dolgoz fel, és C++ kódot generál belőle, ami a kézzel leprogramozott felhasználói felületnek felel meg.

Mivel számomra feltétel volt, hogy Open Source szoftvereket használjak, így a Qt nagyon jó választásnak tűnt, mivel elérhető szabad szofvterként is, LGPL/GLP licenc alatt [qtlicensing]. Emellett a Qt megvásárolható kereskedelmi licenc alatt is, mely tartalmaz olyan további funkcionalitásokat, mint például virtuális billentyűzet, adatvizualizáció. Egy közösség alapú adatgyűjtés szerint jelenleg 206 könyvtár [inqlude] használja a Qt- t, különböző, viszont legtöbbjük valamely szabad szoftver licenc alatt érhető el.

#### qmake

A qmake egy olyan eszköz, amely leegyszerüsíti számunkra a fordítás menetét eltérő platformokra. Egy projekt (.pro) fájlból generál számunkra egy úgynevezett Makefile- et, vagy akár projekt fájlt is tudunk a segítségével generálni, ha egyelőre csak a forrásfájlokkal rendelkezünk. Egy Makefile a GNU make által használt fájl, amely leírja, hogy a make hogyan fordítsa illetve kapcsolja össze az objektum fájlokat egy futtatható állománnyá.

Vegyük példának az alábbi forrásokat:

krajsz@krajsz-Lenovo-G500s:~/$ ls -lh

total 16K

-rw-rw-r-- 1 krajsz krajsz 172 apr 21 14:52 main.cpp

-rw-rw-r-- 1 krajsz krajsz 219 apr 21 14:52 mainwindow.cpp

-rw-rw-r-- 1 krajsz krajsz 291 apr 21 14:52 mainwindow.h

-rw-rw-r-- 1 krajsz krajsz 630 apr 21 14:52 mainwindow.ui

Ezután a qmake- nek átadva a -project kapcsolót tudunk generáni egy projekt fájlt. Átadhatjuk a forrásfájlokat argumentumként a qmake- nek, viszont, ha nem adunk meg egy fájlt sem, akkor az aktuális könyvtárban található forrásfájlok alapján generálja a projekt fájlt:

krajsz@krajsz-Lenovo-G500s:~/$ qmake -project

krajsz@krajsz-Lenovo-G500s:~/$ ls -l

total 20

-rw-rw-r-- 1 krajsz krajsz 172 apr 21 14:52 main.cpp

-rw-rw-r-- 1 krajsz krajsz 219 apr 21 14:52 mainwindow.cpp

-rw-rw-r-- 1 krajsz krajsz 291 apr 21 14:52 mainwindow.h

-rw-rw-r-- 1 krajsz krajsz 630 apr 21 14:52 mainwindow.ui

-rw-rw-r-- 1 krajsz krajsz 354 apr 21 15:21 testProgram.pro

Láthatjuk, hogy generálva lett egy testProgram.pro nevű állomány, aminek a tartalma a következő:

krajsz@krajsz-Lenovo-G500s:~/$ cat testProgram.pro

TEMPLATE = app

TARGET = testProgram

INCLUDEPATH += .

# Input

HEADERS += mainwindow.h

FORMS += mainwindow.ui

SOURCES += main.cpp mainwindow.cpp

Mivel a qmake nem csak Qt- s projektekhez használható, így, mint jelen esetben is, kiegészítésre szorul a generált projekt fájl. Jelen esetben ahhoz, hogy a megfelelő fájlokat kapcsolja a make, a kövező sorral kell kiegészítsük a projekt fájlunkat:

QT+= widgets

Ezután futtatjuk a qmake- et, átadva neki a generált projekt fájlunkat:

krajsz@krajsz-Lenovo-G500s:~/$ qmake testProgram.pro

Ha nem kapunk semmi féle hibaüzenetet, akkor azt jelenti, sikeresen generálva lett egy Makefile. Láthatjuk, hogy mekkora segítség ez számunkra, mivel egy hét soros projekt fájlból egy 537 soros Makefile- et generált a qmake:

krajsz@krajsz-Lenovo-G500s:~/$ wc -l Makefile

537 Makefile

Ezután a make- et futtatva láthatjuk a fordítás menetét:

krajsz@krajsz-Lenovo-G500s:~/$ make

/usr/lib/x86\_64-linux-gnu/qt5/bin/uic mainwindow.ui -o ui\_mainwindow.h

g++ -c -m64 -pipe -O2 -Wall -W -D\_REENTRANT -fPIC -DQT\_NO\_DEBUG -DQT\_WIDGETS\_LIB -DQT\_GUI\_LIB -DQT\_CORE\_LIB -I. -I. -isystem /usr/include/x86\_64-linux-gnu/qt5 -isystem /usr/include/x86\_64-linux-gnu/qt5/QtWidgets -isystem /usr/include/x86\_64-linux-gnu/qt5/QtGui -isystem /usr/include/x86\_64-linux-gnu/qt5/QtCore -I. -I. -I/usr/lib/x86\_64-linux-gnu/qt5/mkspecs/linux-g++-64 -o main.o main.cpp

…

Ha hiba nélkül lefut a make, azt jelenti, hogy elkészült a futtatható állományunk.

#### Modulok

A Qt több, különálló modulból áll össze, ezek a következőek:

* Qt Core
* Qt GUI
* Qt Multimedia
* Qt Multimedia Widgets
* Qt Network
* Qt QML
* Qt Quick
* Qt Quick Controls
* Qt Quick Dialogs
* Qt Quick Layouts
* Qt SQL
* Qt Test
* Qt Widgets

Ezek közül a dolgozatomban a Qt Core, Qt Multimedia, Qt SQL, illetve Qt Widgets modulokat használom. A Qt Core (“mag”) modulra alapszik az összes többi, ez a modul tartalmazza például a QObject osztályt, amely az összes többi osztály bázis osztálya a Qt- ban. Továbbá, a Core modul tartalmazza a konténer osztályokat is, mint például az általam is használt QVector<T>, illetve QList<T> osztályokat.

A következő módon tudunk egy intVector nevű, száz elemet tartalmazó vektort deklarálni:

QVector<int> intVector(100);

Ekkor, ha megadunk egy értéket második paraméterként a konstruktorban, amely típusa megegyezik a vektorban tárolt elemek típusával, akkor a vektor összes eleme azzal az értékkel fog rendelkezni. Ha nem adtunk meg második paraméterként alapértelmezett értéket a konstruktorban a vektor elemeinek, akkor a típushoz megfelelő alapértelmezett konstruktor meghívásával lesznek példányosítva.

A Qt Multimedia modul ahogy nevéből is következtethető, multimédia tartalom kezeléséhez nyújt eszközöket, mint például videók fájlok kezelése, audio fájlok kezelése, rendszer kamerákhoz hozzáférés. Az általam fejlesztett szoftverben ebből a modulból a QCameraInfo osztályt használom. A QCameraInfo osztály segítségével az alábbi módon lekérhetjük a rendszerünkben található összes kamerához tartozó információt:

QList<QCameraInfo> cameras = QCameraInfo::availableCameras();

A QCameraInfo::availableCameras(QCamera::Position position = QCamera::UnspecifiedPosition) metódus egy statikus metódusa a QCameraInfo osztálynak. A paraméter segítségével lekérhetünk csak bizonyos kamerákról információt, a következő módon meghívva csak az elülső kamerák (például mobil készülékek elülső kamerái) információit fogja visszaadni:

... = QCameraInfo::availableCameras(QCamera::FrontFace);

A Qt SQL modul támogatja SQL adatbázisokkal kapcsolatos műveleteket. Az adatbázishoz való kapcsolódásról az QSqlDatabase felelős. Az alábbi módon tudunk egy adatbáziskapcsolatot létrehozn, majd ellenőrizni, hogy sikeresen meg tudjuk- e nyitnii:

QsqlDatabase db = QSqlDatabase::addDatabase(“QSQLite”);

if (db.open())

{

//sikeresen csatlakoztunk az adatbázishoz

}

else

{

//sikertelen csatlakozás

}

Sikertelen kapcsolódás esetén a QSqlError osztály információval tud szolgálni a kapcsolat létrejöttét meggátló hibáról. Az QSqlDatabase osztály QSqlError lastError() const metódusa visszaadja az adatbázis kapcsolatban felmerülő utolsó hibával kapcsolatos információkat, amit a következő módon Qstring- ként (ahogy a neve is sugallja, egy karakterláncot ábrázoló osztály) meg tudjuk kapni:

const Qstring errorString = db.lastError().text();

Lekérdezések végrehajtása a QSqlQuery osztály segítségével lehetségesek. Lehetőségünk van DML, DDL, illetve adatbázis-specifikus parancsok végrehajtására is.

Egy egyszerű példa a működésére, a lekérdezés feldolgozására:

QsqlQuery query (“Select name from employees”);

while (query.next())

{

qDebug() << quey().value(0).toString();

}

A value(0) függvény visszaadja lekérdezés által visszaadott rekord 0. indexű mezőjét.

Utolsónak említem a Qt Widgets modult, amely a felhasználói felület felépítésére szolgál. Minden egyes grafikus felhasználói felületet alkotó osztály bázisosztálya a QWidget osztály. A QWidget osztály szolgáltat egy képernyőn megjeleníthető felületet, illetve gondoskodik a felhasználói bemenetek feldolgozásáról. Minden osztály ebben a modulban rendelkezik egy alapértelmezett esemény feldolgozó implementációval, viszont mivel ezek virtuális függvények, ezért fölül tudjuk írni őket, és saját implementációnkat használhatjuk.

#### Slot-signal mechanizmus

A slot-signal mechanizmus az egyik legfőbb olyan funkciója a Qt- nak amiben eltér a több keretrendszertől. A legegyszerűbben egy felhasználói felület működésével lehet a slot-signal mechanizmus lényegességét ábrázolni. Például, ha egy egy elemen változtatunk a felhasználói felületünkön, az általában hatással lesz egy másik elemre is. Ez azt jelenti, hogy valamilyen módon kommunikációt kell létrehozzunk a két elem között. A slot-signal mechanizmus épp ezt teszi, kommunikációt tudunk felépíteni bármely két osztály példánya között, amely bázisosztálya a QObject osztály. Ahhoz, hogy használni tudjunk ezt a mechanizmust, még egy feltételnek eleget kell tennünk. Minden osztály deklarációjának a legelején szerepelnie kell a Q\_OBJECT makrónak. Egy slot az lehet egy metódus vagy függvény (rendelkezhet visszatérítési értékkel), egy signal pedig egy “jel”, amit a példányok kibocsátani tudnak, ezek típusa csakis void lehet. Egy signalt akár több slothoz is hozzákapcsolhatjuk, ezáltal akár több példányt is értesíthetünk egy bizonyos példányban kialakult változásokról. Az alábbi ábra szemlélteti ezt:

  
1. ábra: Slot-signal absztrakt módon

Láthatjuk, hogy jelen esetben az Object1 példány kommunikál az Object2 illetve az Object4 példánnyal is.

#### Qt Creator

### OpenCV

#### Telepítés

### BGSLib

## Előzmények

### <megelőző forgalomszámáló szoftverről>

## A Robotautó Világbajnokság

### A platform konceptiója

### <>

## A szoftver elkészítésének folyamata

### Tervezés

### Fejlesztés

#### Videó források típusok

#### Adatbázis kapcsolatok kialakítása, kezelése

#### Szálkezelés

#### Videó feldolgozása

#### Járművek detektálása

### Problémák a szoftver fejlesztése során

### A szoftver tesztelése

### Eredmények

## A szoftver használata

### Fordítása

### Indítása

### A grafikus felhasználói felület használata

#### Új forrás kiválasztása

#### Adatbázisok kezelése

#### Fájl videó forrás információi

### A parancssoros felhasználói felület használata

#### Használható parancsok

# Összefoglalás

# Köszönetnyilvánítás

# Irodalomjegyzék