RPI_SER	Mise en œuvre d'une liaison série sur Raspberry	
Système : Raspberry	**Durée : 3 heures**	**Travail individuel**

G.COLIN

Compétences :

Tester et valider un module logiciel et matériel

Savoirs :

Développement logiciel – Composants programmables – Modes de transmission – Test et validation.

Contexte et démarche

L'objectif est de mettre en œuvre une liaison série asynchrone entre une carte Raspberry et une carte Arduino.
La démarche consiste à :

- Tester la transmission sur Raspberry
- Tester la réception sur Raspberry en bouclant la transmission sur la réception.
- Vérifier le dialogue avec un analyseur de signaux

1 – Transmission UART sur Raspberry

⇨ Vérifier que la carte Raspberry a été correctement préparée :

- Se placer dans le répertoire /dev
- Taper la commande ls -l
- Vérifier que la liaison série 0 pointe vers ttyAMA0

```
lrwxrwxrwx 1 root root           7 oct.   7 23:17 serial0 -> ttyAMA0
lrwxrwxrwx 1 root root           5 oct.   7 23:17 serial1 -> ttyS0
```

Si ce n'est pas le cas, procéder de la mainère suivante :
Désactiver le mode console : Menu Préférences/Configuration du Raspberry PI/Interfaces ⇨ Serial console Désactivé
Sur raspberry pi 3, désactiver le bluetooth et retrouver le ttyAMA0 en serial0 en ajoutant : ***dtoverlay=pi3-disable-bt*** **au fichier /boot/config.txt. (édition ⇨ sudo nano /boot/config.txt)**
Rebooter la raspberry

⇨ Créer un nouveau projet (en mode console) sous CodeBlocks. Choisir le langage C++. Le nommer serie_uart et le placer dans le répertoire prog_cpp (/home/pi/prog_cpp).

⇨ Installer la librairie wiringPi dans le projet (répertoire /usr/lib). (voir RPI_GPIO)

⇨ Sélectionner la version 2011 du compilateur (c++11)

⇨ Copier (avec FileZilla par exemple) les 3 fichiers "wiringSerial.c", "gestion_gpio.cpp" et "gestion_gpio.h" dans le répertoire du projet "serie_uart".

⇨ Ajouter ces 3 fichiers au projet

Explications :

La fonction **init_gpio()** du fichier gestion_gpio.cpp:

Les lignes suivantes permettent de configurer la liaison ttyAMA0 à 9600 bauds

```
#define myBaud B9600
//configuration UART
    fileDescriptor=serialOpen ("/dev/ttyAMA0", 9600) ;
```

Dans le fichier **wiringSerial.c** , les lignes suivantes permettent de définir un mode 8 bits, sans parité

```
options.c_cflag |= CS8 ;
options.c_cflag &=~PARENB;// No Parity   |= PARENB ;  // Enable Parity - even by default
```

La fonction **txData(unsigned char don)** du fichier gestion_gpio.cpp

Permet de transmettre une donnée sur la liaison série

La fonction **bool rxData(unsigned char *p)** du fichier gestion_gpio.cpp

La fonction retourne vrai si une donnée a été reçue sur la liaison série, faux sinon.

La donnée reçue est stockée dans la variable dont on a transmis l'adresse.

⇨ Modifier le programme principal de la manière suivante :

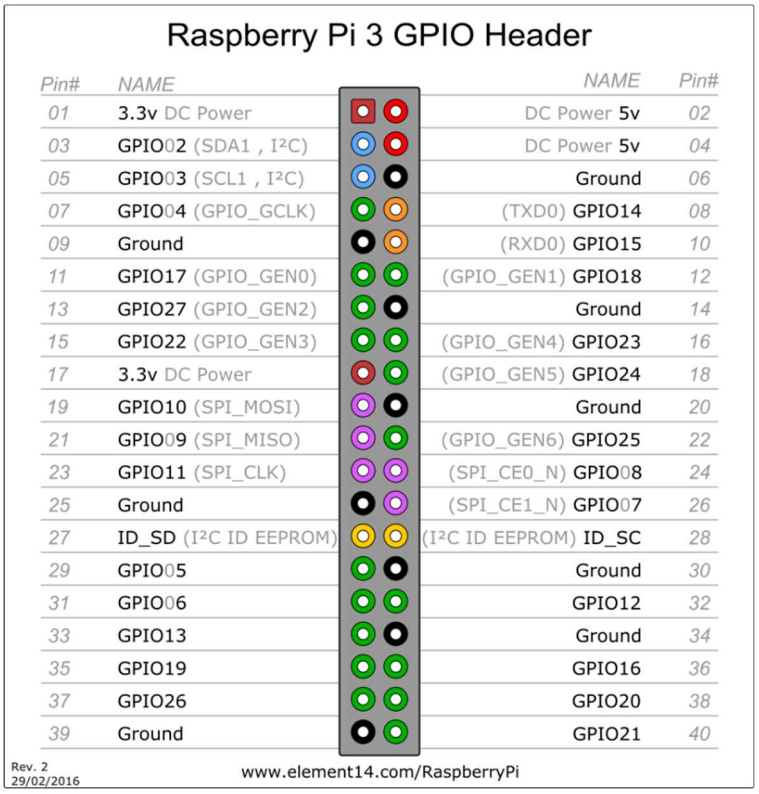
```
#include <iostream>
#include <wiringPi.h>
#include "gestion_gpio.h"

using namespace std;

int main()
{
    init_gpio();
    while(1)
    {
        txData(0x5A);
        delay(2);
    }
    return 0;
}
```

⇨ Expliquer ce que fait le programme principal

⇨ Repérer la ligne TXD0 et la masse sur le connecteur GPIO

⇨ Demander des fils adaptés et une sonde pour relever le signal transmis à l'oscilloscope.

⇨ Relever le signal et vérifier la donnée transmise, la durée d'un bit, la périodicité de la transmission.



⇨ Modifier le programme pour transmettre à 19200 bauds. Tester à l'oscilloscope et conclure.

On souhaite transmettre une chaîne de caractères

⇨ Modifier le programme principal de la manière suivante (les fichiers d'entête restent les mêmes).

```
char PRET[]="PRET?";

void txMessage(char *data);

int main()
{
    init_gpio();
    while(1)
    {
        txMessage(PRET);
        delay(1000);
    }
    return 0;
}

void txMessage(char *data)
{
    while(*data!=0)
    {
        txData(*data);
        data++;
    }
}
```

La fonction txMessage(char *data) transmet une chaîne de caractère.
⇨ Expliquer pourquoi, dans cette fonction, la donnée est comparée à la valeur 0.

On souhaite relever la transmission du message à l'aide d'un analyseur de signaux.

⇨ Demander au prof le module SQ200 et demander une démonstration pour l'utilisation.

⇨ Relever le message transmis avec ce module.

⇨ Modifier la fonction txMessage pour transmette le code 13 (=0x0d= code ASCII de Entrée= Carriage Return) à la fin de la chaîne, donc à la sortie de la boucle while(*data!=0).

⇨ Vérifier la nouvelle transmission avec le module SQ200

2 – Réception UART sur Raspberry

2.1 - Réception d'un caractère :

⇨ Modifier la fonction main() de la manière suivante (le reste est inchangé):

```
int main()
{
    unsigned char a;
    init_gpio();
    while(1)
    {
        txData('A');
        if (rxData(&a))
        {
            cout<<"reçue: "<<a<<endl;
        }
        delay(1000);
    }
    return 0;
}
```

Par mesure de sécurité, on utilise des câbles avec une résistance en série pour éviter les courts circuits.
⇨ Demander un câble au prof et boucler la transmission sur la réception
⇨ Tester et conclure.

2.2 – Réception d'une chaîne de caractère

⇨ Ajouter la fonction rxMessage ci-dessous au fichier main.cpp (ne pas oublier de la déclarer également en début de fichier).

```
string rxMessage(void)
{
    unsigned int i=0;
    unsigned char a;
    string st="";
    while(i<100)
    {
        if(rxData(&a))
        {
            if (a==13) return st;
            st=st + char(a);
            i=0;
        }
        i++;
        delay(10);
    }
    return "PB\n";
}
```

Cette fonction dispose d'un "time out" de 1 seconde. Si elle ne reçoit pas le code 13 avant une seconde, elle retourne "PB". Sinon elle retourne le message reçu à la détection du code 13 (Entrée).

⇨ Justifier, en analysant cette fonction, la valeur de 1 seconde.
⇨ Comment modifier le time out à 2 secondes.

⇨ Modifier la fonction main() de la manière suivante :

```
int main()
{
    init_gpio();
    string st;
    while(st!="f")
    {
        cin>> st;
        if (st=="1") txMessage(PRET);
        if (st=="2") txMessage(ERREUR);
        cout<<"message reçu: "<<rxMessage()<<endl;
    }
    return 0;
}
```

⇨ Ajouter le message "ERREUR":

```
char PRET[]="PRET?";
char ERREUR[]="ERREUR";
```

⇨ Tester la transmission des messages en bouclant la transmission sur la réception. Conclure.

⇨ Avec FilleZilla, sauvegarder le répertoire du projet (serie_uart) dans le répertoire personnel sur H :

⇨ Effacer le répertoire serie_uart sur la raspberry.