Montage

17 Communication réseau

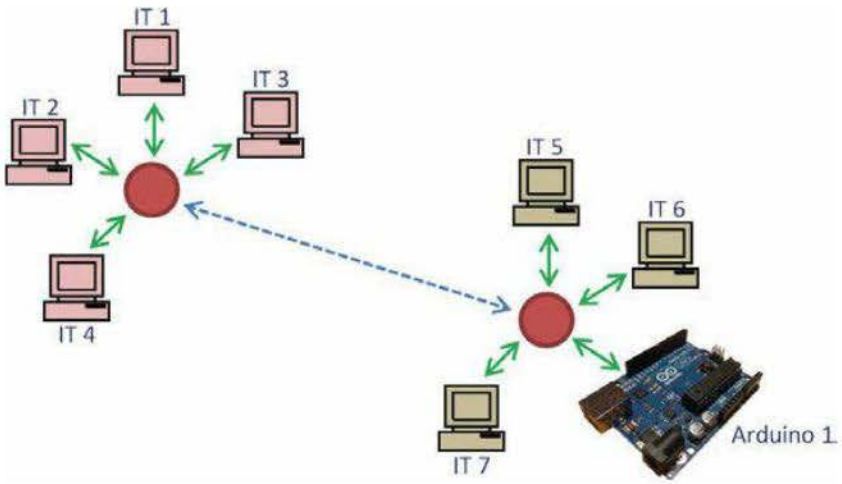
Au sommaire :

- savoir ce qu'est un réseau ;
- apprendre à incorporer la carte Arduino dans un réseau ;
- savoir ce qu'est un serveur web ;
- un exercice complémentaire.

Qu'est-ce qu'un réseau ?

Le plus gros réseau que l'Homme utilise quotidiennement est le *World Wide Web* ou www sous sa forme abrégée. Il s'agit de l'interconnexion d'une multitude de systèmes informatiques en contact l'un avec l'autre dans le monde entier. On parle de réseau dès l'instant où deux ordinateurs sont associés via un support de transmission approprié (par exemple : câble Ethernet, fibre optique ou Wlan). Vous pouvez l'imaginer comme un cerveau contenant plusieurs centaines de milliards de cellules nerveuses. Chacune d'elles possède jusqu'à dix mille synapses. Ce sont des voies de communication qu'elles utilisent pour transmettre ou échanger des informations. Chaque cellule nerveuse du cerveau figure un ordinateur en contact avec d'autres systèmes au moyen des synapses – donc de sa carte réseau (ou de ses cartes le cas échéant).

Figure 17-1 ▶
Petit réseau avec carte Arduino



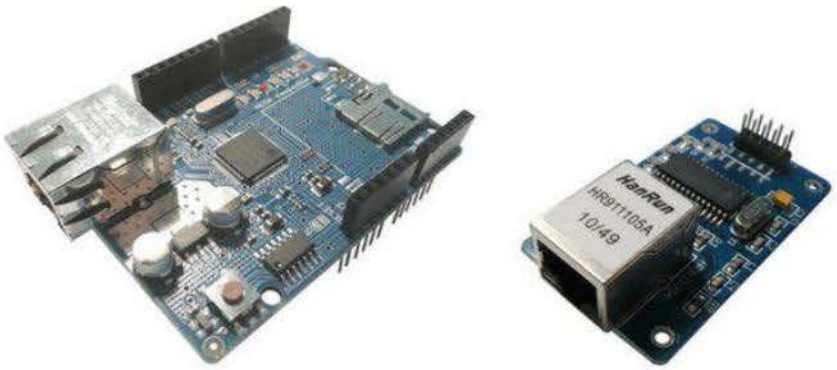
Les différents systèmes informatiques, que j'ai appelé IT1 à IT7 dans la figure 17-1 pour plus de commodité, sont reliés entre eux au moyen des cartes ou plutôt des câbles de réseau. Cette représentation est bien sûr simplifiée car les composants du réseau sont par exemple reliés par des *switches* dans la réalité. Ces répartiteurs ou coupleurs de réseau transmettent les données de manière intelligente aux différents utilisateurs. La figure 17-2 montre un connecteur de type RJ45 d'un câble de réseau couramment utilisé.

Figure 17-2 ▶
Connecteur RJ45 d'un câble de réseau

Je pense que vous avez déjà vu une fiche de cette sorte puisque votre ordinateur est relié à coup sûr par un câble de réseau au routeur qui établit une liaison vers votre fournisseur d'accès, autrement dit vers Internet.

Je ne vois pas de prise femelle pour cette fiche sur ma carte Arduino. Comment fait-on pour la connecter au réseau ?

Vous allez ici encore plus vite que la musique, Ardus. Mon introduction n'est même pas terminée. La carte Arduino ne dispose évidemment pas d'une connexion réseau. Un composant réseau supplémentaire est donc nécessaire.



Shield Ethernet — Module Ethernet ENC28J60

Figure 17-3
Deux composants Ethernet

La figure 17-3 montre à gauche un shield Ethernet, qui dispose en plus d'un socle microSD. Vous pouvez y stocker temporairement des données, mais là n'est pas le sujet. À droite se trouve un module Ethernet ENC28J60. Il est certes préférable au shield Ethernet, mais ne permet pas de stocker des données sur une carte SD et ne peut être branché directement sur la carte Arduino. Des cordons de raccordement doivent être utilisés pour relier le module à votre carte Arduino.

Vous avez déjà utilisé plusieurs fois le mot Ethernet. De quoi s'agit-il au juste ? Ça doit avoir quelque chose à voir avec Internet ou le réseau, n'est-ce pas ?

C'est vrai, Ardus ! Et c'est une belle opportunité pour aborder certains points spécifiques aux réseaux.

Ethernet

Le mot Ethernet qualifie une technologie câblée pour transmettre des données. Depuis les années 1990, elle est la norme pour toute une gamme de technologies LAN *(Local Area Network)*. Le transfert des données est, en principe, assuré par un câble à paire torsadée *(Twisted-Pair-Cable)* selon la norme CAT-5 ou supérieure.

TCP/IP

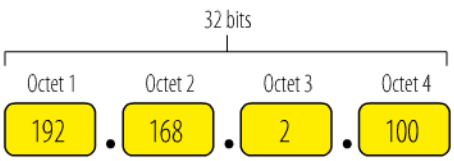
Ethernet utilise un protocole appelé TCP (*Transfer Control Protocol*, protocole de contrôle de transfert en français) pour transmettre des données. Ce protocole permet de transférer des informations au moyen d'un réseau local ou global et garantit une communication sans erreurs. Des mécanismes permettent, en cas d'erreur de données menaçante, de corriger ou de retransmettre les paquets de données à

transférer. La désignation IP *(Internet Protocol)* concerne l'adressage des paquets de données à transférer qui doivent être acheminés de l'émetteur à un destinataire bien défini. Ce protocole assure donc l'adressage des paquets de données à transmettre. Chaque utilisateur du réseau possède une adresse précise, comparable au numéro de maison dans une rue. Pour qu'un colis puisse être par exemple livré à coup sûr par la Poste, les numéros des maisons ne doivent pas être en double, ce qui est le cas normalement. L'IP est toujours indiqué ou utilisé en rapport avec le TCP.

Adresse IP

L'adresse IP d'un utilisateur doit satisfaire à l'exigence d'univocité dans un réseau. Elle est affectée à un appareil qui fait partie du réseau, garantissant ainsi qu'il est adressable et accessible. Les adresses IP de la notation Ipv4 sont composées de quatre octets (32 bits).

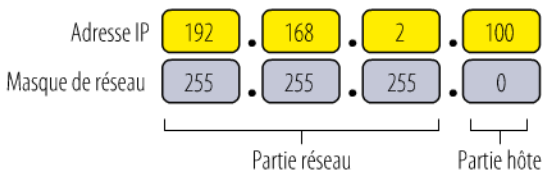
Figure 17-4 ▶
Capture d'écran



Cette adresse a été attribuée par mon routeur à mon ordinateur, afin que je sois disponible sur le réseau.

Masque de réseau

Une adresse IP comprend toujours une partie réseau et une partie hôte. Le masque de réseau définit quant à lui combien d'appareils doivent être atteints dans un réseau et lesquels se trouvent dans d'autres réseaux.



Pour parvenir à la partie hôte, l'adresse IP est combinée au masque de réseau par une opération ET. D'après le masque ci-dessus, il est théoriquement possible d'avoir $2^8 = 256$ ordinateurs dans le réseau. Je dis bien théoriquement car 255, par exemple, a une signification particulière. Des détails supplémentaires sortiraient du cadre de ce livre,

c'est pourquoi je vous invite à consulter la littérature spécialisée ou à regarder sur Internet.

Adresse MAC

L'adresse MAC *(Media Access Control)* doit être sans ambiguïté à l'échelle mondiale. Elle a été attribuée à chaque adaptateur de réseau. Elle se compose de six octets, les trois premiers contenant un code fabricant OUI *(Organizational Unit Identifier)*. Les trois autres octets contiennent l'indicatif de l'appareil, assigné par le fabricant en question. Voici un exemple d'adresse MAC pour une carte réseau :

1C-6F-65-94-D5-1A

Passerelle

Une passerelle (*gateway*, en anglais) est un passage vers une zone particulière qui, rapporté à notre thématique, peut être appelé passerelle de réseau. De quel appareil pourrait-il s'agir ? Le routeur, qui se trouve à moitié sur Internet, passe pour être une passerelle. Mon routeur a par exemple `192.168.2.1` comme adresse IP et transmet mes demandes à mon fournisseur d'accès, c'est-à-dire sur Internet. Si vous ouvrez une console DOS et que vous écrivez la commande `ipconfig /all`, vous obtenez, entre autres, les indications suivantes :

Standardgateway . . . . . : 192.168.2.1
DHCP-Server . . . . . : 192.168.2.1

La figure 17-5 montre le shield Ethernet combiné à vote carte Arduino.

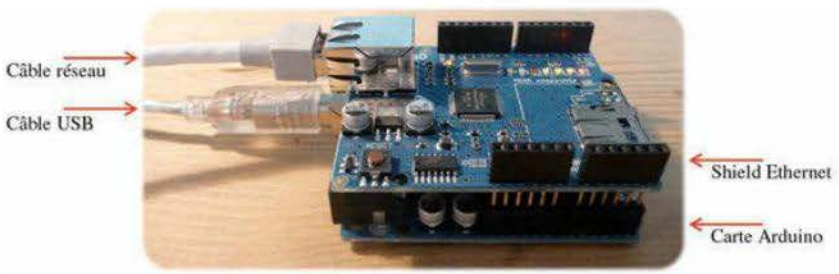


Figure 17-5
Shield Ethernet et carte Arduino

Composants nécessaires

1 shield Ethernet (*http://arduino.cc/en/Guide/ArduinoEthernetShied*)

1 câble réseau, suffisamment long pour aller du routeur au shield Ethernet

1 shield d'entrée analogique

Attention !

Utilisez un câble normal pour relier votre shield Ethernet à votre routeur. Ces câbles sont jaunes, blancs ou même noirs. Ne vous servez pas d'un câble réseau rouge entre votre routeur et le shield Ethernet, car il s'agit généralement d'un câble croisé qui ne doit être employé que pour relier votre shield directement à la carte réseau de votre ordinateur. Les lignes de réception et d'émission sont alors croisées. Vous trouverez des informations plus précises sur Internet.

Le sketch suivant permet d'utiliser le shield Ethernet comme un serveur Web. Quand vous passez par un navigateur web (tel Firefox, Opera ou IE) pour vous connecter à Internet, vous établissez une liaison avec un serveur web (voir figure 17-6).

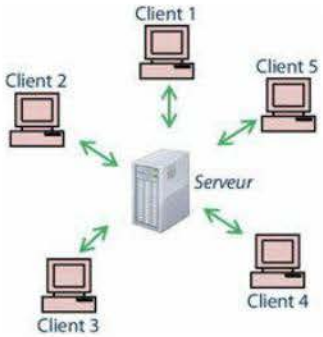


Figure 17-6
Shield Ethernet et carte Arduino

La figure 17-6 montre un serveur (fournisseur) au centre, qui répond aux requêtes de nombreux clients (utilisateurs). Un serveur est un logiciel qui réagit à une demande de contact venant de l'extérieur et délivre des informations. Il peut s'agir d'un serveur mail ou FTP ou encore d'un serveur web. Un client peut être un client mail, tel que Thunderbird ou Outlook. S'il s'agit d'un client web, cela peut être Firefox, Opera ou IE, tous déjà mentionnés dans ce livre. Prenons maintenant un exemple concret, dans lequel le shield Ethernet, en tant que serveur web, doit envoyer les valeurs des entrées analogiques de la carte Arduino. La figure 17-7 offre un aperçu de l'affichage dans le navigateur web.

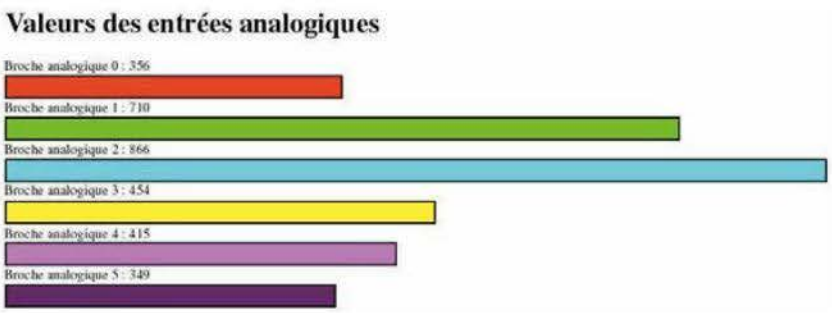


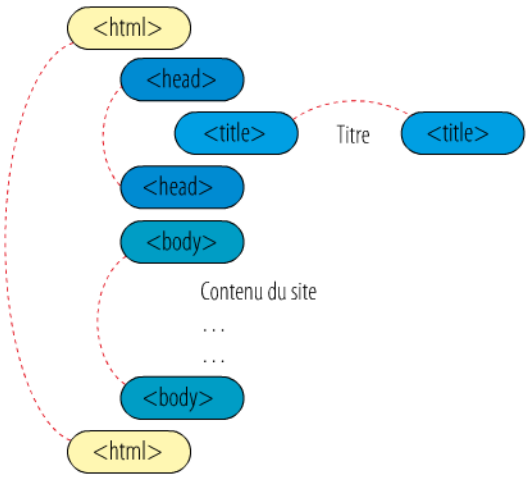
Figure 17-7
Affichage de la page HTML dans le navigateur web (représentation numérique et graphique)

Vous n'êtes pas sérieux ! Dois-je en plus apprendre à programmer un site Internet ?

Eh oui, Ardus ! On ne peut pas faire autrement, mais soyez rassuré. Nous n'allons qu'effleurer le sujet car celui-ci pourrait sans peine remplir toute une bibliothèque. Les sites Internet sont programmés en HTML *(Hypertext Markup Language)*. Il s'agit d'un langage de balisage à base de texte permettant par exemple de représenter du texte, des images, des vidéos ou des liens sur un site Internet, et de lire et afficher ceux du navigateur web. Vous trouverez ci-après la trame de

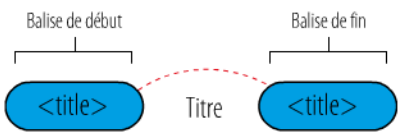
base d'un site, que nous remplirons par la suite pour présenter nos informations. La plupart des éléments HTML sont identifiés par des paires de balises *(tags)*. Il y a toujours une balise de début (ouvrante) et une balise de fin (fermante). La figure 17-8 montre la trame de base en question, les paires correspondantes étant indiquées en couleurs.

Figure 17-8 ▶
Trame de base d'un site Internet



Les lignes pointillées en rouge vous permettent de voir les formations de paires. Les différentes balises ou éléments HTML sont constituées par les noms des éléments entre chevrons. Voyons maintenant une paire de balises de plus près :

Figure 17-9 ▶
La paire de balises `title`



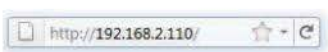
Cette paire génère le titre du site Internet, le texte se trouvant entre la balise de début et la balise de fin. La balise de fin présente le même nom d'élément que la balise de début, cependant il est précédé d'une barre oblique (appelée *slash*).

Code du sketch

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

byte MACAddress[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
                                          // Adresse MAC
byte IPAddress[] = {192, 168, 2, 110}; // Adresse IP
int const HTTPPORT = 80;                  // Port HTTP 80 (port standard)
```

```
String barColor[] = {"ff0000", "00ff00", "0000ff",
                     "ffff00", "ff00ff", "550055"};
                                  // Couleurs RGB pour barres de couleur
#define HTML_TOP      "<html>\n<head><title>Arduino Web-Server</title>
            </head>\n<body>"
#define HTML_BOTTOM "</body>\n</html>"
EthernetServer myServer(HTTPPORT); // Démarrage du serveur web sur
                                   //le port indiqué

void setup(){
   Ethernet.begin(MACAddress, IPAddress); //Initialisation Ethernet
   myServer.begin();                      // Démarrage du serveur
}

void loop(){
  EthernetClient myClient = myServer.available();
  if(myClient){
     myClient.println("HTTP/1.1 200 OK");
     myClient.println("Content-Type: text/html");
     myClient.println();

      myClient.println(HTML_TOP);      // HTML début
      showValues(myClient);            // Contenu HTML
      myClient.println(HTML_BOTTOM); // HTML fin
   }
   delay(1);               // Courte pause pour navigateur Web
   myClient.stop();        // Fermeture connexion client
}

void showValues(EthernetClient &myClient){
  for(int i = 0; i < 6; i++){
    myClient.print("Analog Pin");
    myClient.print(i);
    myClient.print(": ");
    myClient.print(analogRead(i));
    myClient.print("<div style\"height: 15px; background-color: #");
       myClient.print(barColor[i]);
       myClient.print("; width: ");
       myClient.print(analogRead(i));
       myClient.println("px; border: 2px solid;\"></div>");
   }
}
```

Pour accéder au serveur web Arduino, écrivez l'adresse IP du code de sketch dans la ligne d'adresse de votre navigateur web Arduino. Dans mon cas, l'adresse est la suivante.



Si cette adresse vous semble trop énigmatique, vous pouvez bien sûr en choisir une plus parlante comme :

Il vous suffit d'adapter dans Windows le fichier `hosts` avec des droits d'administrateur sous `C:\Windows\System32\drivers\etc` et d'ajouter la ligne dans laquelle j'ai indiqué le nom `Arduino` :

```
# localhost name resolution is handled within DNS itself
#    127.0.0.1       localhost
#    ::1             localhost
192.168.2.110        Arduino
```

L'appel est alors plus simple et vous n'avez pas besoin de retenir l'adresse IP.

Revue de code

Du point de vue logiciel, les variables suivantes sont nécessaires à notre serveur web expérimental.

Tableau 17-1 ▶
Variables nécessaires et leur objet

Variable	Objet
`MACAddress[]`	Tableau unidimensionnel pour stocker l'adresse MAC du shield Ethernet
`IPAddress[]`	Tableau unidimensionnel pour stocker l'adresse IP du shield Ethernet
`HTTPPORT`	Variable pour stocker l'adresse du port HTML
`BarColor[]`	Tableau unidimensionnel pour stocker les informations de couleurs des barres horizontales
`HTML_TOP`	Équivalent du code HTML pour la partie supérieure (en-tête)
`HTML_BOTTOM`	Équivalent du code HTML pour la partie inférieure (fin)

Deux bibliothèques doivent être incorporées pour pouvoir utiliser la fonctionnalité du shield Ethernet :

- `SPI.h` : Serial-Peripheral-Interface-Bus est nécessaire pour les versions Aduino > 0018 ;
- `Ethernet.h`.

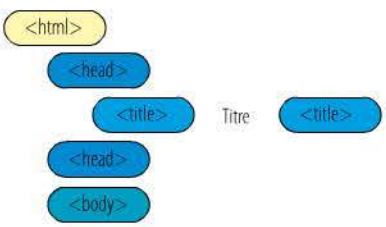
Je voudrais vous poser une question au sujet de la variable HTTPPORT. N'est-ce pas une faute de frappe ? Ne s'agit-il pas de HTMLPORT ? Je croyais qu'il était question ici de pages HTML.

C'est vrai Ardus qu'on s'y perd un peu au début. HTTP est la forme abrégée de *Hypertext Transfer protocol*. Comme vous l'avez peut-être remarqué, on a affaire à un grand nombre de protocoles différents en informatique. Quand il s'agit de pages web, ce protocole est chargé de la transmission. Quand vous tapez une adresse web dans votre navigateur, celle-ci commence la plupart du temps par `http://` et non par `html://`. Passons maintenant à la définition du port. Le port standard pour des serveurs web qui utilisent le protocole HTTP est le numéro 80. Imaginez-vous ce numéro comme une sorte de bifurcation sur la route du réseau, où d'autres protocoles circulent encore. Voici une courte liste d'applications dont vous avez peut-être déjà entendu parler.

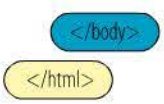
Tableau 17-2
Courte liste avec numéros de port et services

Port	Service	Rôle
21	FTP	Transfert de fichier via FTP-Client
25	SMTP	Envoi d'e-mails
110	POP3	Accès client à un serveur e-mails

Je voudrais encore vous parler brièvement de la structure d'une page HTML. La seule partie variable de notre page est la partie que j'ai appelée *Contenu de ma page*. Tout ce qui est au-dessus ou en dessous ne change pas. C'est pour cette raison que j'ai créé les raccourcis pour la partie supérieure :



dans la définition `HTML_TOP` et pour la partie inférieure :



dans `HTML_BOTTOM`. Vous retrouverez la même chose dans le sketch, avec les lignes suivantes :

```
#define HTML_TOP      "<html>\n<head><title>Arduino Web-Server
                      </title></head>\n<body>"
#define HTML_BOTTOM   "</body>\n</html>"
```

La séquence d'échappement `\n` provoque un saut de ligne, de telle sorte que le code HTML soit formaté d'une certaine manière et que tout ne soit pas mis sur une seule ligne. Venons-en maintenant au déroulement du sketch proprement dit. Diverses parties du programme sont comme toujours initialisées dans la fonction `setup`.

```
void setup(){
  Ethernet.begin(MACAddress, IPAddress); // Initialisation Ethernet
  myServer.begin();                      // Démarrage du serveur
}
```

La première étape consiste à doter le shield Ethernet de l'adresse MAC et d'une adresse IP unique.

Dites-moi s'il vous plaît d'où vous sortez l'adresse IP `192.168.2.110` en question. Ça reste un mystère pour moi.

Eh bien Ardus, c'est tout simple ! Mon routeur se trouve dans la zone d'adresse `192.168.2` et l'adresse d'hôte `1` lui est attribuée, autrement dit son adresse IP est `192.168.2.1`. Je peux donc affecter des adresses comprises entre `192.168.2.2` et `192.168.2.254` à d'autres utilisateurs du réseau. Revenons à l'initialisation. La deuxième étape consiste à démarrer le serveur web, de sorte qu'il puisse réagir à des demandes entrantes. Celui-ci épie le réseau et reste sur le qui-vive jusqu'à ce qu'un client l'aborde et lui demande quelque chose.

Il accomplit ensuite son travail et délivre les données avant de se remettre à nouveau en position d'attente. Passons maintenant au traitement proprement dit dans la fonction `loop`. La présence de la demande d'un client est d'abord vérifiée :

```
EthernetClient myClient = myServer.available();
if(myClient){
```

Si l'interrogation `if` est satisfaite, le serveur peut commencer à envoyer ses informations au client.

C'était quoi déjà l'interrogation `if` ? On y lit `myClient` au lieu d'une expression à évaluer.

Pas de problème, Ardus ! Ce n'est que la forme abrégée du code suivant :

```
if(myClient == true){…}
```

L'interrogation sur `true` est facultative du fait que si l'expression est vraie dans l'instruction `if`, c'est le bloc subséquent qui est exécuté. Vous n'avez pas besoin de vérifier à nouveau avec `==` que l'expression est vraie. Vous comprenez maintenant ? Donc, si un client a effectué une demande auprès du serveur, ce dernier renvoie pour commencer les lignes suivantes :

```
myClient.println("HTTP/1.1 200 OK");
myClient.println("Content-Type: text/html");
myClient.println();
```

Dans la première ligne, le serveur confirme la demande du client en transmettant la version `1.1` du protocole HTTP, suivie du code d'état `200` indiquant que la demande a été traitée avec succès et que le résultat de la demande est transmis dans la réponse. Dans la deuxième ligne, le *type MIME*, `text/html` dans notre cas, est communiqué. Celui-ci décrit le genre des données envoyées par le serveur. S'agit-il d'informations purement textuelles ou une image est-elle éventuellement délivrée au client ? Les données transmises doivent alors être bien sûr interprétées en conséquence et non pas affichées en texte clair. Passons maintenant au code, qui envoie les données lues de votre carte Arduino :

```
myClient.println(HTML_TOP);       // HTML début
showValues(myClient);             // Contenu HTML
myClient.println(HTML_BOTTOM);    // HTML fin
```

Les tâches de `HTML_TOP` et `HTML_BOTTOM` vous sont connues. L'appel des données de la carte est exécuté par la fonction `showValues` que je vous redonne ici :

```
void showValues(EthernetClient &myClient){
  for(int i = 0; i < 6; i++){
    myClient.print("Analog Pin");
    myClient.print(i);
    myClient.print(": ");
    myClient.print(analogRead(i));
    myClient.print("<div style\"height: 15px; background-color: #");
```

```
    myClient.print(barColor[i]);
    myClient.print("; width: ");
    myClient.print(analogRead(i));
    myClient.println("px; border: 2px solid;\"></div>");
   }
}
```

Ayant par chance une loupe sur moi, je vois dans l'en-tête de fonction un ET commercial (&) devant le paramètre `myClient`. Vous connaissant, il ne s'agit pas d'une faute de frappe, n'est-ce pas ?

Non Ardus, ce n'est pas une erreur. Ce signe distinctif est en fait une *référence*. Quand on passe une variable comme paramètre d'une fonction, celle-ci travaille avec une copie de cette variable, ce qui n'a aucune influence sur la variable d'origine. La fonction peut par exemple doubler la valeur du paramètre. L'originale demeure inchangée. Mais pour pouvoir utiliser l'objet `Client` original dans la fonction, l'adresse mémoire de l'original est communiquée au moyen de l'opérateur de référence `&`. Dans la fonction, je travaille quasiment avec l'original. La fonction affiche d'une part les valeurs des entrées analogiques et de l'autre des barres horizontales. J'utilise pour ce faire la balise `div`, qui peut servir de contenant pour d'autres éléments HTML. Je m'en sers ici pour colorer une certaine zone. Il est possible de donner des informations de hauteur ou de largeur au moyen d'une indication *style*. Une ligne HTML peut alors ressembler à cela :

```
Analog Pin 0: 168<div style="height: 25px; background-color:
                  #ff0000; width: 168px; border: 2px solid;"></div>
```

La zone `div` a ici une hauteur de 25 pixels et une largeur de 168 pixels. Vous trouverez des informations détaillées dans la littérature spéciale ou sur Internet.

Pour aller plus loin

Pour compléter ce chapitre, vous pouvez effectuer une recherche sur Internet sur les mots-clés :

- selfhtml ;
- cascading stylesheets ;
- div-tag.

Il y a quelque chose que je n'ai pas trouvé commode lors de ma réalisation : les valeurs des entrées analogiques s'affichent un point c'est tout. Si je tourne l'un des potentiomètres, rien ne bouge sur la page Internet. J'aurais pourtant bien voulu.

C'est inutile, Ardus. Le navigateur web appelle une page auprès du serveur web et en assure la présentation (ce procédé est également appelé *rendu*). Si le navigateur n'émet aucune autre demande, le contenu de la page demeure bien entendu inchangé. Vous pouvez toujours appuyer assez souvent sur la touche Actualiser (F5) mais je doute que cela vous donne satisfaction. Modifiez plutôt dans votre sketch la ligne de code où `HTML_TOP` a été défini et vous verrez que le comportement de votre navigateur changera.

```
#define HTML_TOP "<html>\n<head><title>Arduino Web-server</title>\
        </head>\n<meta http-equiv=\"refresh\" content=\"1\">\n<body>"
```

Le passage suivant est décisif :

```
<meta http-equiv=\"refresh\" content=\"1\">
```

La balise `meta` en question demande au navigateur d'exécuter automatiquement une actualisation (`refresh`) toutes les secondes. Le *backslash* (barre oblique inverse) \ à la fin de la première ligne définissant `HTML_TOP` permet que cette ligne se poursuive sur la suivante. Faute de quoi une erreur de compilateur se produirait.

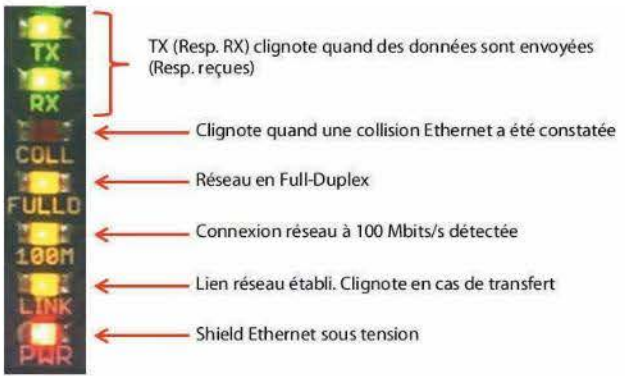
Problèmes courants

Vérifiez ce qui suit si la page du serveur web ne s'affiche pas.

- Avez-vous saisi la bonne adresse IP dans la ligne d'adresse de votre navigateur ? Elle doit correspondre à celle de votre sketch.
- Pouvez-vous atteindre le serveur web en tapant une commande `ping` dans la ligne de commande ? Sinon, vérifiez votre câble réseau ou éventuellement les réglages du pare-feu. Une exécution réussie de la commande `ping` donne le résultat suivant.

```
C:\Users\Olivier>ping 192.168.2.110

Envoi d'une requête 'Ping'  192.168.2.110 avec 32 octets de données :
Réponse de 192.168.2.110 : octets=32 temps=2 ms TTL=128
Réponse de 192.168.2.110 : octets=32 temps=3 ms TTL=128
Réponse de 192.168.2.110 : octets=32 temps=2 ms TTL=128
Réponse de 192.168.2.110 : octets=32 temps=2 ms TTL=128

Statistiques Ping pour 192.168.2.110:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 2ms, Maximum = 3ms, Moyenne = 2ms

C:\Users\Olivier>
```

- Le shield Ethernet possède certaines LED qui donnent des informations sur l'état (voir figure suivante).



- Vérifiez l'affichage des LED. Les LED PWR et LINK doivent être allumées. La LED 100M ne s'allume que dans le cas d'un réseau 100 Mbits/s. Elle reste éteinte pour 10 Mbits/s. Si des données sont envoyées toutes les secondes comme dans le dernier exemple, Les LED TX et RX clignotent au même rythme.

Qu'avez-vous appris ?

- Vous avez appris à réaliser un serveur web avec le shield Ethernet.
- Vous avez interrogé les entrées analogiques et vu comment s'affichent les valeurs à peu de choses près en temps réel.
- La trame de base d'une page HTML devrait maintenant vous être plus familière.

Exercice complémentaire

Écrire un nouveau sketch montrant, en plus des entrées analogiques, l'état des entrées numériques sur votre page web Arduino.