Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Automatizálási és Alkalmazott Informatikai Tanszék

Számítógép-alapú rendszerek specializáció

**Rendszer és alkalmazástechnika laboratórium 2.**

Jegyzőkönyv

8. mérés

Alkalmazásfejlesztés Qt környezetben

|  |  |
| --- | --- |
| **Mérést végző hallgatók:** | Stummer Tamás |
| **Mérőcsoport:** | Kurzus: CDE, Csoport: [11 |
| **Mérés időpontja:** | 2020.04.01 |
| **Mérés helyszíne:** | BME AAIT, labor: QB 121/125/127 |
| **Mérésvezető:** | Dr. Blázovics László |

# Felhasznált eszközök

|  |  |  |
| --- | --- | --- |
| [eszköz neve] | [eszköz típusa] | [gyári v. leltári szám] |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Mérési feladatok

# feladat

Feladat megoldása

main.c:

// példányosítjuk az osztályt, majd elinditjuk a kommunikációt

SerialConnector mySerialConnector;

bool return\_data = mySerialConnector.connectSerial();

if(return\_data == false)

{

qDebug() << "Connect to the serialport was not successful" << endl;

}

Serialconnector.h:

#ifndef SERIALCONNECTOR\_H

#define SERIALCONNECTOR\_H

#include <QObject>

#include "serialsimulator.h"

class **SerialConnector** : public QObject

{

Q\_OBJECT

public:

explicit **SerialConnector**(QObject \*parent = nullptr);

SerialSimulator \*serialsimulator;

bool **connectSerial**();

bool **disconnectSerial**();

signals:

private slots:

void **serial\_data\_available\_from\_simulator**();

};

#endif // SERIALCONNECTOR\_H

Serialconnector.c:

#include "serialconnector.h"

#include <QDebug>

SerialConnector::**SerialConnector**(QObject \*parent) : QObject(parent)

{

this->serialsimulator = new SerialSimulator();

QObject::connect(this->serialsimulator, SIGNAL(readyRead()), this, SLOT(serial\_data\_available\_from\_simulator()));

}

bool SerialConnector::**connectSerial**()

{

bool wasSuccessful = this->serialsimulator->open(QIODevice::ReadOnly);

return wasSuccessful;

}

bool SerialConnector::**disconnectSerial**()

{

this->serialsimulator->close();

return true;

}

void SerialConnector::**serial\_data\_available\_from\_simulator**()

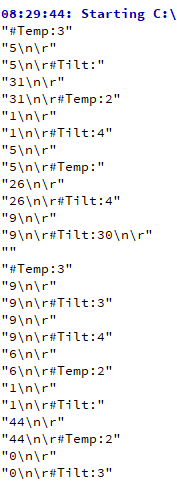
{

QByteArray getted\_serial\_data = this->serialsimulator->readAll();

qDebug() << "New serial data comes from the simulator: " <<getted\_serial\_data << endl;

}

Indítás után az elvártak szerint jüttek a az adatok a szimulátorból:



# feladat

Feladat megoldása

Serialconnector.c:

void SerialConnector::**getMessage**()

{

QString local\_buffer = QString::fromStdString(this->buffer->toStdString()); // konvertáljuk és másoljuk a buffert

QStringList pieces = local\_buffer.split("\n\r"); //elválasztjuk a \n\r mentén

QString needed\_word = pieces.value(0); // elmentjük az első apott parancsot

int stringLength = needed\_word.length();

this->buffer->remove(0,stringLength + 2); // töröljük a bufferből az kivett adatot (+ a \n\r -t is)

this->decodeMessage(needed\_word);

}

void SerialConnector::**decodeMessage**(QString getted\_string)

{

QStringList pieces = getted\_string.split(":");

QVariant number\_value;

if(pieces.length() == 2)

{

number\_value = pieces.value(1);

if(pieces.value(0).compare("#Temp") == 0) {emit tempChanged(number\_value); qDebug() << "temp" << number\_value << endl;}

if(pieces.value(0).compare("#Tilt") == 0) {emit tiltChanged(number\_value);qDebug() << "timt" << number\_value << endl;}

if(pieces.value(0).compare("#Alert") == 0){emit alarmChanged(number\_value);qDebug() << "Alert" << number\_value << endl;}

}

}

stb.

# feladat

Feladat megoldása

main.qml>

Window {

visible: true

width: 640

height: 480

title: *qsTr*("Hello World")

PanelMeter

{

objectName: "TempMeter"

x:10

y:10

width: 100

height: 100

}

PanelMeter

{

objectName: "TiltMeter"

x:120

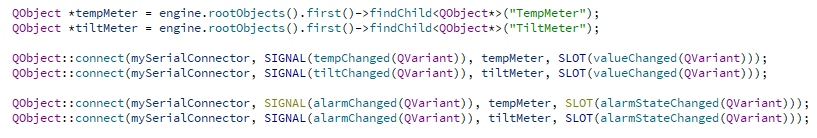
y:10

width: 100

height: 100

}

main.c:



stb.

# Összefoglalás

[A mérés eredményeinek összefoglalása]

# Függelék

[Programlista, stb.]