

```

QT -= gui
QT += network

CONFIG += c++11 console
CONFIG -= app_bundle

TARGET = serviceAfficheur
target.path = /home/pi/c++
INSTALLS += target

HEADERS += seriallib.h afficheur.h service.h
SOURCES += main.cpp seriallib.cpp afficheur.cpp service.cpp

```

```

#include <QCoreApplication>
#include <service.h>

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    Service service;
    Service::connect(&service, SIGNAL(stop()), &a, SLOT(quit()));
    return a.exec();
}

```

```

#ifndef AFFICHEUR_H
#define AFFICHEUR_H

#include "seriallib.h"
#include <string>
using namespace std;

class Afficheur
{
    seriallib usb;
    bool connexion;
public:
    Afficheur() { connexion = usb.Open("/dev/ttyUSB0", 9600) == 1; }
    ~Afficheur() { usb.Close(); }
    bool etat() const { return connexion; }
    void affichePage(const string &message);
private:
    string checksum(string texte);
};

#endif // AFFICHEUR_H

```

```

Fichier  Édition  Affichage  Rechercher  Terminal  Aide
manu@N550JV ~ $ telnet 172.16.2.35 7777
Trying 172.16.2.35...
Connected to 172.16.2.35.
Escape character is '^]'.
Bienvenue !
Message à visualiser (stop pour arrêter le service) : bienvenue!
Connection closed by foreign host.
manu@N550JV ~ $ telnet 172.16.2.35 7777
Trying 172.16.2.35...
Connected to 172.16.2.35.
Escape character is '^]'.
Bienvenue !
Message à visualiser (stop pour arrêter le service) : stop
Connection closed by foreign host.
manu@N550JV ~ $

```

```

#include "afficheur.h"
#include <sstream>

void Afficheur::affichePage(const string &message)
{
    string commande = "<L1><PA><FE><MA><WC><FE><CE>";
    commande += message;
    ostringstream trame;
    trame << "<ID01>" << commande << checksum(commande) << "<E>";
    usb.WriteString(trame.str().c_str());
    usleep(50000);
}

string Afficheur::checksum(string texte)
{
    char calcul = 0;
    for (char octet : texte) calcul^=octet;
    char bas = calcul & 0b00001111;
    char haut = (calcul & 0b11110000) >> 4;
    bas = bas < 10 ? bas+=0x30 : bas+=0x41-10;
    haut = haut < 10 ? haut+=0x30 : haut+=0x41-10;
    return {haut, bas};
}

```

```

#ifndef SERVICE_H
#define SERVICE_H

#include <QObject>
#include <QTcpServer>
#include <afficheur.h>

class Service : public QObject
{
    Q_OBJECT
public:
    explicit Service(QObject *parent = nullptr);
signals:
    void stop();
private slots:
    void nouvelleConnexion();
    void receptionMessage();
    void deconnexion();
private:
    QTcpServer service;
    Afficheur lcd;
};

#endif // SERVICE_H

```

```

#include "service.h"
#include <QTcpSocket>

Service::Service(QObject *parent) : QObject(parent)
{
    if (lcd.etat()) {
        connect(&service, SIGNAL(newConnection()), this, SLOT(nouvelleConnexion()));
        service.listen(QHostAddress::Any, 7777);
        lcd.affichePage("Service actif");
    }
    else stop();
}

void Service::nouvelleConnexion()
{
    QTcpSocket* client = service.nextPendingConnection();
    connect(client, SIGNAL(readyRead()), this, SLOT(receptionMessage()));
    connect(client, SIGNAL(disconnected()), this, SLOT(deconnexion()));
    QTextStream soumettre(client);
    soumettre << "Bienvenue! Saisissez votre message" << endl;
}

void Service::receptionMessage()
{
    QTcpSocket* client = (QTcpSocket*) sender();
    QByteArray message = client->readAll().simplified();
    if (message=="stop") {
        lcd.affichePage("Service inactif");
        stop();
    }
    else lcd.affichePage(message.toStdString());
}

void Service::deconnexion()
{
    QTcpSocket* client = (QTcpSocket*) sender();
    disconnect(client, SIGNAL(readyRead()), this, SLOT(receptionMessage()));
    disconnect(client, SIGNAL(disconnected()), this, SLOT(deconnexion()));
}

```

```

QT += quick network
CONFIG += c++11
TARGET = Afficheur

DEFINES += QT_DEPRECATED_WARNINGS

SOURCES += main.cpp controleur.cpp
HEADERS += controleur.h

RESOURCES += qml.qrc

# Default rules for deployment.
qnx: target.path = /tmp/${TARGET}/bin
else: unix:!android: target.path = /opt/${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

```

```
#include <QGuiApplication>
#include <QQmlApplicationEngine>
#include <controleur.h>
#include <QtQml>

int main(int argc, char *argv[])
{
    QCoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling);
    QGuiApplication app(argc, argv);

    Controleur controleur;
    QQmlApplicationEngine engine;

    engine.rootContext()->setContextProperty("controleur", &controleur);

    engine.load(QUrl(QStringLiteral("qrc:/main.qml")));
    if (engine.rootObjects().isEmpty()) return -1;

    return app.exec();
}
```

```
#ifndef CONTROLEUR_H
#define CONTROLEUR_H

#include <QObject>
#include <QTcpSocket>

class Controleur : public QObject
{
    Q_OBJECT
    Q_PROPERTY(QString message READ getMessage NOTIFY messageChanged)
    Q_PROPERTY(QString adresse MEMBER adresse NOTIFY adresseChanged)

public:
    explicit Controleur(QObject *parent = nullptr) : QObject(parent), requete(&service) {}
    QString getMessage() const { return message; }

signals:
    void messageChanged();
    void adresseChanged();
public slots:
    void soumettre(const QString& texte);
    void reception();
    void connexion();
    void deconnexion();
private:
    QTcpSocket service;
    QString adresse="172.16.2.35", message;
    QTextStream requete;
    int port=7777;
};

#endif // CONTROLEUR_H
```

```
#include "controleur.h"

void Controleur::connexion()
{
    connect(&service, SIGNAL(readyRead()), this, SLOT(reception()));
    service.connectToHost(adresse, port);
}

void Controleur::soumettre(const QString &texte)
{
    requete << texte << endl;
}

void Controleur::reception()
{
    message = service.readAll().simplified();
    messageChanged();
}

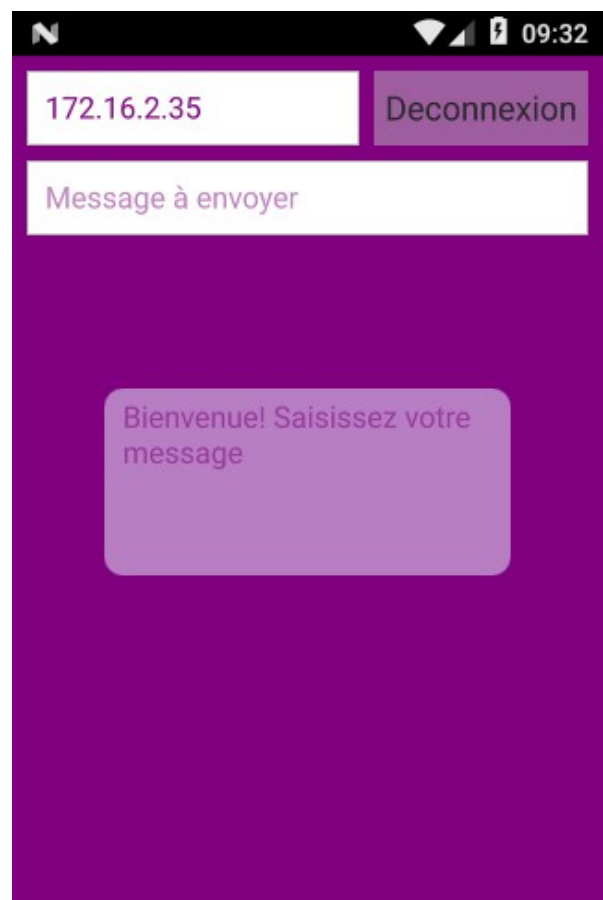
void Controleur::deconnexion()
{
    service.close();
    message = "Connexion interrompue !";
    messageChanged();
}
```

```
import QtQuick 2.9
import QtQuick.Controls 2.4

ApplicationWindow {
    visible: true
    width: 320
    height: 480
    color: "purple"
    title: qsTr("Afficheur AM-03127-LED")

    TextField {
        id: adresse
        anchors {
            top: parent.top
            topMargin: 8
            left: parent.left
            leftMargin: 8
            right: connexion.left
            rightMargin: 8
        }
        color: "purple"
        text: controleur.adresse
        inputMethodHints: Qt.ImhDigitsOnly
    }

    ToolButton {
        id: connexion
        anchors {
            top: parent.top
            topMargin: 8
            right: parent.right
            rightMargin: 8
        }
        text: qsTr("Connexion")
        checkable: true
    }
}
```



```
onCheckedChanged: {
    if (checked) {
        controleur.adresse = adresse.text
        controleur.connexion()
        text = "Deconnexion"
        envoi.enabled = true
    }
    else {
        controleur.deconnexion()
        text = "Connexion"
        envoi.enabled = false
    }
}

TextField {
    id: envoi
    anchors {
        top: adresse.bottom
        topMargin: 8
        left: parent.left
        leftMargin: 8
        right: parent.right
        rightMargin: 8
    }
    placeholderText: qsTr("Message à envoyer")
    color: "purple"
    onAccepted: {
        controleur.soumettre(text)
        selectAll()
    }
    enabled: false
}

TextArea {
    id: réponse
    anchors.centerIn: parent
    width: 220
    height: 100
    font.pixelSize: 16
    background: Rectangle {
        color: "lavender"
        radius: 10
    }
    color: "purple"
    readOnly: true
    wrapMode: "Wrap"
    text: "Connectez-vous"
    OpacityAnimator on opacity {
        id: alerte
        from: 1
        to: 0
        duration: 4000
        easing.type: Easing.InQuart
    }
}

Connections {
    target: controleur
    onMessageChanged: {
        alerte.start()
        réponse.text = controleur.message
    }
}
}
```