BTS SN IR	Réseaux	
	ARDUINO - SERVEUR	

Arduino et Ethernet : serveur

L'utilisation de l'Arduino en mode serveur est sûrement plus courante que celle en client.

Deux grands rôles peuvent être accomplis :

- L'envoi de données *à la demande* (l'utilisateur vient demander les données quand il les veut).
- La réception d'ordre pour effectuer des actions.

Ces deux rôles ne sont pas exclusifs, ils peuvent tout à fait cohabiter ensemble.

Un serveur est chargé de réceptionner du trafic, l'interpréter puis agir en conséquence. Pour cela, il possède un **port** particulier qui lui est dédié. Chaque octet arrivant sur ce port lui est donc destiné. On dit que le serveur **écoute** sur un **port**.

C'est donc à partir de cela que nous allons pouvoir mettre en place notre serveur !

Il faut commencer par les options du shield (MAC, IP…) afin que ce dernier puisse se connecter à votre box/routeur.

On aura les variables suivantes :

```
// Ces deux bibliothèques sont indispensables pour le shield
#include <SPI.h>
#include <Ethernet.h>

/* Connexion ethernet */
// L'adresse MAC du shield
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
// L'adresse IP que prendra le shield
byte ip[] = { 192,168,1, 20 };
// L'adresse IP de la passerelle
byte gateway[] = { 192,168,1, 1 };
```

Maintenant que nous avons nos variables, nous allons pouvoir démarrer notre shield dans le setup().
Pour cela, il suffira d'appeler une fonction bien nommée : begin().
Cette fonction prendra trois paramètres, l'adresse MAC, l'adresse IP à utiliser et l'adresse IP de la passerelle.

```
void setup()
{
 // Initialisation liaison série pour déboguer
  Serial.begin(9600);
  Serial.println("Liaison Serie ok ...");
  // Configuration de la Ethernet shield
  Ethernet.begin(mac, ip, gateway);
  // Donne une seconde au shield pour s'initialiser
  delay(1000);
}
```

Pour le serveur, il faut créer une variable de type EthernetServer qui prendra un paramètre : le port d'écoute. J'ai choisi 2000 de manière un peu aléatoire, car je sais qu'il n'est pas utilisé sur mon réseau

```
// Attachement d'un objet "server" sur le port 2000
EthernetServer server(2000);
```

Puis, à la fin de notre setup il faudra démarrer le serveur avec la commande suivante :

```
// On lance le serveur
server.begin();
Serial.println("Serveur ok ...");
```

En résumé, on aura donc le code suivant pour l'initialisation :

```
#include <SPI.h>
#include <Ethernet.h>

/* Connexion ethernet */
// L'adresse MAC du shield
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
// L'adresse IP que prendra le shield
byte ip[] = { 192,168,1, 20 };
// L'adresse IP de la passerelle
byte gateway[] = { 192,168,1, 1 };

// Attachement d'un objet "server" sur le port 2000
EthernetServer server(2000);

void setup()
{
 // Initialisation liaison série
  Serial.begin(9600);
  Serial.println("Liaison Serie ok ...");
  // Configuration de la ethernet shield et du server
  Ethernet.begin(mac, ip, gateway);
  server.begin();
  // Donne une seconde au shield pour s'initialiser
  delay(1000);
  Serial.println("Serveur ok ...");
}
```

Et voilà, votre serveur Arduino est en train de surveiller ce qui se passe sur le réseau !

Maintenant que le serveur est prêt et attend qu'on lui parle, on va pouvoir coder la partie "communication avec le demandeur".
La première étape va être de vérifier si un client se connecte à notre serveur. On va donc retrouver notre objet EthernetClient et une fonction du serveur : available().

```
// Attente de la connexion d'un client
EthernetClient client = server.available();
```

Ensuite les choix sont simples. Soit un client (donc une application externe) est connecté avec l'Arduino et veut interagir, soit il n'y a personne et donc on ne fait… rien (ou autre chose). Pour cela, on va simplement regarder si client vaut autre chose que zéro. Si c'est le cas, alors on traite les données.

```
if (client && client.connected()) {
                Serial.println("Client connecté ...");
                // Traitement
}
```

Voici un exemple de traitement, le client envoie un caractère :

```
// si le client nous envoie quelque chose
 if (client.available() > 0) {

  // On regarde ce que le client nous demande
  switch(client.read()){
  case 'A': // allumer la led
    Serial.println("allumer la led A ...");
    break;
  case 'a': // éteindre la led
    Serial.println("Eteindre la led ...");
    break;
  }
 }
```

En résumé, on aura donc le code suivant pour la boucle :

```
void loop()
{
 // Attente de la connexion d'un client
 EthernetClient client = server.available();
 if (client && client.connected()) {
 Serial.println("Client connecté ...");
  // si le client nous envoie quelque chose
  if (client.available() > 0) {

   // On regarde ce que le client nous demande
   switch(client.read()){
   case 'A': // allumer la led
     Serial.println("allumer la led A ...");
     break;
   case 'a': // éteindre la led
     Serial.println("Eteindre la led ...");
     break;
   }
  } } }
```

Voici un autre exemple de traitement, le client envoie une chaine de caractères :

```
// Variable de réception d'une chaine de caractère
String msg ;

void loop()
{
 // Attente de la connexion d'un client
 EthernetClient client = server.available();
 if (client) {
            msg = "";   // Remise à zero de la variable de réception
            Serial.println("Client connecté ...");
            while(client.connected()) {
                        // si le client nous envoie quelque chose
                        if (client.available() > 0) {
                                    char carlu = client.read(); //on lit ce qu'il raconte
                                    msg+= carlu;
                                    if(carlu == '\n') break;
                        }
                        else
                                    break;
            }
    // Message reçu
    Serial.println("Message : "+ msg);
    // On regarde ce que le client nous demande

     if(msg == "AA"){
      // allumer la led
      Serial.println("Allumer la led A ...");
      client.println("Allumer la led A ...");          // Message envoyé au client
     }
     else if (msg == "aa"){
      Serial.println("Eteindre la led ...");
      client.println("Eteindre la led A ...");          // Message envoyé au client
     }
     else {
      Serial.println("Erreur code ...");
      client.println("Erreur code ...");                 // Message envoyé au client
     }
  }
 }
```

Et voilà !

Il est maintenant temps de tester. Branchez votre Arduino, connectez-la au réseau.

Lancez KawaClient réalisé avec Mr Breysse.
Configurez votre client avec ip:port que vous avez paramétré (en l'occurrence : 192.168.1.20:2000/ pour moi). Si tout se passe bien, vous devez pouvoir communiquer avec le serveur.

FIN