Créer une application de contrôle de l'UART avec Qt Creator

Objectifs : Apprendre à réaliser une application Qt mettant en œuvre les communications séries asynchrones et gérer un protocole de communication simple.

1 Présentation du TP

Le tuto propose de réaliser un logiciel de gestion d'un appareil connecté à un PC par UART.



La carte microcontrôleur possède :
Une led de signalisation
Un bouton poussoir

Le logiciel gère ces éléments en fonctions du protocole de communication avec un PC suivant, chaque commande étant suivie d'un tiret d'un message et d'un passage à la ligne :

Périphérique	Sens	Action	Message ASCII	Paramètre ASCII
LED	PC→ uC	Allumage	LD	1
LED	PC→ uC	Extinction	LD	0
Bouton	uC→ PC	Etat bouton	BT	1 ou 0 (relaché/appuyé)

Exemples :
Pour allumer la led le PC transmet : LD1
Si l'on appuie sur le bouton le uC transmet BT0

2 Interfaces séries réelles et simulées

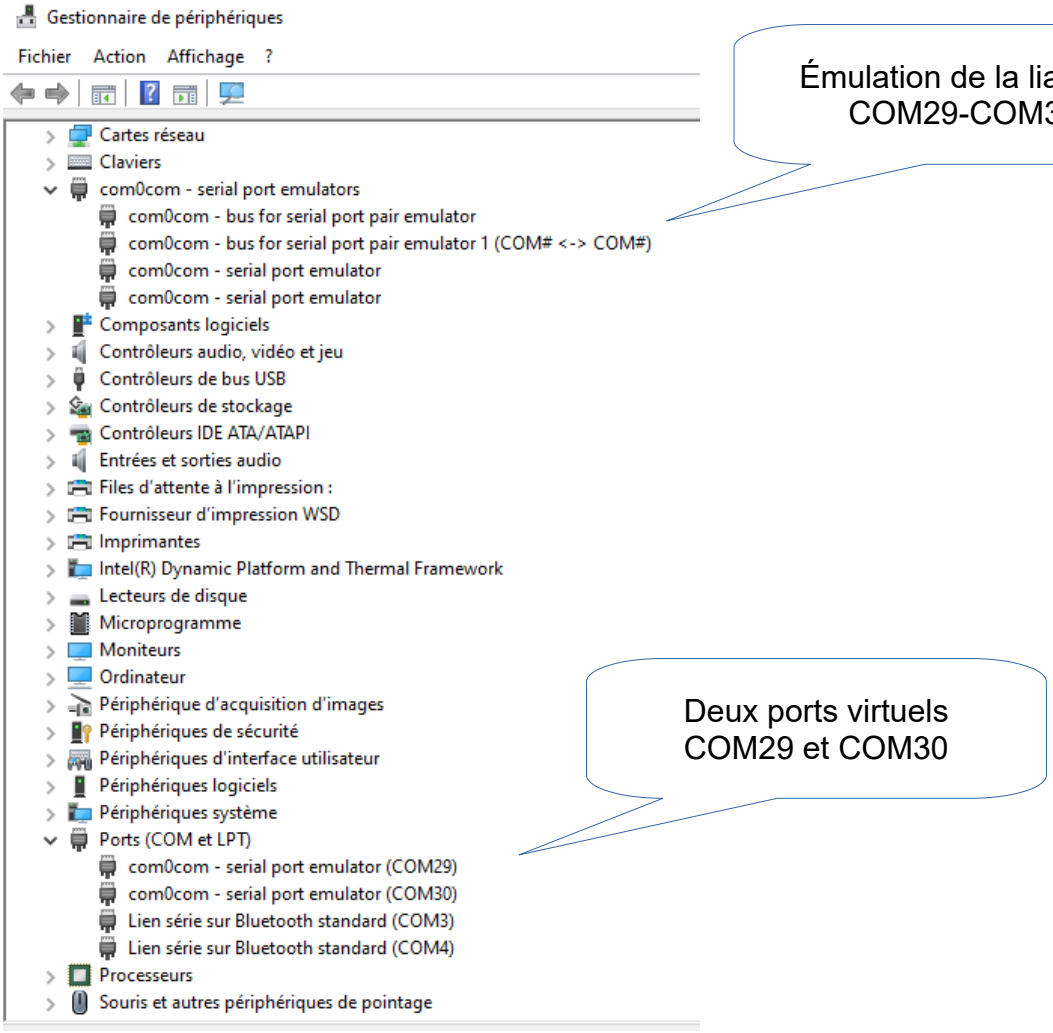
Le format de communications est : 9600,n,8,1

Dans ce TP l'activité du microcontrôleur sera simulée par un simple terminal ASCII (ex Teraterm).

Deux interfaces séries asynchrones (UART) sont nécessaires.
Une pour le PC et une pour le terminal ASCII.

Il est possible d'utiliser deux adaptateurs USB-UART de type FTDI dont les broches RX/TX sont croisées.
Il est cependant plus simple d'utiliser un émulateur de port série com com0com.
Com0com crée deux interfaces UART virtuelles connectées ensemble.

Pour connaître les numéros de ports attribués par com0com, consulter le gestionnaire de périphériques.


Gestionnaire de périphériques
Fichier Action Affichage ?
Cartes réseau
Claviers
com0com - serial port emulators
com0com - bus for serial port pair emulator
com0com - bus for serial port pair emulator 1 (COM# <-> COM#)
com0com - serial port emulator
com0com - serial port emulator
Composants logiciels
Contrôleurs audio, vidéo et jeu
Contrôleurs de bus USB
Contrôleurs de stockage
Contrôleurs IDE ATA/ATAPI
Entrées et sorties audio
Files d'attente à l'impression :
Fournisseur d'impression WSD
Imprimantes
Intel(R) Dynamic Platform and Thermal Framework
Lecteurs de disque
Microprogramme
Moniteurs
Ordinateur
Périphérique d'acquisition d'images
Périphériques de sécurité
Périphériques d'interface utilisateur
Périphériques logiciels
Périphériques système
Ports (COM et LPT)
com0com - serial port emulator (COM29)
com0com - serial port emulator (COM30)
Lien série sur Bluetooth standard (COM3)
Lien série sur Bluetooth standard (COM4)
Processeurs
Souris et autres périphériques de pointage
Émulation de la liaison COM29-COM30
Deux ports virtuels COM29 et COM30

3 Tuto

On souhaite réaliser avec Qt un logiciel de gestion des communications.
La LEDs sont commandées par un bouton
L'état du bouton est affiché avec image

Lancer et configurer Qt Creator

Nouveau projet → Application → Qt Widgets Application
Nommer le projet : democomucpc (pour microcontroleur-PC) et choisir l'emplacement du projet
Nommer la classe du projet democomucpc
Valider la création du projet

Afinc de pouvoir utiliser la bibliothèque de gestion des ports séries éditer ucpc.pro et ajouter la ligne
QT += serialport

```
QT       += core gui
QT       += serialport

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++11
```

Bibliothèque QtSerialPort

Ajouter le bibliothèque QtSerialPort à democomucpc.h et déclarer dans la zone private un pointeur de type QserialPort.

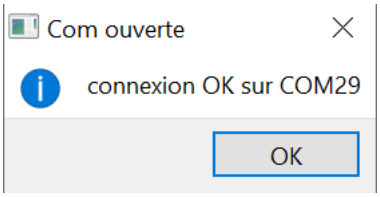
```
#include <QtSerialPort/QserialPort>

private:
    Ui::interfaceuart *ui;
    QSerialPort *portserie;
```

Compléter comme suit le fichier democomucpc.cpp.
Remplacer COM29 par un numéro de port actif sur votre ordinateur.
Prenez soin d'expliquer chaque ligne de code dans le compte-rendu.

```
#include "democomucpc.h"
#include "ui_democomucpc.h"
#include <QMessageBox>
#include <QtSerialPort/QSerialPort>

// decommenter pour version Windows
#define nomport "COM29"
// decommenter pour version Linux
//#define nomport "/dev/ttyUSB0"

democomucpc::democomucpc(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::democomucpc)
{
    ui->setupUi(this);
    portserie = new QSerialPort(this);
```

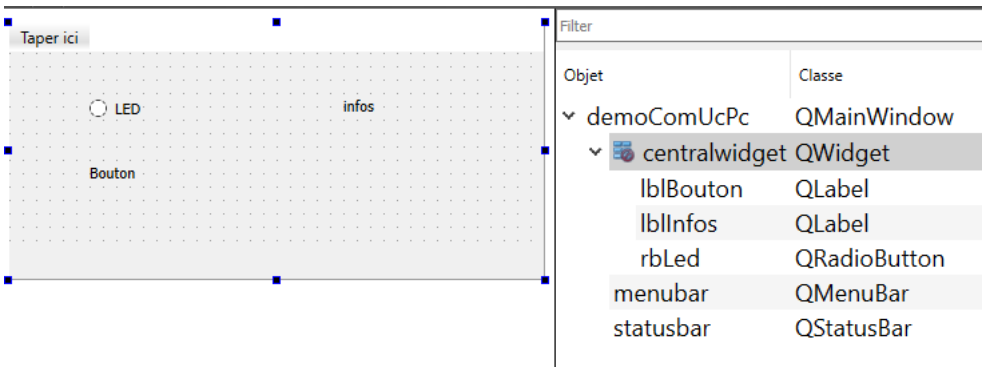
```
    portserie->setPortName(nomport);
    portserie->setBaudRate(QSerialPort::Baud9600);
    portserie->setDataBits(QSerialPort::Data8);
    portserie->setParity(QSerialPort::NoParity);
    portserie->setStopBits(QSerialPort::OneStop);
    portserie->setFlowControl(QSerialPort::NoFlowControl);
    if (portserie->open(QIODevice::ReadWrite))
    {
        QMessageBox::information(this, "Com ouverte","connexion OK sur "+QString(nomport));
        portserie->write("Communications etablies\n");
    }
    else
    {
        // probleme d'ouverture du port serie
        QMessageBox::critical(this, "Erreur sur port "+QString(nomport), portserie->errorString());
        exit(1);
    }
}

demoComUcPc::~demoComUcPc()
{
    if (portserie->isOpen()) portserie->close();
    delete ui;
}
```

Compiler, si le port série existe la fenêtre suivante doit s'afficher.



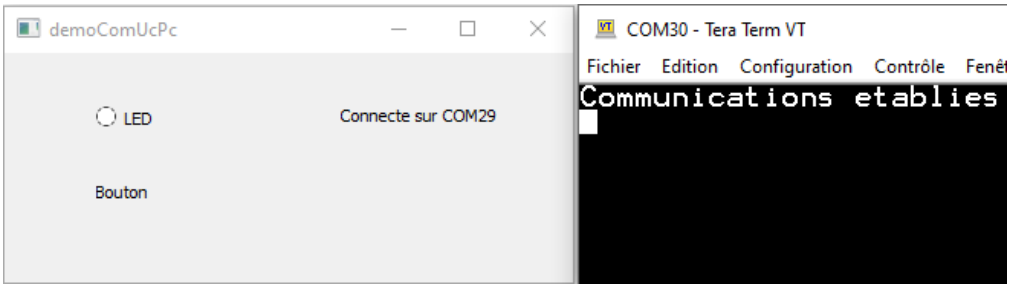
Compléter le programme

Créer un … comme suit, et renommer les widgets RadioButton et Label



Compléter l'initialisation des communications avec un message d'information

```
 if (portserie->open(QIODevice::ReadWrite))
    {
        QMessageBox::information(this, "Com ouverte","connexion OK sur "+QString(nomport));
        portserie->write("Communications etablies\n");
        ui->lblInfos->setText("Connecte sur "+QString(nomport));
    }
```



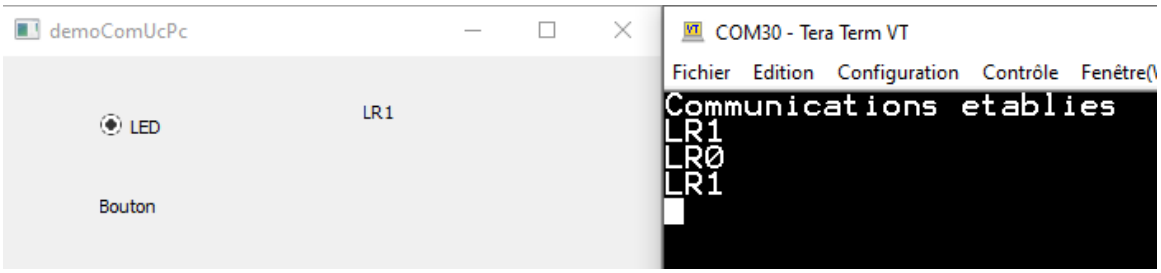
Configurer l'action du radio button

Faire un clic-droit sur rbLed, puis « Allez au slot » puis « clicked() »

Compléter la méthode

```
void demoComUcPc::on_rbLed_clicked()
{
    QString chargeUtile;
    if (ui->rbLed->isChecked()==true) chargeUtile="LD1\n\r";
    else chargeUtile="LD0\n\r";
    portserie->write(chargeUtile.toLocal8Bit());
    ui->lblInfos->setText(chargeUtile);
}
```

Compiler lors du clic sur le radio button, le terminal affiche



Configurer l'action sur lblBouton

lblBouton affiche l'état du bouton en fonction de la commande reçu sur le port série

Compléter l'inialisation des communications par un slot sur le signal readyRead()

```
if (portserie->open(QIODevice::ReadWrite))
    {
        QMessageBox::information(this, "Com ouverte","connexion OK sur "+QString(nomport));
        // si un caractere est arrive on l'ecrit
        connect(portserie, SIGNAL(readyRead()), this, SLOT(litData()));
        portserie->write("Communications etablies\n");
        ui->lblInfos->setText("Connecte sur "+QString(nomport));
    }
```

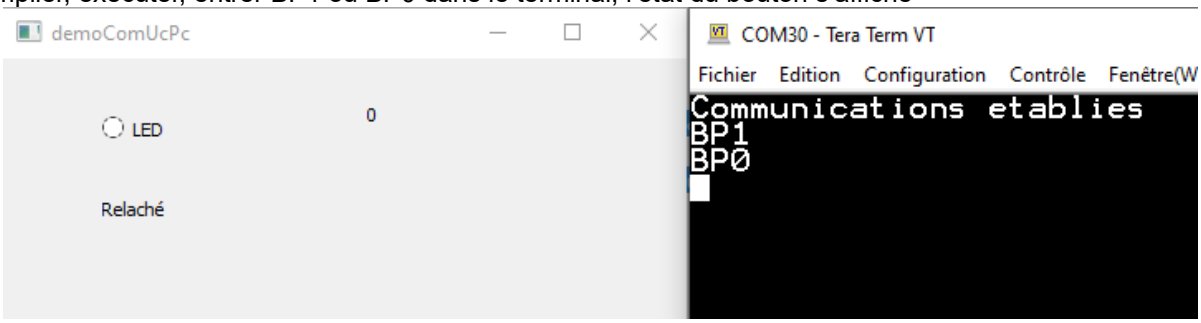
Ajouter le prototype de la méthode litData() dans democomucpc.h

```
private slots:
    void on_rbLed_clicked();
    void litData();
```

Ajouter la méthode litData(), qui récupère les caractères ASCII dans un Qstring jusqu'à trouver un passage à la ligne.

```
void demoComUcPc::litData()
{
   // on recupere tous les caracteres dans le tampon
   static QString data;
   data+=portserie->readAll();    // !!! a verifier
   // on affiche dans le QLabel
   if (data.endsWith("\r")==true || data.endsWith("\n")==true)
   {
   // decodage du message
   QString chargeUtile=data.mid(2); //recupere le payload, les deux premiers caractères sont le type de
message
   if (data.startsWith("BP")==true) {
                if(chargeUtile[0]=='1') ui->lblBouton->setText("Apuyé");
                else ui->lblBouton->setText("Relaché");
                        }
   ui->lblInfos->setText(chargeUtile);
   data="";
   }

}
```

Compiler, executer, entrer BP1 ou BP0 dans le terminal, l'état du bouton s'affiche



Afficher des images

Il est possible de remplacer les textes de lblBouton par des images.
Créer (ou télécharger) deux images de boutons (appué et relaché)


led-off-100x100.png


led-off-100x100.png

Le type d'image est png et la taille des images 100x100 pixels, les images doivent être dans le dossier de compilation.

Modifier la méthode litData() comme suit :

```
if (data.startsWith("BP")==true) {
if(chargeUtile[0]=='1') ui->lblBouton->setPixmap(QPixmap("led-on-100x100.png"));
else ui->lblBouton->setPixmap(QPixmap("led-off-100x100.png"));
ui->lblBouton->show();
```

}

Cocher l'attribut la géométrie et scaledContens du widget lblBouton

geometry	[(60, 80), 100 x 100]
X	60
Y	80
Largeur	100
Hauteur	100

QLabel	
text	Bouton
textFormat	AutoText
pixmap	
scaledContents	☑

Compiler, executer, entrer BP1 , BP0 dans le terminal

Vous êtes capable de réaliser une application Qt mettant en œuvre les communications séries asynchrones et gérer un protocole de communication simple.

4 Coté microcontrôleur

Le code ci-dessous est compilable sur MBED pour STM32 mais peut être facilement adapté sur Arduino

Version avec interruption :

```
#include "mbed.h"

RawSerial pc(SERIAL_TX, SERIAL_RX); // IT sur serial
InterruptIn boutonBleu(USER_BUTTON);     // IT sur bouton bleu
DigitalOut led(LED1);          // led verte

void pc_recv()
{
static int etat=0;  // compteur machine à états
char c;
      c=pc.getc(); // recupere le caractère dans l'UART
      switch(etat)
      {
       case 0: if (c=='L')
              etat=1;
              break;
       case 1: if (c=='D')
              etat=2;
              break;
       case 2: if (c=='1')
              led=1;
              else led=0;
              etat=0;
      }
//       pc.printf("%d %c\n",etat,c);
}

void btnFall()
{
   pc.printf("BP1\n");
}

void btnRise()
{
   pc.printf("BP0\n");
}


int main()
{
   boutonBleu.fall(&btnFall); // IT sur appui bouton
   boutonBleu.rise(&btnRise); // IT sur relache bouton
   pc.attach(&pc_recv, Serial::RxIrq); // IT sur reception cacactere UART
   pc.baud(9600);
   while(1)
   {
      sleep();              // basse consommation
   }
}
```

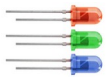
Version par scrutation

```
#include "mbed.h"

Serial pc(SERIAL_TX, SERIAL_RX);l
DigitalIn boutonBleu(USER_BUTTON);
DigitalOut led(LED1);          // led verte

void decode()
{
static int etat=0;  // compteur machine à états
char c;
      c=pc.getc(); // recupere le caractère dans l'UART
      switch(etat)
      {
       case 0: if (c=='L')
              etat=1;
              break;
       case 1: if (c=='D')
              etat=2;
              break;
       case 2: if (c=='1')
              led=1;
              else led=0;
              etat=0;
      }
//      pc.printf("%d %c\n",etat,c);
}

int main()
{
bool etatBtn=false;
   pc.baud(9600);
   while(1)
   {
      if (boutonBleu && !etatBtn) //le bouton vient d'etre relache
         {
            pc.printf("BP1\n");
            etatBtn=true;
         }
      if (!boutonBleu && etatBtn)//le bouton vient d'etre appuye
         {
            pc.printf("BP0\n");
            etatBtn=false;
         }
      if (pc.readable())      // un caractere a été recu par l'UART
      {
         decode();
      }
   }
}
```

5 Projet

On désire réaliser un logiciel de gestion d'un appareil connecté à un PC par UART.






La carte microcontrôleur possède :
Une led de signalisation
Un afficheur LCD
Un potentiomère
Un bouton poussoir
Un capteur de température.

Le logiciel gère ces éléments en fonctions du protocole de communication avec un PC suivant, chaque commande étant suivie d'un tiret d'un message et d'un passage à la ligne :

Périphérique	Sens	Action	Message ASCII	Paramètre ASCII
LED	PC→ uC	Allumage	LD	1
LED	PC→ uC	Extinction	LD	0
LED	PC→ uC	éclairement	EC	0 à 100
Aff LCD	PC→ uC	Afficher	LC	Message ASCII
Bouton	uC→ PC	Etat bouton	BT	1 ou 0
Potentiometre	uC→ PC	Valeur tension	VP	0.0 à 5.0
Capteur t°	uC→ PC	Valeur temperature	TE	0 à + 50°C

Exemples :
Pour allumer la led rouge le PC transmet : LD1
Pour afficher « bonjour » sur l'afficheur LCD le PC transmet : LCbonjour
Pour envoyer la température 25°C le uC transmet : TE25

Le format de communications est : 9600,n,8,1

Lors du développement du projet l'activité du microcontrôleur sera simulée par un simple terminal ASCII (ex Teraterm).

Le projet met en œuvre les widgets :

QRadioButton
QSlider
QLineEdit
QPushButton
QLCDNumber
QLabel
QProgressBar

Nommer le projet : ducpc
Nommer la classe du projet ucpc

Résultat attendu