# Disseny i implementació d'interfícies (2/3 M07UF1)

## Examen del tercer trimestre

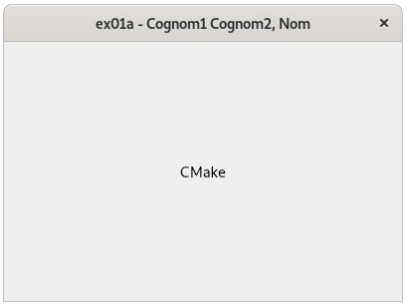
| NOM | | DATA | **13 / 05 / 2024** |
|---|---|---|---|
| ÀREA/MATÈRIA | **2/3 M07UF1** | CURS | **2023 - 2024** |

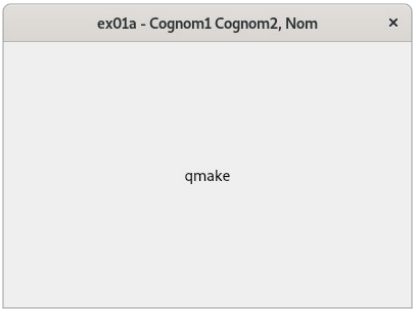
QUALIFICACIÓ

Feu totes les captures de pantalla que cregueu convenients per a documentar les respostes a les preguntes d'aquest examen. Al final de l'examen heu de trametre per correu electrònic els codis i les respostes en format pdf (**tema del correu**: *dam_cognom1cognom2nom_m07uf1* i **nom de l'arxiu**: *dam_cognom1cognom2nom_m07uf1.pdf*). També desareu els codis i el document a la carpeta M07UF1 de la vostra carpeta compartida (i, si cal, amb algun vídeo que demostri el funcionament de l'exercici). Si feu qualsevol millora després de l'examen, haureu d'afegir la versió al final del nom de l'arxiu (per exemple _v2).

**1)*(1 punt qmake / 2 punts CMake)*** Desenvolupeu el projecte **ex01a**. El nom de la classe ha de ser **Ex01Modbus**, essent la classe base de tipus **QWidget**. Decidiu si ho voleu desenvolupar emprant **CMake o qmake** (mireu la puntuació). Amb aquestes característiques:
- La **mida** de la finestra ha de ser de **400 x 260** píxels.
- **Títol** de la finestra: **ex01a – Cognom1 Cognom2, Nom** (Canvieu Cognom1 pel vostre primer cognom, Cognom2 pel vostre segon cognom i Nom pel vostre nom).
- Al mig hi afegiu una **etiqueta** de tipus **QLabel**, **centrada horitzontalment**, amb el nom **etRx** amb el text **CMake** si feu servir CMake **o qmake** si feu servir qmake.
- A sobre i a sota de l'etiqueta hi afegiu espaiadors verticals.
- Agrupeu els espaiadors i l'etiqueta a una distribució vertical.
- Feu que l'agrupació vertical quedi ajustada a la mida de la finestra (dinàmicament, l'etiqueta sempre ha de quedar al centre).
- **Vinculeu** la biblioteca **SerialPort** al projecte. En cas d'haver escollit fer-ho emprant CMake, recordeu:

```
find_package(QT NAMES Qt6 Qt5 REQUIRED COMPONENTS Core Gui SerialPort Widgets)
find_package(Qt${QT_VERSION_MAJOR} REQUIRED COMPONENTS Core Gui SerialPort Widgets)
target_link_libraries(ex01a PRIVATE
    Qt${QT_VERSION_MAJOR}::Core
    Qt${QT_VERSION_MAJOR}::Gui
    Qt${QT_VERSION_MAJOR}::SerialPort
    Qt${QT_VERSION_MAJOR}::Widgets
)
```




**2)*(1 punt)*** Desenvolupeu el projecte **ex01b**, basant-vos en ex01a. Reemplaceu tots els texts ex01a per ex01b a l'arxiu CMakeLists.txt A la classe **Ex01Modbus** afegiu l'**apuntador privat m_serial** que apunta a objectes de la classe **QSerialPort**. Al constructor de la classe **Ex01Modbus** instancieu **m_serial** a ***nullptr***. **Canvieu** el **títol** de la finestra i el contingut de l'etiqueta **etRx** pel de la captura:

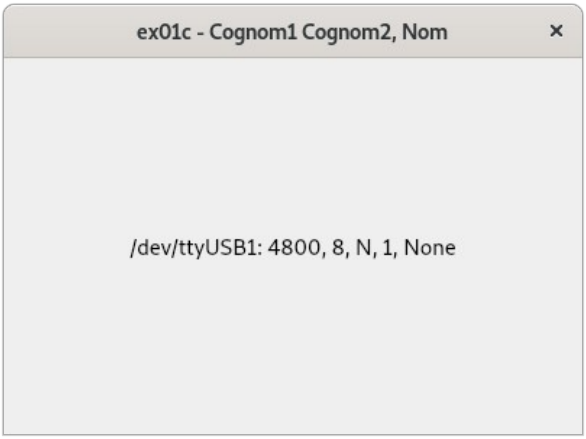
**3)*(2 punts)*** Desenvolupeu el projecte **ex01c**, basant-vos en ex01b. Reemplaceu tots els texts ex01b per ex01c a l'arxiu CMakeLists.txt A la classe **Ex01Modbus** afegiu l'atribut privat **pcSettings** del tipus **parCom**. La configuració de l'arxiu **ex01modbus.h** ha de permetre un constructor com aquest:

```
Ex01Modbus::Ex01Modbus(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::Ex01Modbus)
{
    ui->setupUi(this);
    m_serial = nullptr; // m_serial = new QSerialPort();
    pcSettings = {SERIAL_PORT,
        {QSerialPort::Baud4800,"4800"},
        {QSerialPort::Data8,"8"},
        {QSerialPort::NoParity,"N"},
        {QSerialPort::OneStop,"1"},
        {QSerialPort::NoFlowControl,"None"}
    };
    QString qText = pcSettings.qsPort + ": " +
                    pcSettings.vnBaudRate.qsVar + ", " +
                    pcSettings.vnDataBits.qsVar + ", " +
                    pcSettings.vnParity.qsVar + ", " +
                    pcSettings.vnStopBits.qsVar + ", " +
                    pcSettings.vnFlowControl.qsVar;
    ui->etRx->setText(qText);
}
```

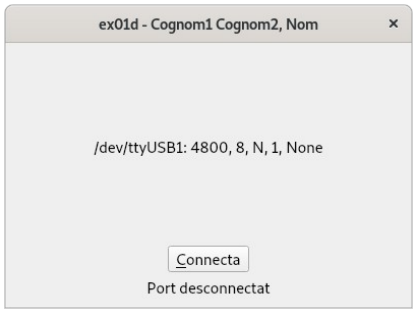
A l'arxiu **ex01modbus.h** afegiu la definició (#define) de **SERIAL_PORT** pel nom del port sèrie a on connectareu el convertidor USB a Modbus RTU. Definiu el nou tipus de variable **parCom** a sobre de la definició de la classe:

```
struct qint32_qstring{
    quint32 nVar;
    QString qsVar;
};
typedef struct qint32_qstring varNom;

struct parametresComunicacio{
    QString qsPort;
    varNom vnBaudRate;
    varNom vnDataBits;
    varNom vnParity;
    varNom vnStopBits;
    varNom vnFlowControl;
};
typedef struct parametresComunicacio parCom;
```

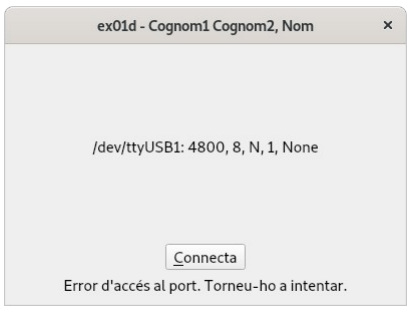
**Canvieu** el **títol** de la finestra i el contingut de l'etiqueta **etRx** pel de l'objecte local **qText** del contructor:



**4)*(3 punts)*** Desenvolupeu el projecte **ex01d**, basant-vos en ex01c. Reemplaceu tots els texts ex01c per ex01d a l'arxiu CMakeLists.txt A la interfície gràfica afegiu una nova etiqueta anomenada **etParamCom** amb el text per defecte **Port desconnectat**. Afegiu el botó **btConnecta** amb el text per defecte **Connecta**. Per a fer la prova us farà falta connectar un element físic al port USB que faci de convertidor a port sèrie.








- Al constructor instancieu l'objecte apuntador **m_serial** a un **nou** espai de la classe **QSerialPort()**
- Desenvolupeu el mètode **void showStatusMessage(QString)** de la classe **Ex01Modbus** que presenta l'argument QString com a text de l'etiqueta **etParamCom**
- Desenvolupeu el mètode **void closeSerialPort()** que tanca el port sèrie, **m_serial→close()**, si **m_serial** és obert. I presenta al text de l'etiqueta **etParamCom** : **Port desconnectat**
- Desenvolupeu el mètode **bool bConnectaPort(parCom)** que retorna **true** si es pot connectar amb els paràmetres passats de tipus **parCom**. I **false** si no s'hi pot connectar.

```
bool Ex01Modbus::bConnectaPort(parCom pc){
    m_serial->setPortName(pc.qsPort);
    m_serial->setBaudRate(pc.vnBaudRate.nVar);
    m_serial->setDataBits((QSerialPort::DataBits)pc.vnDataBits.nVar);
    m_serial->setParity((QSerialPort::Parity)pc.vnParity.nVar);
    m_serial->setStopBits((QSerialPort::StopBits)pc.vnStopBits.nVar);
    m_serial->setFlowControl((QSerialPort::FlowControl)pc.vnFlowControl.nVar);
    return m_serial->open(QIODevice::ReadWrite);
}
```

- Desenvolupeu el **mètode void Ex01Modbus::on_btConnecta_clicked()** que es crida quan es prem el botó **btConnecta** de connexió / desconnexió.

```
void Ex01Modbus::on_btConnecta_clicked()
{
    if(ui->btConnecta->text() == "&Connecta"){
        if(bConnectaPort(pcSettings)){
            ui->btConnecta->setText("&Desconnecta");
            showStatusMessage(tr("Connectat a %1 : %2, %3, %4, %5, %6")
                              .arg(pcSettings.qsPort,
                                   pcSettings.vnBaudRate.qsVar,
                                   pcSettings.vnDataBits.qsVar,
                                   pcSettings.vnParity.qsVar,
                                   pcSettings.vnStopBits.qsVar,
                                   pcSettings.vnFlowControl.qsVar));
        }else{
            showStatusMessage(tr("Error d'accés al port. Torneu-ho a intentar."));
        }
    }else{
        closeSerialPort();
        ui->btConnecta->setText("&Connecta");
    }
}
```

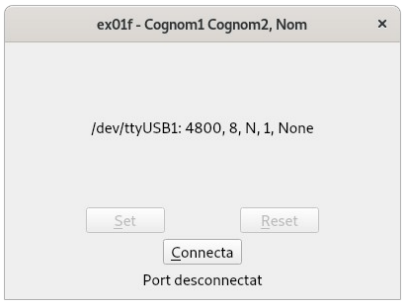
**5)*(1 punt)*** Desenvolupeu el projecte **ex01e**, basant-vos en ex01d. Feu el mètode **QByteArray qbaCrc(QString)** basant-vos en el codi **uint16_t crc16_update(uint16_t crc, uint8_t a)** treballat a classe. Al constructor feu el *testing* del mètode **qbaCRC** que retorna un **QByteArray** amb els dos bytes de CRC calculats a partir del QString passat per valor.

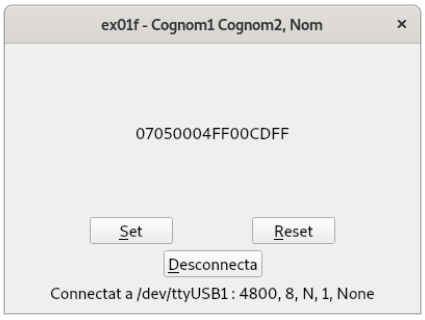
```
uint16_t Ex01Modbus::crc16_update(uint16_t crc, uint8_t a){
    int i;

    crc ^= (uint16_t)a;
    for (i = 0; i < 8; ++i) {
        if (crc & 1)
            crc = (crc >> 1) ^ 0xA001;
        else
            crc = (crc >> 1);
    }

    return crc;
}

QByteArray Ex01Modbus::qbaCrc(QString qs){
    QByteArray ba = QByteArray::fromHex(qs.toUtf8()),baR;
    uint16_t crc = 0xFFFF;

    QByteArray::iterator iteratorByte;
    int count = 0;
    for (iteratorByte = ba.begin(); iteratorByte != ba.end() ; iteratorByte++ ) {
        QByteArray test1Byte(1,0); //Define a 1 Byte fixed length variable
        test1Byte[0]= ba.at(count++); //Assign each byte of the QByteArray ba to this variable
        // qDebug() << test1Byte.toHex().toUpper(); //Print that 1 Byte in hex format
        crc = crc16_update(crc, (uint8_t)test1Byte[0]);
    }
    baR.append((0x00FF & (int)crc));
    baR.append((int)crc >> 8);
    // qDebug() << baR.toHex().toUpper();
    return baR;
}
```

Proposta de *testing* al constructor (cada byte és representat amb dos caràcters):

```
QString qHexString = "07050004FF00";
QByteArray ba = QByteArray::fromHex(qHexString.toUtf8())+qbaCrc(qHexString);
qDebug() << "ba: " << ba.toHex().toUpper();
```

**6)*(2 punts)*** Desenvolupeu el projecte **ex01f**, basant-vos en ex01e. Afegiu dos botons, **btSet** i **btRst**.
- Els botons **btSet** i **btRst** són desactivats fins que hi ha connexió amb el dispositiu físic.
- Quan es prem el botó de **Set** es tramet 0x07050004FF00 més dos bytes de CRC.
- Quan es prem el botó de **Reset** es tramet 0x070500040000 més dos bytes de CRC.
- A l'etiqueta **etRx** es veu la resposta de l'element Modbus RTU.




***Molta sort a totes i tots !!!!***