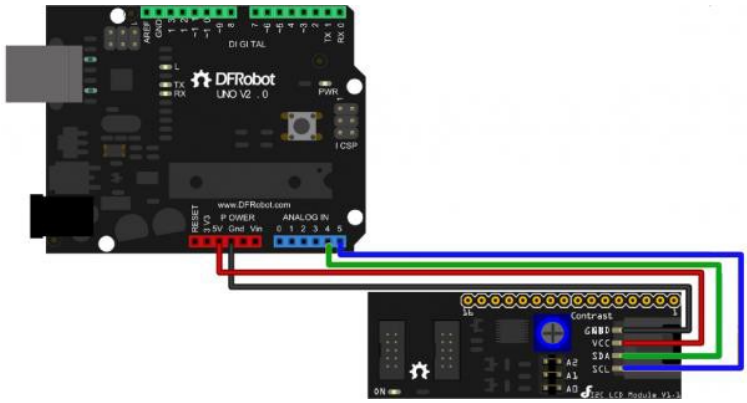
Ce programme va vous permettre de réaliser une communication en UDP entre le SmartPhone et l'environnement Arduino

Pour que la carte Arduino soit connectée au réseau un shield Ehernet doit être utilisé (n'ayant pas de shield Wifi…) et ce dernier est relié en RJ45 sur le routeur Wifi.

Afin de visualiser la réception des messages UDP côté Arduino, un écran LCD 16*2 I2C doit être câblé de la manière suivante (attention les couleurs des fils ne correspondent pas) :



L'adresse I2C du shield est définie par trois cavaliers se trouvant au dos de l'écran, et respectant l'adressage suivant :

A2	A1	A0	Adresse	
0	0	0	0x20	• **0: The Jumper Cap is connected**
0	0	1	0x21	
0	1	0	0x22	
0	1	1	0x23	
1	0	0	0x24	• **1: The Jumper Cap is disconnected**
1	0	1	0x25	
1	1	0	0x26	
1	1	1	0x27	

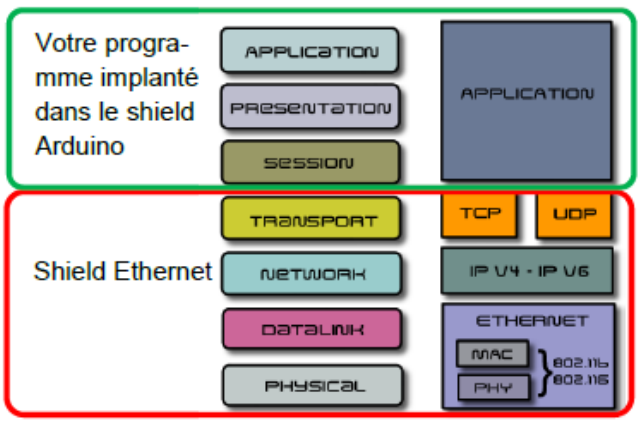
Par défaut ces trois cavaliers ne sont pas en place, l'adresse est donc 0x27

Présentation du shield Ethernet

Le Shield est essentiellement composé d'un module Wiznet W5100 qui contient un contrôleur Ethernet associé à une pile TCP/IP "matérielle", pouvant gérer 4 sockets simultanément.

Au niveau OSI il gère entièrement les couches entourées dans le schéma ci-contre :

L'utilisation du shield impose d'inclure dans votre projet la librairie Ethernet.



L'objet TCP_IP permet de communiquer sur un réseau Ethernet en utilisant le protocole TCP ou UDP. Les possibilités sont donc les suivantes :

- Protocole UDP
- Protocole TCP et shield en mode serveur
- Protocole TCP et shield en mode client. (Ce mode est rarement utilisé, il ne sera pas étudié)

Réception sur l'Arduino de message UDP

1. Le programme Arduino est dans le dossier Ressources – Arduino – UDP1. Après l'avoir copié dans votre dossier personnel, ouvrez-le.
2. Avec la version 1.7.8 et le shield Ethernet 1, les librairies posent des problèmes. Aussi on n'a gardé que les inclusions :

```
#include <Ethernet.h>

#include <EthernetUdp.h>
```

3. Pour dialoguer avec le shield Ethernet votre programme doit aussi utiliser la librairie SPI.
4. La méthode permettant de configurer le shield est Ethernet.begin dont la syntaxe est la suivante : 4 formes sont possibles, les 2 derniers paramètres étant optionnels :

Ethernet.begin(mac); // mode DHCP

Ethernet.begin(mac, ip); // mode ipfixe

Ethernet.begin(mac, ip, dns); // mode ipfixe + serveur dns

Ethernet.begin(mac, ip, dns, gateway); // ipfixe + serveur dns + passerelle

Ethernet.begin(mac, ip, dns, gateway, subnet); // ipfixe + serveur dns + passerelle + masque sous-réseau

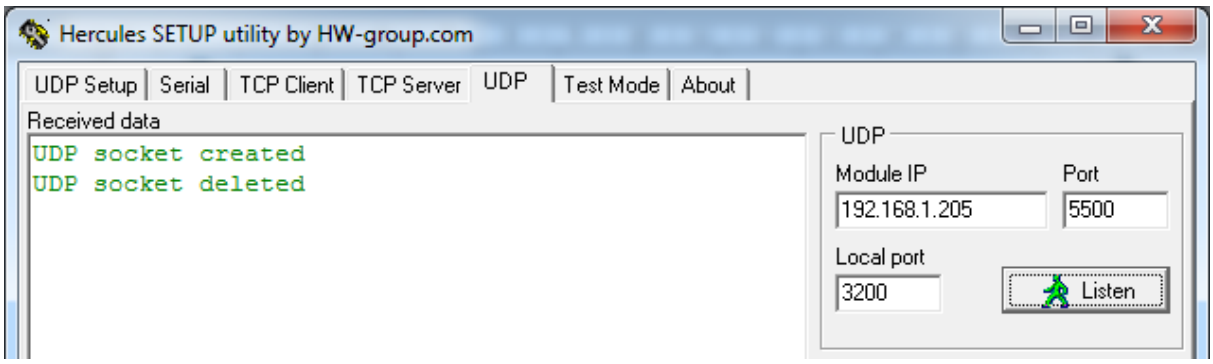
- *Remarque : L'adresse MAC est positionnable de façon logicielle. La ligne :*
- ***IPAddress ip_shield(192,168,1,205)** permet de créer une variable nommée **ip_shield** de type adresse IP.*

5. La réception se passe en plusieurs étapes :
   - On crée un objet de type EthernetUDP -> 1
   - On ouvre le socket UDP entrant (port 5500) -> 2 ;
   - On lit la taille des octets présents dans le buffer -> 3 ;
   - Si celle-ci est différente de zéro, on lit le contenu du buffer -> 4.

UDP1 §

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

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

#include <SPI.h>         // needed for Arduino versions later than 0018

LiquidCrystal_I2C lcd(0x27,16,2);
byte mac[]={0x90,0xA2,0xDA,0x0D,0x81,0x40};
IPAddress ip_shield(192,168,1,205);

EthernetUDP UDP;

void setup()
{
  lcd.init();
  lcd.backlight();
  lcd.print("Hello, world!");
  Ethernet.begin(mac,ip_shield);

  UDP.begin(5500);
}

void loop()
{
  int Size=UDP.parsePacket();
  if (Size>0)
  {
    char message[Size];
    UDP.read(message,Size);
    lcd.setCursor(0,1);
    lcd.print("                ");
    lcd.setCursor(0,1);
    lcd.print(message);
  }
}
```

6. Programmez l'Arduino.
7. Depuis votre poste faites un « ping » sur l'adresse IP du shield afin qu'il vous réponde.
8. Pour le tester, vous pouvez utiliser Hercules en le configurant correctement :

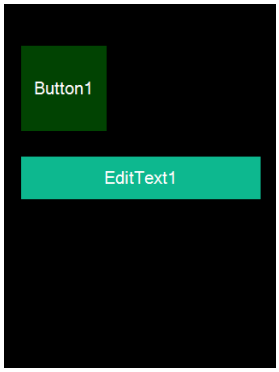


Le message saisi dans la zone Send d'Hercules doit apparaitre sur l'écran LCD de l'Arduino (à condition d'avoir modifié l'adresse 192.168.1.205 avec celle de votre shield).

Emission depuis le SmartPhone de message UDP

Le programme en Basic4Android va pour le moment être des plus simple. Vous allez utiliser la librairie UDP (il faut aussi la NetWork). Votre application proposera :

- Un bouton permettant l'envoi du message ;
- Un EditText pour saisir le message à envoyer vers l'Arduino ;



9. Lancez B4A et créez un nouveau programme : "programme UDPArduino"
10. Créez la feuille du programme avec l'outil Designer
11. Générez les membres : Button1, Button1_Click, EditText1
12. Cochez l'utilisation des librairies : NetWork, UDP et ByteConverter
13. Le programme va se limiter à l'initialisation du protocole UDP en ne définissant, pour le moment aucun port entrant.

```
16
17  Sub Globals
18      Dim button1 As Button
19      Dim EditText1 As EditText
20  End Sub
21
22  Sub Activity_Create(FirstTime As Boolean)
23      Activity.LoadLayout("layout1")
24      UDP.Initialise(0)
```

14. Puis suite à l'appui sur le Button1 à l'envoi sur le socket UDP de l'Arduino le texte saisi dans l'EditText1 (attention de bien placer l'adresse IP de votre shield Arduino).

```
28
29  Sub button1_click
30      Dim emission() As Byte
31      Dim bc As ByteConverter
32      Dim message As String
33      message=EditText1.Text
34      emission=bc.StringToBytes(message,"ASCII")
35      UDP.emission("192.168.1.205",5500,emission)
36  End Sub
```

15. Programmez votre SmartPhone et testez votre programme

Réponse en UDP de l'Arduino

Vous allez modifier vos deux programmes (celui sur l'Arduino et celui sur B4A) pour que lorsque l'Arduino reçoit un message UDP sur son port entrant (5500), il l'affiche sur l'écran LCD, puis il en retourne un autre sur son port sortant (3200) afin de l'afficher sur le SmartPhone.

16. Sur B4A vous allez rajouter avec le Designer un Label afin de pouvoir y écrire les messages reçus.

17. Il faut ensuite définir le port UDP sortant du SmartPhone :

```
16
17  Sub Globals
18      Dim button1 As Button
19      Dim EditText1 As EditText
20      Dim Label1 As Label
21  End Sub
22
23  Sub Activity_Create(FirstTime As Boolean)
24      Activity.LoadLayout("layout1")
25      UDP.Initialise(3200)
26
27  End Sub
```

18. Puis créez la fonction gérant le flux entrant :

```
38  Sub UDP_PacketArrived (packet As UDPPacket)
39
40      Dim message As String
41      message=UDP.reception(packet)
42      Label1.Text=message
43
44  End Sub
```

19. Programmez le SmartPhone

20. Sur l'Arduino ouvrez le fichier UDP2. Les lignes suivantes ont été rajoutées :

```
void loop()
{
  int Size=UDP.parsePacket();
  if (Size>0)
  {
    char message[Size];
    UDP.read(message,Size);

    lcd.setCursor(0,1);
    lcd.print("                ");
    lcd.setCursor(0,1);
    lcd.print(message);


    UDP.beginPacket(UDP.remoteIP(), UDP.remotePort());
    UDP.print(Size);
    UDP.print (" octet(s) transmis");
    UDP.endPacket();
    delay(5000);
   }
```

On débute l'écriture du paquet à envoyer en précisant l'adresse IP du destinataire et le port sortant ; ici on utilise les méthodes remoteIP() et remotePort() qui retournent l'adresse IP et le port Entrant du SmartPhone.

L'écriture des données est effectuée dans le paquet ;

On finalise l'écriture du paquet.

× *Remarque : Dans le programme précédent, votre shield reçoit des données qu'il affiche à son tour sur l'écran LCD. S'il veut répondre encore faut-il qu'il sache qui lui a adressé le message. Pour cela la bibliothèque propose deux méthodes permettant dans le paquet UDP reçu d'extraire l'adresse IP et le port de l'expéditeur :* ***remoteIP()*** *et* ***remotePort().***

21. Programmez l'Arduino.

22. Vérifiez le bon fonctionnement de l'ensemble.

Pour terminer, je vous propose un petit programme pour l'Arduino vous permettant de manipuler les variables liées à la gestion du protocole.

23. Ouvrez le programme Arduino « UDP3.ino » qui se trouve dans le dossier ***Ressource – UDP3***.

24. Programmez l'Arduino (il faudra peut-être changer l'adresse IP pour positionner la vôtre).

25. Le programme pour le SmartPhone ne change pas. Vérifiez le bon fonctionnement de l'ensemble.

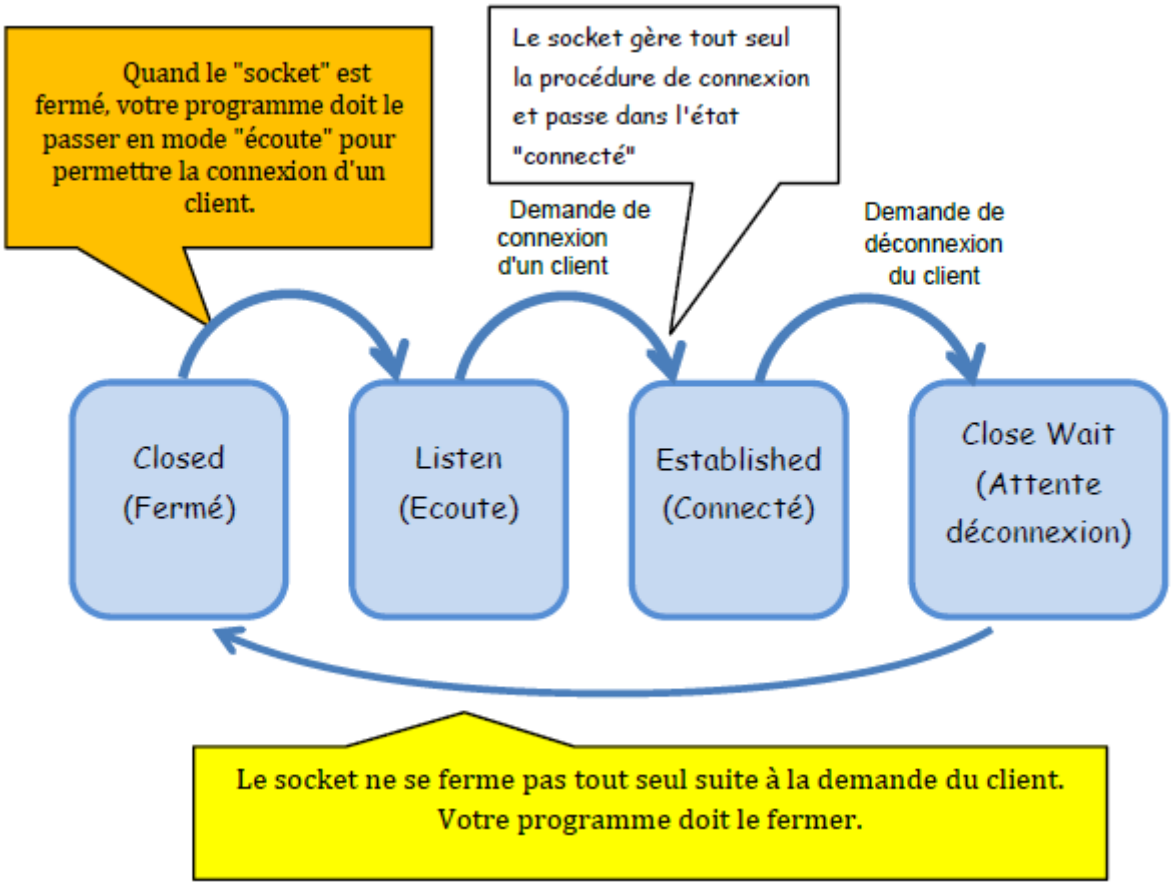
Utilisation du shield Arduino en mode TCP serveur

Le protocole TCP est un protocole "orienté connexion" avec la notion de Client/Serveur. Dans cette étude l'arduino sera utilisé en Serveur.

Cela implique quelques différences d'utilisation du "socket" TCP par rapport au "socket" UDP.

En mode "TCP serveur" le résumé du fonctionnement est le suivant :

- Il faut bien sûr commencer par créer un "socket" TCP (Par défaut le socket est "fermé" : Aucun client ne peut se "connecter")
- Ensuite il faut placer le "socket"en mode "écoute". (A partir de cet instant, un client peut se connecter)
- Quand un client est connecté, on peut échanger des données avec lui.
- Quand le client ne désire plus échanger de données, il fait une demande de déconnexion, à ce moment-là le serveur doit "fermer" le "socket" (pour terminer le processus de déconnexion) et ensuite replacer le socket en mode "écoute" pour permettre à nouveau la connexion d'un client…



26. Le programme Arduino complet est dans le dossier Ressources – Arduino – TCPserveur.

```
void setup()
{
  Ethernet.begin(mac, ip_shield);
  server.begin();
  lcd.init();
  lcd.backlight();
  lcd.print("Server started");
  delay(3000);
}

void loop()
{
    lcd.home();
  lcd.print("Attente client");

  EthernetClient client = server.available();
```

le socket est créé, le serveur est démarré ;

Le serveur est en mode « écoute » ;

```
  if (client)
    {
     while (client.connected()==true)
       {

          client.flush();
          message="";
          change=false;
                while (client.available() > 0)
                {
                  char c = client.read();
                  message+=c;
                  change=true;
                }

               if (change == true) {
                  lcd.clear();
                  lcd.print("Client connecte");
                  lcd.setCursor(0,1);
                  lcd.print(message);
                  client.print("Message recu :"+message);

                  }
       }
        lcd.clear();
    }
}
```

Le client est connecté suite à l'envoi de données, on les affiche sur l'écran LCD;

27. Programmez l'Arduino.

- Ouvrez sur votre ordinateur le logiciel Hercule onglet TCP Client et configurez-le.
- Dans la zone *Send* saisissez un message et cliquez sur le bouton *Send* ; le message doit s'afficher sur l'écran LCD.
- Fermez puis ré-ouvrez la connexion dans Hercule et constatez que l'émission reste possible.

28. Côté B4A vous pouvez reprendre le TCP avec les librairies.

```
36
37  Sub Activity_Create(FirstTime As Boolean)
38      Activity.LoadLayout("layout1")
39      Label1.text="Non connecté"
40      Connect.Visible=True
41      deconnect.Visible=False
42      Send.Visible=False
43      EditText1.Visible=False
44  End Sub
45  Sub TCP_client_Connected(Succes As Boolean)
46      Label1.text="Connecté"
47      Connect.Visible=False
48      deconnect.Visible=True
49      Send.Visible=True
50      EditText1.Visible=True
51  End Sub
52
53  Sub Send_Click
54      Dim emission() As Byte
55      Dim bc As ByteConverter
56      Dim message As String
57      message=EditText1.Text
58      emission=bc.StringToBytes(message,"ASCII")
59      TCP_client.emission(emission)
60  End Sub
61
62  Sub Connect_Click
63      TCP_client.connexion("192.168.1.205",5500)
64  End Sub
65
66  Sub deconnect_Click
67      TCP_client.fin_connexion
68      Label1.text="Non connecté"
69      Connect.Visible=True
70      deconnect.Visible=False
71      Send.Visible=False
72      EditText1.Visible=False
73  End Sub
74
75  Sub TCP_client_newdata(buffer()As Byte)
76      Dim msg As String
77      msg=BytesToString(buffer,0,buffer.length,"ascii")
78      Label2.Text=msg
79  End Sub
80
81
```