

François SCHNEIDER – Lycée Victor-Hugo BESANÇON

## Développement à base de cartes de prototypage rapide MBED

### Programmation Wifi réseau avec MBED et Wlifly

Prérequis : langage C, programmation objet, notion de programmation socket : client, serveur, réseau IP, Wifi ...



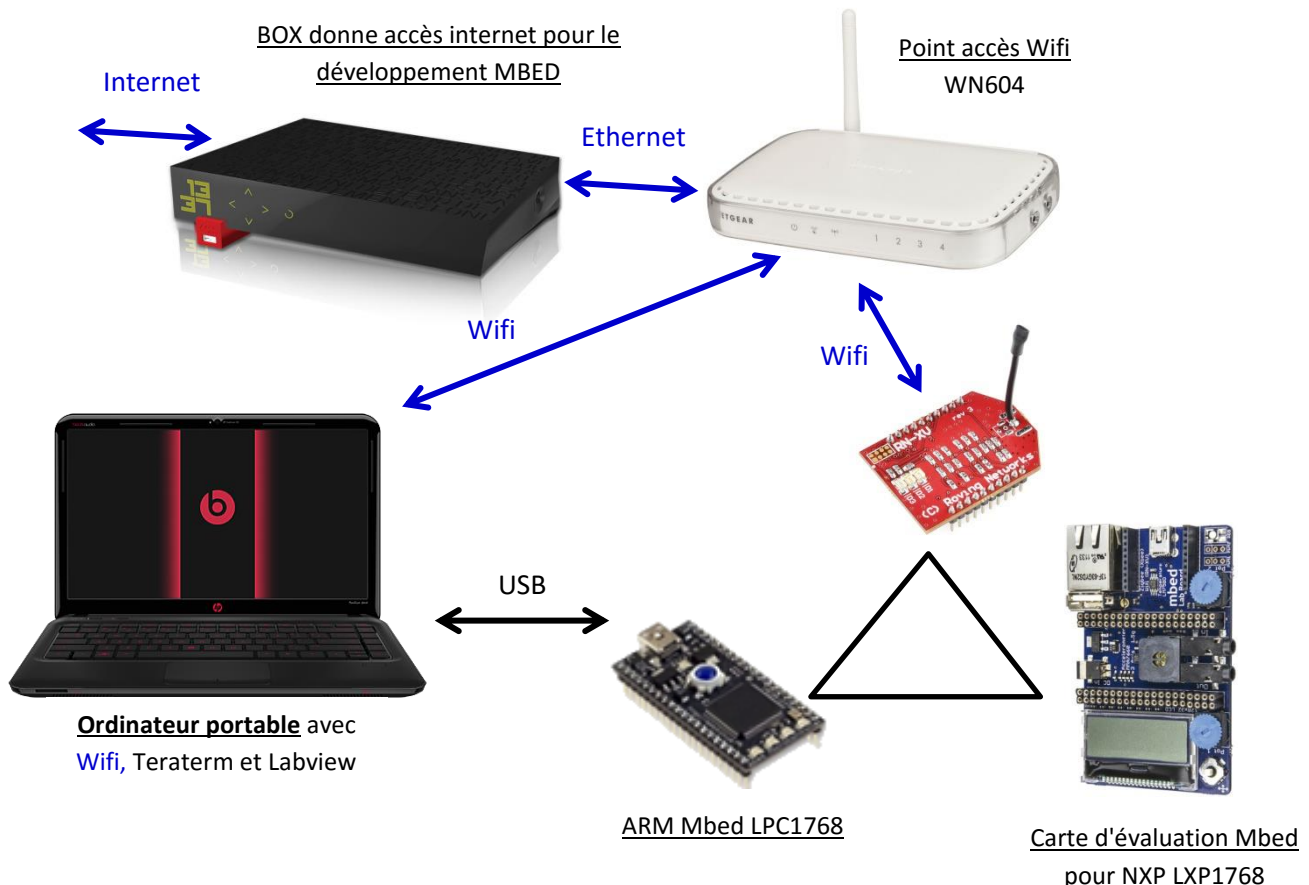
## Contenu

Ce que vous saurez faire.....	3
Matériel proposé.....	3
Remarque importante pratique. ....	4
Création du projet. ....	4
Mise en place d'un port série virtuel sur la connexion USB.....	4
Création d'un nouveau projet et préparation de l'environnement MBED pour effectuer de la programmation réseau.....	6
Programmation réseau – mise en situation et définition des rôles.....	7
Configuration du point d'accès : .....	7
Paramètres Ip et Tcp .....	8
Sauvegarde du projet en cours. ....	8
Mise en place des couches Wifi et IP. ....	9
Programmation réseau premier serveur.....	11
Programmation réseau deuxième serveur.....	13

## Ce que vous saurez faire.

Vous allez apprendre à effectuer de la programmation réseau à partir d'une solution de prototypage MBED et un module Wifly. Vous serez capable de dialoguer en socket entre un PC et une carte MBED en wifi, dans un premier temps en mode serveur puis en mode client. Vous saurez aussi utiliser la liaison USB de ma carte MBED en mode série virtuelle.

## Matériel proposé.



Un seul ordinateur est utilisé pour le développement et les essais :

- La solution est une solution de type infrastructure. Un point d'accès relie les équipements en dialogue. Le point d'accès WN604 à ce rôle. Il est relié à la box, qui donne un accès à Internet pour le développement (site MBED) au Pc.
- Dans cette configuration d'autres solutions sont envisageables, par exemple le Pc peut être relié à la box ou au point d'accès en Ethernet ou le Pc peut être relié à la box en Wifi.
- Le PC et la carte MBED pourront être configurés en DHCP sachant que la Box rend ce service. Je conseille d'attribuer une adresse Ip fixe à la carte MBED afin de pouvoir la contacter plus facilement.

Dans le cadre d'un lycée, je conseille d'utiliser 2 ordinateurs PC, afin d'avoir un réseau de développement indépendant du réseau de production : très grand danger de pollution. Dans ce cas :

- Le PC de développement est relié au réseau du lycée en Ethernet et à la carte MBED en USB.
- Le PC de test est relié au point d'accès Wifi en Wifi ou en Ethernet. Il faudra dans ce cas pour ce réseau utiliser des adresses Ip fixe pour ce Pc et la carte MBED, sachant qu'il n'y a pas de serveur DHCP dans ce réseau.

### Remarque importante pratique.

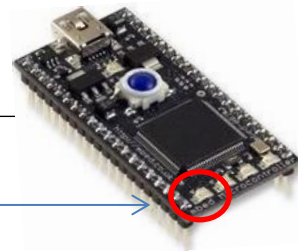
Ce document est écrit pour une utilisation avec un seul Pc et un point d'accès relié à un réseau avec un serveur DHCP. **Il faudra donc s'adapter pour le cas de l'utilisation de 2 Pc.** Nous a **Création du projet.**

Nous allons tout reprendre à 0. Vous devez certainement savoir faire le travail, qui suit.

Dans votre interface de travail sur mbed.org, vous créez un nouveau projet TP5\_Wifi. Ce projet vous servira de base pour la programmation réseau.



Nous utilisons la template « Blinky LED hello world »



Nous testons le fonctionnement, la LED indiquée clignote.

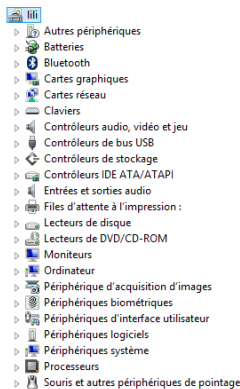
### Mise en place d'un port série virtuel sur la connexion USB.

Nous allons mettre en place un port série virtuel sur la liaison USB. Ce port série virtuel sera utile pour tous vos développements et vous permettra de mettre un dialogue entre le pc de développement et la carte MBED. Vous pourrez dans vos programmes ajouter des messages de fonctionnement et ainsi voir comment votre programme se déroule.

Pour Windows, il suffit d'installer le driver pour le port série virtuel pour la carte MBED :

<http://developer.mbed.org/handbook/Windows-serial-configuration>

Après avoir installé le port COM virtuel, nous allons le tester :



Les périphériques de Windows ne possèdent pas de port série MBED, lorsque que le driver n'est pas installé.

Lorsque le driver est installé et que vous branchez la carte MBED un port série apparait. Attention si vous utilisez un PC pour lequel vous avez des droits limités (par exemple Scribe) vous devez faire installer le driver par l'administrateur du poste.

Pour utiliser un port série virtuel dans les programmes, il suffit d'instancier un objet avec la classe serial dont les attributs sont USBTX, USBRX :

```
Serial pc(USBTX, USBRX);
```

Le fichier main.cpp devient.


```
#include "mbed.h"
DigitalOut myled(LED1);
Serial pc(USBTX, USBRX);
int main() {
    while(1) {
        myled = 1;
        wait(0.2);
        myled = 0;
        wait(0.2);
    }
}
```

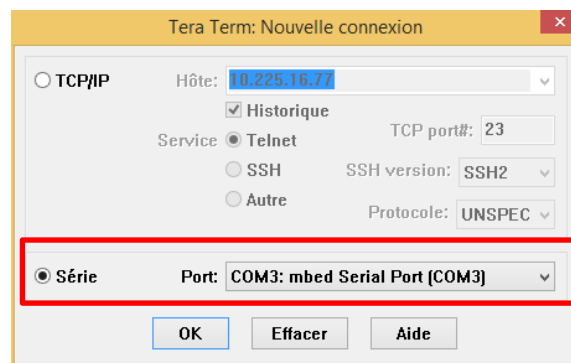
Le port série pc possède les attributs de la classe Serial. Il aura donc les mêmes attributs et méthodes qu'un port série physique.

Nous pourrons donc utiliser la méthode printf() pour envoyer un message ou les méthodes readable(), getc() pour lire un caractère ...

Le fichier main.ccp pour l'envoi du message "Hello, I'm happy my Virtual Serial port is ok." est donné ci-dessous :

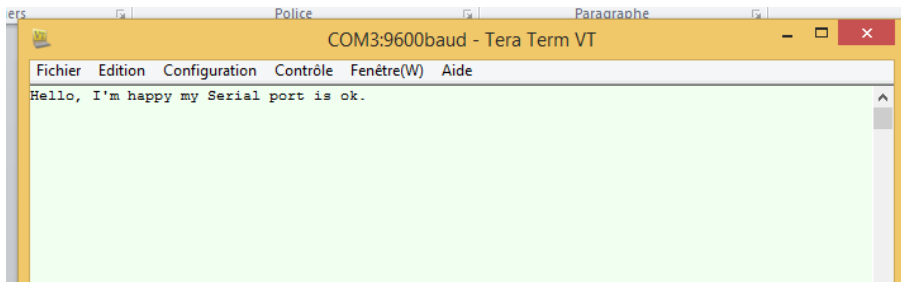
```
#include "mbed.h"
DigitalOut myled(LED1);
Serial pc(USBTX, USBRX);
int main() {
    pc.printf("Hello, I'm happy my Virtual Serial port is ok.\n\r");
    while(1) {
        myled = 1;
        wait(0.2);
        myled = 0;
        wait(0.2);
    }
}
```

Il ne reste plus qu'à tester son fonctionnement à l'aide du logiciel TeraTerm . Personnellement j'utilise la version portable. Vous lancez Teraterm et sélectionnez le port virtuel série :



Vous compilez le programme, vous le chargez dans la mémoire de la carte MBED.

Vous appuyez sur le bouton RESET de la carte MBED pour exécuter le programme et en principe vous obtenez le résultat suivant :



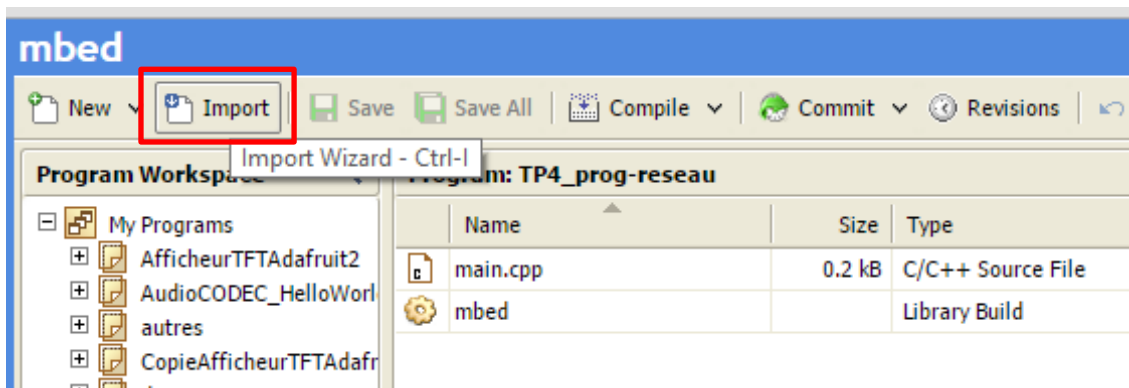
Tout va bien, nous allons pouvoir commencer la programmation réseau.

Le port virtuel va servir à envoyer des messages.

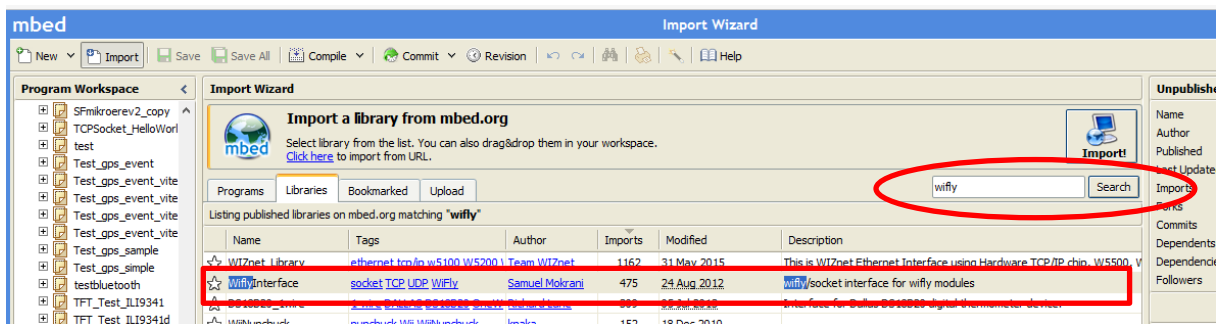
### Création d'un nouveau projet et préparation de l'environnement MBED pour effectuer de la programmation réseau.

Nous allons maintenant ajouter au projet courant la librairie « WiflyInterface » pour le module Wifly. Elle comprend la gestion du module Wifly et les librairies Socket.

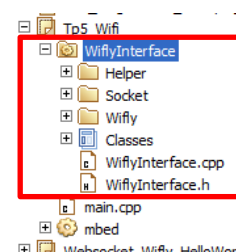
Vous sélectionnez votre projet et vous importez la librairie « WiflyInterface » en cliquant sur « Import ».



Vous recherchez la chaîne Wifly



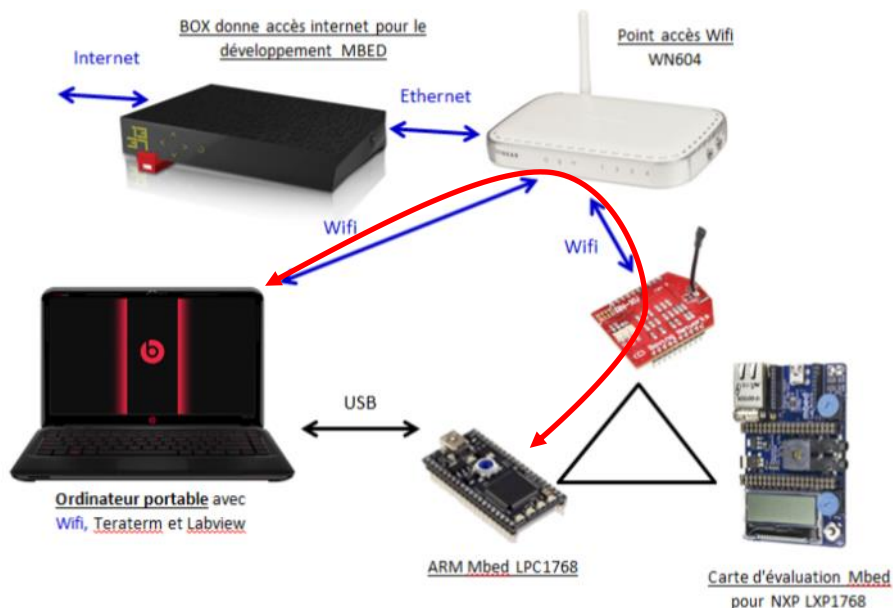
Dans la fenêtre en dessous vous sélectionnez la librairie WiflyInterface. L'importation s'effectue. Vous pouvez vérifier que la librairie a été ajoutée à votre projet.



[Votre environnement est prêt pour le développement réseau.](#)

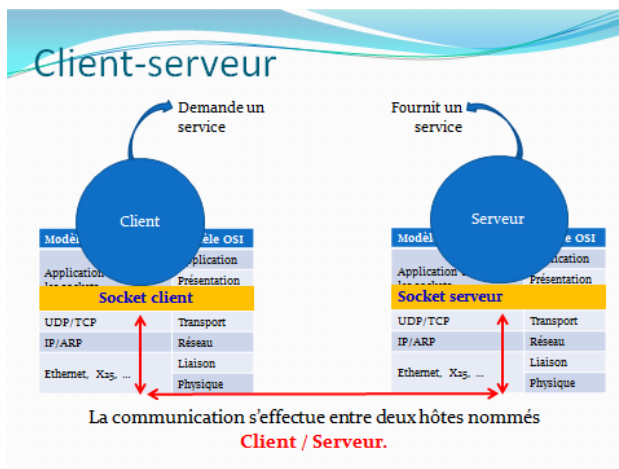
**Programmation réseau – mise en situation et définition des rôles.**

Nous allons maintenant établir une communication entre la carte MBED et le pc à travers le réseau :



Remarque : La box fournit un service DHCP. Dans un premier temps nous allons utiliser ce service pour la carte MBED. Dans un deuxième temps, nous attribuerons une adresse Ip fixe à la carte MBED.

**Rappel : notion de client-serveur.**



La programmation réseau permet d'établir un dialogue en mode client-serveur :

Nous allons choisir le protocole connecté donc TCP.

Le PC aura le rôle de client et utilisera pour cela le client Teraterm.

La carte MBED aura le rôle de serveur. Nous allons la programmer pour cela.

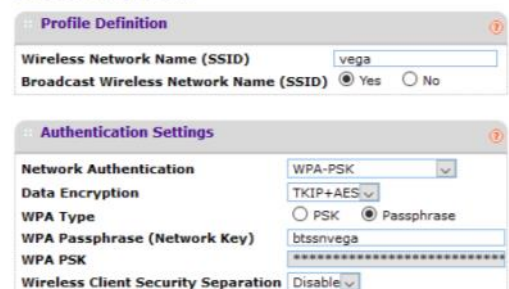
**Configuration du point d'accès :**

Adresse Ip : 192.168.0.100.

**Sécurité Wifi :**

- SSID : **vega**,
- Authentification : **WPA-PSK**,
- Cryptage : **TKIP + AES**,
- Mot de passe : **btssnvega**.

**Edit Security Profile**



Vous devez personnaliser le SSID et le mot de passe.

## Paramètres Ip et Tcp

Nous allons définir les paramètres Ip, qui seront utilisés tout au long de cette activité.

### Configuration Ip

Adresse réseau	192.168.0.0
Masque sous réseau	255.255.255.0
Passerelle	192.168.0.254
PC	DHCP
Carte MBED	DHCP puis 192.168.0.20

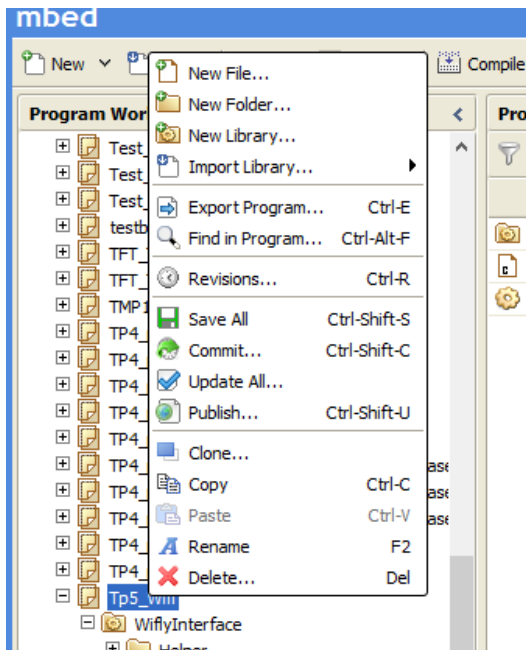
Attention si vous utilisez une adresse Ip fixe pour la carte Mbed, vous devez soit la réserver au niveau du serveur DHCP ou au minimum vérifier, qu'elle n'est pas prise par un autre appareil.

### Configuration Tcp

Port utilisé	2000
--------------	------

## Sauvegarde du projet en cours.

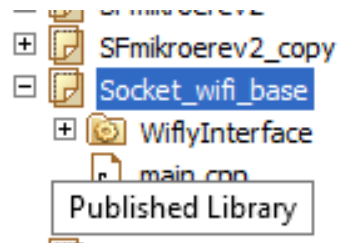
Le projet courant « Tp5\_Wifi » est une base intéressante pour la programmation réseau en wifi.



Il est possible de garder une copie avant de continuer.

Vous indiquez le nouveau nom par exemple « Socket\_wifi\_base » et validez.

Le nouveau projet apparaît :



Nous continuons à travailler avec le projet « TP5\_Wifi ».



## Mise en place des couches Wifi et IP.

Nous allons utiliser le port série virtuel pour afficher les informations de fonctionnement.

Au niveau du programme, il va s'agir d'ajouter un objet de class « **WiflyInterface** » et ensuite de vérifier son fonctionnement. Nous allons ajouter les éléments en rouge au programme précédent.

```
#include "mbed.h"
#include "WiflyInterface.h"
DigitalOut myled(LED1);
Serial pc(USBTX, USBRX);
/* creation de l'instance wifly
- sécurité : WPA
- SSID : vega
- mt de passe : btssnvega
Attention au sens des broches : TX RX Reset Status */
WiflyInterface wifly(p9, p10, p30, p29, "vega", "btssnvega", WPA);
int main() {
    pc.printf("Hello, I'm happy my Virtual Serial port is ok.\n\r");
    pc.printf("Init Wifi.\n\r");
    if (wifly.init()!=0) { //Initialise interface ici dhcp
        pc.printf("ERREUR INIT ETHERNET\r\n");
        return -1;
    }
    pc.printf("Init Wifi ok.\n\rConnect Wifly.\n\r");
    unsigned i = 0;
    while (!wifly.connect()) { // Connecte interface
        pc.printf("Attente connexion %ds\r\n",i);
        i = i + 1;
        if (i==5) {
            pc.printf("5 tests en erreur, ERREUR Connect, quitte programme\r\n");
            return -1;
        }
    }
    pc.printf("Ip sur Wifly is Ok.\n\r");
    printf("IP Address is %s\n\r", wifly.getIPAddress());
    pc.printf("\r\nVous devez maintenant installer Tcp.\n\r");
    while(1) {
        myled = 1;
        wait(0.2);
        myled = 0;
        wait(0.2);
    }
}
```

Les méthodes `init()` et `connect()` lorsqu'elles se déroulent sans erreur renvoient une valeur nulle. En cas d'erreur d'une de ces méthodes, nous constatons qu'un message est envoyé vers l'interface série virtuel et que le programme est quitté en envoyant la valeur -1 au système d'exploitation MBED.

```

COM4:9600baud - Tera Term VT
Fichier Edition Configuration Contrôle Fenêtre(W) Aide
Hello, I'm happy my Virtual Serial port is ok.
Init Wifi.
Init Wifi ok.
Connect Wifly.
[Wifly : INFO]
ssid: vega
phrase: btssnvega
security: WPA

Ip sur Wifly is Ok.
IP Address is 192.168.0.10

Vous devez maintenant installer Tcp.

```

Nous testons le programme, le résultat est le suivant :

Nous constatons que l'ensemble s'est bien déroulé.

Nous voyons l'utilité d'utiliser le port virtuel pour envoyer des messages.

L'adresse IP de la carte MBED a été fournie par le serveur DHCP, nous allons maintenant attribuer une adresse Ip fixe à l'interface Wifi. Pour cela il suffit de modifier les paramètres de la méthode `init()` :

```

if (wifly.init("192.168.0.20","255.255.255.0","192.168.0.254")!=0) { //Initialise interface ici dhcp
    pc.printf("ERREUR INIT ETHERNET\r\n");
    return -1;
}

```

Nous recommençons le test. Nous constatons que l'adresse de l'interface de la carte Mbed est bien 192.168.0.20. Attention la carte MBED et le PC de tests doivent être dans le même réseau.

```

COM4:9600baud - Tera Term VT
Fichier Edition Configuration Contrôle Fenêtre(W) Aide
Hello, I'm happy my Virtual Serial port is ok.
Init Wifi.
Init Wifi ok.
Connect Wifly.
[Wifly : INFO]
ssid: vega
phrase: btssnvega
security: WPA

Ip sur Wifly is Ok.
IP Address is 192.168.0.20

Vous devez maintenant installer Tcp.

```

Nous pouvons vérifier que la carte MBED répond à un ping :

```

Microsoft Windows [version 10.0.10586]
(c) 2015 Microsoft Corporation. Tous droits réservés.

C:\Users\francois>ping 192.168.0.20

Envoi d'une requête 'Ping' 192.168.0.20 avec 32 octets de données :
Réponse de 192.168.0.20 : octets=32 temps=3 ms TTL=255
Réponse de 192.168.0.20 : octets=32 temps=1 ms TTL=255
Réponse de 192.168.0.20 : octets=32 temps=7 ms TTL=255
Réponse de 192.168.0.20 : octets=32 temps=2 ms TTL=255

Statistiques Ping pour 192.168.0.20:
    Paquets : envoyés = 4, recus = 4, perdus = 0 (perte 0%),
    Durée approximative des boucles en millisecondes :
        Minimum = 1ms, Maximum = 7ms, Moyenne = 3ms

```

**Programmation réseau premier serveur.**

Nous allons dans un premier temps sauvegarder le projet « TP5\_Wifi » sous le nom « Tp5\_Wifi-premier-serveur ». Nous utiliserons ce projet dans cette partie.

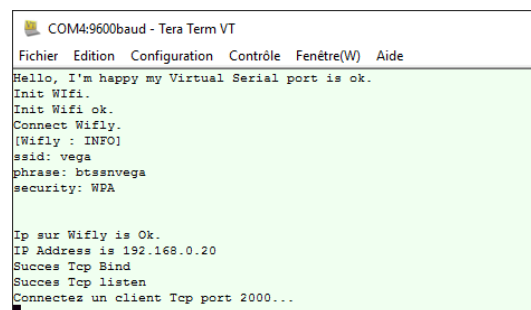
Ce premier serveur n’a pas de véritable utilité mais il servira de base pour le développement de serveur plus complexe. Son fonctionnement :

Serveur : carte MBED	
Création socket serveur	#define serveur_port 2000
Affectation d’un port d’écoute au serveur	TCPsocketServer server;
Préparation du serveur pour l’écoute	server.bind (serveur_port);
Création socket client	server.listen();
Attente connexion d’un client	TCPsocketConnection client;
Envoi message au client	server.accept (client)
Fermeture connexion client	client.send("Hello\r\n",strlen("Hello\r\n"));
	client.close();

Dans la phase développement, il est conseillé d’envoyer des messages à travers l’interface série virtuel, afin de vérifier le fonctionnement.

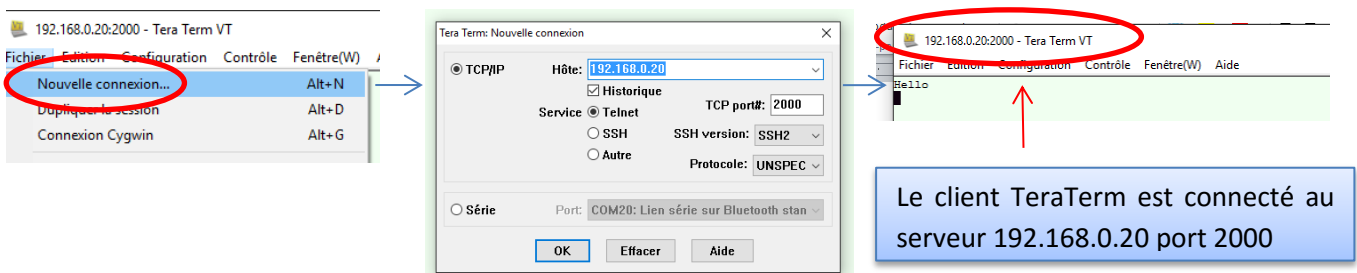
Vous trouverez le programme du serveur page suivante. Le programme n’est pas commenté, il faut le faire au fur et à mesure de l’écriture. Nous allons vérifier son fonctionnement. Vous compilez le programme et le copiez dans la carte MBED.

Vous connectez avec Teraterm le port série virtuel, si cela n’est pas fait avant de « reseter » la carte MBED. Tout se passe bien vous obtenez le message à droite :



Nous constatons que la LED ne clignote pas, cela veut dire que la fonction « server.accept (client) » est bloquante, elle attend la connexion d’un client.

Vous créez une nouvelle connexion, qui va ouvrir une nouvelle fenêtre. La première fenêtre est ouverte avec COM3, tandis que la deuxième avec l’adresse IP et le port du serveur.



Le serveur répond « Hello ». Le message s'affiche dans la fenêtre du client TCP Teraterm. Lorsque vous avez terminé, vous pouvez fermer la fenêtre client TCP.

L'instruction « // client.close(); » est placée en commentaire. Retirer la mise en commentaire, compiler et exécuter à nouveau le programme. Expliquer le résultat obtenu.

#### Main.ccp() premier serveur TCP :

```
#include "mbed.h"
#include "WiflyInterface.h"
DigitalOut myled(LED1);
Serial pc(USBTX, USBRX);
/* creation de l'instance wifly
- sécurité : WPA
- SSID : vega
- mt de passe : btssnvega
Attention au sens des broches : TX RX Reset Status */
WiflyInterface wifly(p9, p10, p30, p29, "vega", "btssnvega", WPA);
TCPsocketServer server;
TCPsocketConnection client;
int serveur_port = 2000;
int main()
{
    pc.printf("Hello, I'm happy my Virtual Serial port is ok.\n\r");
    pc.printf("Init Wifi.\n\r");
    if (wifly.init("192.168.0.20","255.255.255.0","192.168.0.254")!=0) { //Initialise interface ici dhcp
        pc.printf("ERREUR INIT ETHERNET\r\n");
        return -1;
    }
    pc.printf("Init Wifi ok.\n\rConnect Wifly.\n\r");
    unsigned i = 0;
    while (!wifly.connect()) { // Connecte interface
        pc.printf("Attente connexion %ds\r\n",i);
        i = i + 1;
        if (i==5) {
            pc.printf("5 tests en erreur, ERREUR Connect, quitte programme\r\n");
            return -1;
        }
    }
    pc.printf("Ip sur Wifly is Ok.\n\r");
    printf("IP Address is %s\n\r", wifly.getIPAddress());
    if (server.bind (serveur_port)!=0) {
        pc.printf("Erreur Tcp bind\r\n");
        return -1;
    }
    pc.printf("Succes Tcp Bind\r\n");
    if (server.listen()!=0) {
        pc.printf("ERREUR Tcp Listen\r\n");
        return -1;
    }
    pc.printf("Succes Tcp listen\r\n");
    pc.printf("Connectez un client Tcp port 2000...\r\n");
    if (server.accept (client)!=0) {
        pc.printf("ERREUR Tcp Listen\r\n");
        return -1;
    }
    pc.printf("Client Tcp connecte\r\n");
    pc.printf("Adresse Ip client %s\r\n",client.get_address());
    client.send("Hello\r\n",strlen("Hello\r\n"));
    // client.close()
}
```

Nous constatons que tant que le serveur est en attente d'une connexion d'un client, la LED ne clignote pas. Cela est dû au fait que la méthode « accept () » est bloquante. Il n'est plus ensuite possible de connecter un client au serveur. Nous allons voir par la suite comment opérer.

### Programmation réseau deuxième serveur.

---

Vous avez vu précédemment :

- comment créer un serveur,
- accepter une demande connexion d'un client,
- envoyer un message à ce client,
- fermer la connexion avec le client.

Maintenant, il va falloir répéter ceci et permettre de gérer des connexions de clients. Il ne sera cependant possible de connecter qu'un seul client à la fois.

Nous allons dans un premier temps sauvegarder le projet « Tp5\_Wifi-premier-serveur » sous le nom «Tp5\_Wifi-deuxieme-serveur». Nous utiliserons ce projet dans cette partie.

Résultat demandé : les données en provenance du port série virtuel seront copiées vers l'interface Ethernet en TCP et vice versa.

Certaines fonctions sockets sont bloquantes, pour les rendre non bloquantes, nous allons utiliser la méthode : **client.set\_blocking(false, 100); // Timeout after (100ms)**. Attention à la valeur du Timeout, trop long, il donne un temps de latence, trop court il peut y avoir des erreurs de temps de réponse.

Le programme page suivante vous donne une base. Attention la fonction receive() ne renvoie de données que lorsqu'elle reçoit le code de la touche entrée.

Une variable "**bool client\_connected = false;**" est créée et mise à false au départ. Elle permet de vérifier si un client est connecté.

Quelques explications :

Vous regardez la boucle while(1) et établissez un organigramme du programme. Validez le fonctionnement de ce programme. Vous ajoutez au programme l'allumage de la LED1 lors de la connexion avec un client.

Main.ccp() deuxième serveur Tcp :

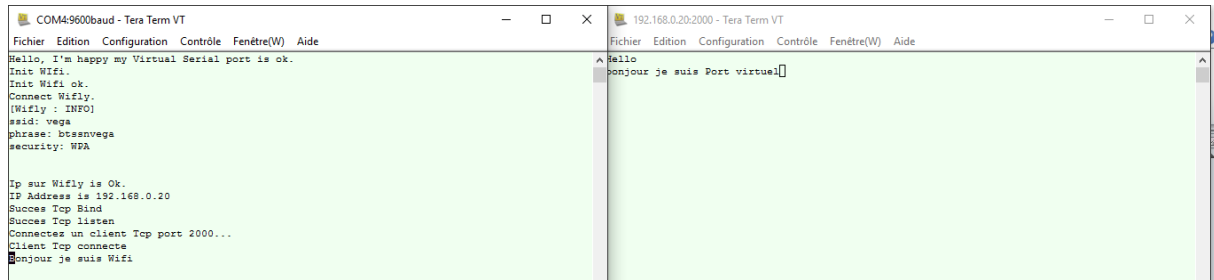
```

#include "mbed.h"
#include "WiflyInterface.h"
DigitalOut myled(LED1);
Serial pc(USBTX, USBRX);
/* creation de l'instance wifly
- sécurité : WPA
- SSID : vega
- mt de passe : btssnvega
Attention au sens des broches : TX RX Reset Status */
WiflyInterface wifly(p9, p10, p30, p29, "vega", "btssnvega", WPA);
TCPSocketServer server;
TCPSocketConnection client;
int serveur_port = 2000;
int main()
{
    bool client_connected = false;
    char buffer [100];
    pc.printf("Hello, I'm happy my Virtual Serial port is ok.\n\r");
    pc.printf("Init Wifi.\n\r");
    if (wifly.init("192.168.0.20", "255.255.255.0", "192.168.0.254")!=0)
    { //Innitialise interface ici dhcp
        pc.printf("ERREUR INIT ETHERNET\r\n");
        return -1;
    }
    pc.printf("Init Wifi ok.\n\rConnect Wifly.\n\r");
    unsigned i = 0;
    while (!wifly.connect()) { // Connecte interface
        pc.printf("Attente connexion %ds\r\n",i);
        i = i + 1;
        if (i==5) {
            pc.printf("5 tests en erreur, ERREUR Connect, quitte
programme\r\n");
            return -1;
        }
    }
    pc.printf("Ip sur Wifly is Ok.\n\r");
    printf("IP Address is %s\n\r", wifly.getIPAddress());

    if (server.bind (serveur_port)!=0) {
        pc.printf("Erreur Tcp bind\r\n");
        return -1;
    }
    pc.printf("Succes Tcp Bind\r\n");
    if (server.listen()!=0) {
        pc.printf("ERREUR Tcp Listen\r\n");
        return -1;
    }
    pc.printf("Succes Tcp listen\r\n");
    pc.printf("Connectez un client Tcp port 2000...\r\n");

    client.set_blocking(false, 100); // Timeout after (100ms)
    while(1) {
        if (client_connected==false) {
            if (server.accept (client)!=0) {
                pc.printf("ERREUR connexion client\r\n"); // erreur
                return -1;
            }
            client_connected = true;
            pc.printf("Client Tcp connecte\r\n");
            client.send("Hello\r\n",strlen("Hello\r\n"));
        } else {
            if ( client.is_connected() ) {
                if (pc.readable()) { // interface virtuel serie recoit donnees
                    buffer[0] = pc.getc();
                    client.send(buffer,1);
                }
                int nbcaractere = client.receive(buffer,sizeof(buffer));
                if (nbcaractere>0) { // interface Wifi recoit donnees
                    buffer[nbcaractere] = 0;
                    pc.printf("%s",buffer);
                }
            } else {
                client_connected = false;
                client.close();
                pc.printf("Client deconnecte\r\n");
                pc.printf("Connectez un client Tcp port 2000...\r\n");
            }
        }
    }
}

```



The image shows two side-by-side Tera Term VT windows. The left window, titled 'COM4:9600baud - Tera Term VT', displays the output of a program running on a virtual serial port. The text includes: 'Hello, I'm happy my Virtual Serial port is ok.', 'Init WiFi.', 'Init WiFi ok.', 'Connect WiFi.', '[WiFi : INFO]', 'ssid: vega', 'phrase: btsanvega', 'security: WPA', 'Ip sur Wifly is Ok.', 'IP Address is 192.168.0.20', 'Succes Tcp Bind', 'Succes Tcp listen', 'Connectez un client Tcp port 2000...', 'Client Tcp connecte', and 'Bonjour je suis Wifi'. The right window, titled '192.168.0.20:2000 - Tera Term VT', shows the output of a program running on a TCP port. It displays 'hello' and 'bonjour je suis Port virtuel'.

Nous voyons le dialogue s'effectuer entre le port virtuel série (écran gauche) et le port Ethernet en TCP (écran droit).

- Nous nous plaçons sur l'écran à gauche (port série virtuel). Nous écrivons et nous constatons que les données s'affichent immédiatement sur l'écran à droite.
- Par contre lorsque nous nous plaçons sur l'écran à droite, les données ne sont pas affichées immédiatement sur l'écran à gauche. Il faut valider avec la touche « enter ».

Vous pouvez compléter ce Tp en utilisant le Tp de programmation socket Ethernet « **TP4 Mbed-programmation reseau - serveur - client – labview** ».