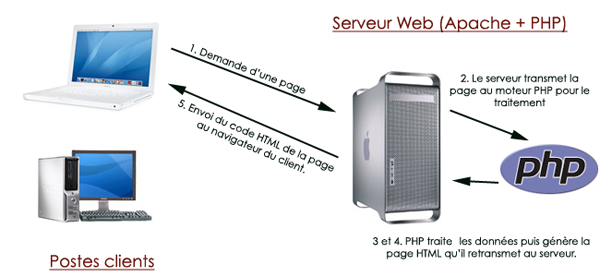
|  |  |  |
| --- | --- | --- |
|  | **Architecture client-serveur** | Description : pem |
| **Réseaux**  **Cours / TD** | Mots-clés : client-serveur, requête, protocole, HTTP, FTP, méthode, GET, POST, URL. |



**Objectifs**: Connaître l’architecture client-serveur d’un système communicant et comprendre comment les machines échangent des données avec le protocole HTTP.

**Table des matières**

[1 Le modèle client-serveur 2](#_Toc509213778)

[1.1 Mise en situation 2](#_Toc509213779)

[1.2 Généralités 2](#_Toc509213780)

[1.3 Modèle de dialogue entre les machines 2](#_Toc509213781)

[2 HTTP (HyperText Transfert Protocol) 3](#_Toc509213782)

[2.1 Généralités 3](#_Toc509213783)

[2.2 Les méthodes spécifient le type de la requête 3](#_Toc509213784)

[2.3 Organisation d'une requête et d'une réponse 4](#_Toc509213785)

[2.4 Les scénarios client-serveur 5](#_Toc509213786)

[3 Autre protocole : FTP (File Transfert Protocol) 6](#_Toc509213787)

[4 Synthèse 6](#_Toc509213788)

[5 Exercices 7](#_Toc509213789)

[Annexe 1 : schéma du réseau support des projets 9](#_Toc509213790)

[Annexe 2 : code exécuté lors du transfert d'une donnée 10](#_Toc509213791)

**Pour aller plus loin…**

**Cours sur OPENCLASSROOMS "Comprendre le Web – Découvrez les protocoles"**

<http://bit.ly/3p8xk7V>

**Utilisation du "moniteur réseau" de Mozilla**

<http://mzl.la/3h2hqJz>

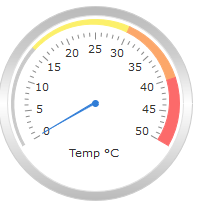
**Client-serveur sur Wikipédia**

<http://bit.ly/3rdM5rQ>

**Logiciel FTP**

<https://filezilla-project.org/>

# Le modèle client-serveur



## Mise en situation

La température ambiante dans une pièce, mesurée par une carte **Arduino Uno Wifi,** est **surveillée à distance** à l'aide d'une jauge, affichée dans un navigateur.

Le code de la jauge est installé sur un Nas Synology car la carte Arduino ne dispose pas des ressources matérielles et logicielles nécessaires à son fonctionnement.

Le navigateur communique avec la carte Arduino via le NAS. Les couples PC, NAS et NAS, carte Arduino mettent en œuvre une architecture **client-serveur**.

Le fonctionnement est le suivant :

**Étape 1.** Le navigateur situé sur le PC1 demande au NAS la page contenant la jauge.

**Étape 2.** Après le chargement de la page dans le navigateur, celui-ci demande périodiquement la valeur de la température et modifie la position de l'aiguille de la jauge.

Les échanges se font alors selon le schéma ci-dessous.

(5) Merci **@IP2**



**Client HTTP**

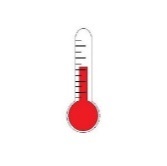
**PC1[@IP1]**

**Serveur HTTP**

**Arduino Uno Wifi [@IP3]**

**GET**





(2) Hé **@IP3** Donne-moi la valeur de la température !

(3) La voilà **@IP2** !

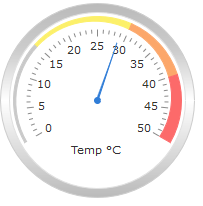
(1) Hé **@IP2**, "Demande à **@IP3** la valeur de la température" !

(4) Merci **@IP3**, la voilà **@IP1** !

**Client et serveur HTTP**

**NAS [@IP2]**

**GET**

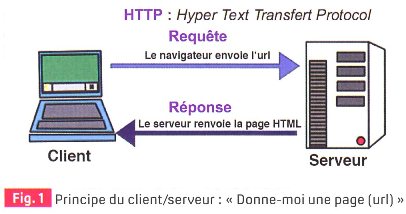


**valeur**

**valeur**



## Généralités



L’architecture **client-serveur** désigne un **mode de communication** entre plusieurs composants d’un réseau. Chaque entité est considérée comme un **client** ou un **serveur**. Chaque logiciel client peut envoyer des **requêtes** à un serveur.

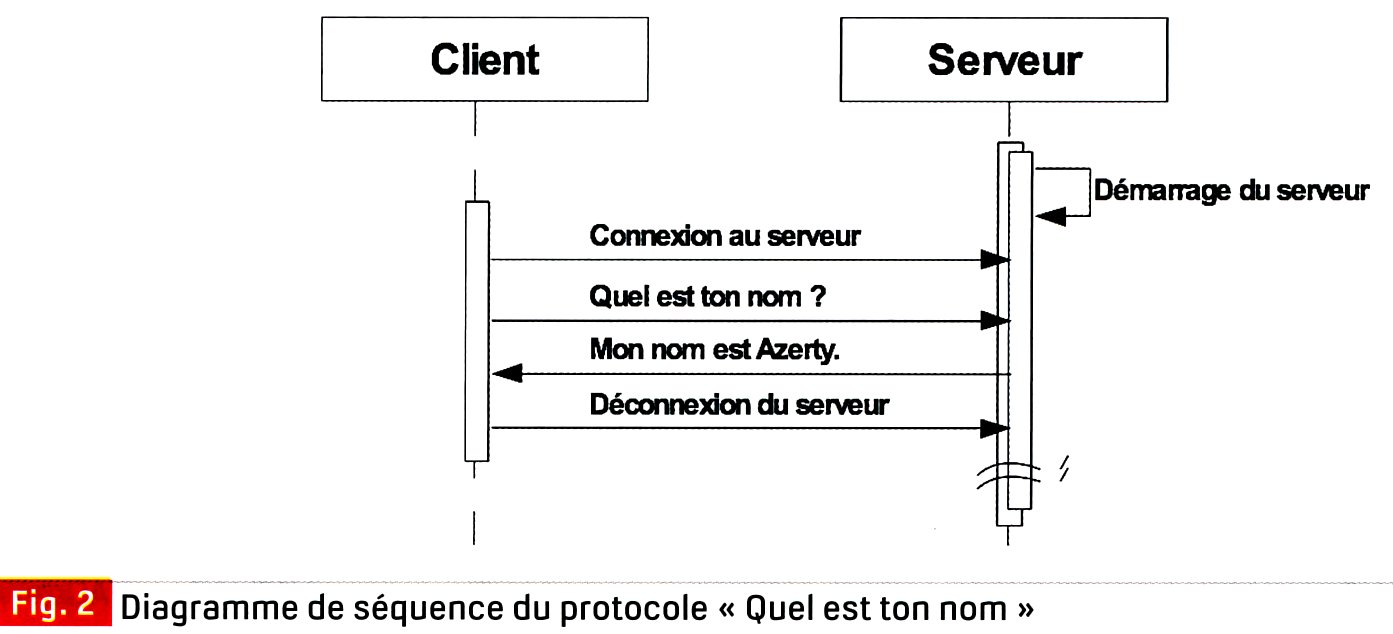
Un serveur peut être spécialisé en serveur d’**applications**, de **fichiers**, de terminaux, ou encore de **messagerie électronique**.

**En résumé, le client pose une question (ou donne un ordre), le serveur répond (ou obéit).**

## Modèle de dialogue entre les machines

* **Le serveur**

Un serveur est initialement passif, **il attend**. Il est à l’**écoute**, prêt à répondre aux **requêtes** envoyées par des clients, dès qu’une requête lui parvient, il la traite et envoie une réponse.



* **Le client**

Le client est d’abord actif (ou maître), il envoie des **requêtes** au serveur, il attend et reçoit les réponses du serveur.

* **Le dialogue**

Le client et le serveur doivent utiliser le même **protocole** de communication.

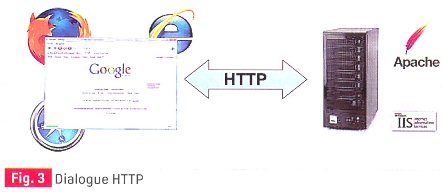
Remarque : Un serveur est généralement capable de **servir plusieurs clients** simultanément.

# HTTP (HyperText Transfert Protocol)

## Généralités

La consultation de pages sur un site Web a un fonctionnement basé sur l’architecture client-serveur. Un internaute se connecte au réseau via son ordinateur et un navigateur Web. Le navigateur est le client. Le serveur est constitué par la ou les machines (ordinateurs, carte à microcontrôleur, etc.) contenant les applications qui délivrent les pages demandées.

Dans ce cas, c’est le protocole de communication HTTP qui est utilisé.

**

L’**H**yper**Text** **T**ransfert **P**rotocol, plus connu sous l’abréviation HTTP, littéralement le « protocole de transfert hypertexte », est le **protocole de communication client-serveur** développé pour le **Web**.

Les **navigateurs** (Firefox, Safari, Edge, etc.) **sont les clients HTTP**. Ces clients se connectent à des serveurs HTTP tels qu’**Apache, Nginx** ou IIS (Internet Information Service).

## Les méthodes spécifient le type de la requête

Dans le protocole HTTP, une **méthode est une commande** spécifiant un type de **requête**, c'est-à-dire qu’elle demande au serveur d’effectuer une action. En général, l’action concerne une ressource identifiée par l’**URL** qui suit le nom de la méthode.

**URL (U**niform **R**essource **L**ocator**)**

**protocol ://**[user:password@]**host**[:port][/path][/document]

La **requête HTTP** est **créée à partir de l’URL entrée dans le navigateur**. Elle est constituée :

**1. de l’adresse du serveur,**

**2. du numéro de port,**

Ceux-ci sont codés sur **16 bits**

|  |  |
| --- | --- |
| **Port** | **Service ou Application** |
| 21 | [FTP](http://www.commentcamarche.net/contents/internet/ftp.php3) |
| 23 | [Telnet](http://www.commentcamarche.net/contents/internet/telnet.php3) |
| 25 | [SMTP](http://www.commentcamarche.net/contents/internet/smtp.php3) |
| 53 | [Domain Name System](http://www.commentcamarche.net/contents/internet/dns.php3) |
| 63 | Whois |
| 70 | Gopher |
| 79 | Finger |
| 80 | [HTTP](http://www.commentcamarche.net/contents/internet/http.php3) |
| 110 | [POP3](http://www.commentcamarche.net/contents/internet/smtp.php3) |
| 119 | NNTP |

- Les ports 0 à 1023 sont les "**ports reconnus**" ou réservés ("**Well Known Ports**"). Ils sont, de manière générale, réservés aux processus système (démons) ou aux programmes exécutés par des utilisateurs privilégiés. Un administrateur réseau peut néanmoins lier des services aux ports de son choix.

- Les ports 1024 à 49151 sont appelés "**ports enregistrés**" ("**Registered Ports**").

- Les ports 49152 à 65535 sont les "ports dynamiques et/ou privés" ("Dynamic and/or Private Ports").

**3. du chemin d’accès au fichier.**

**L'exemple de la mise en situation**

Lorsque le navigateur demande au NAS la page index.html contenant le widget jauge, l'URL [**http://192.168.1.6:10003**](http://192.168.1.6:10003/scripts/get_val1.php) identifie la ressource (page index.html).

Il existe plusieurs méthodes (GET, POST, PUT, HEAD, OPTION etc.). Les méthodes les **plus utilisées sont GET et POST**.

|  |  |
| --- | --- |
| PUT | Cette méthode permet de remplacer ou d'ajouter une ressource sur le serveur. L'URL fournie est celle de la ressource en question |
| **GET** | C’est la méthode la plus courante pour **demander une ressource**. Une requête GET est sans effet sur la ressource. |
| **POST** | Cette méthode est utilisée pour **soumettre des données** en vue d'un traitement à une ressource (typiquement depuis un formulaire HTML). L'URL fournie est celle d'une ressource à laquelle s'appliqueront les données envoyées. Le résultat peut être la création de nouvelles ressources ou la modification de ressources existantes.  Remarque : À cause de la norme HTML qui ne supporte que les méthodes GET et POST pour les formulaires, cette méthode est souvent utilisée en remplacement de la requête PUT, qui devrait être utilisée pour la mise à jour de ressources. |

## Organisation d'une requête et d'une réponse

* **Requête**

Quelle que soit la méthode invoquée, la syntaxe d’une requête HTTP a le même format de base composé de **deux parties** : l'**en-tête** et le **corps.**

**En-tête**

|  |  |
| --- | --- |
| ***Methode URL Version***  ***En-tête : valeur***  *En-tête : valeur*  …  **Corps** de la requête | ***Méthode*** = (GET, POST, etc.)  ***URL*** placée dans le champ d’en-tête Host  ***Version*** = HTTP1.1  ***En-tête : valeur*** *sont des* ***l*ignes facultatives** (Connection, Accept-, User-Agent, etc.) à l’exception de Host !  **Le corps** est utilisé par exemple pour transmettre des données (POST) |

**L'exemple de la mise en situation**

Lorsque le navigateur demande au NAS la page index.html contenant la jauge, la requête transmise est créée à partir de l'URL [**http://192.168.1.6:10003**](http://192.168.1.6:10003)**.**

**Méthode**

**Version**

**En-tête**

**GET HTTP/1.1**

Host: 192.168.1.6:10003

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0

Accept: text/html, \*/\*; q=0.01





Requête copiée dans la page Network Monitor de Mozilla

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

etc.

* **Réponse**

Également constituée de deux parties :

**En-tête**

|  |  |
| --- | --- |
| ***Version CodeRéponse***  *En-tête : valeur*  *En-tête : valeur*  …  **Corps** de la réponse | **Version** = HTTP/1.1  ***CodeRéponse*** = (200 OK, 204 NO RESPONSE, 404 NOT FOUND…)  ***En-tête : valeur*** *sont des* ***l*ignes facultatives** (type de serveur, type MIME, etc.)  **Corps** : Données HTTP (généralement, la page WEB au format HTML) |

**L'exemple de la mise en situation**

**Version**

**Code de la réponse**

**Étape 1 :** Le NAS renvoie au navigateur la page contenant la jauge.

**En-tête**

**HTTP/1.1 200 OK**

Server: nginx

Date: Wed, 07 Mar 2018 14:21:27 GMT

Content-Type: text/html; charset=UTF-8

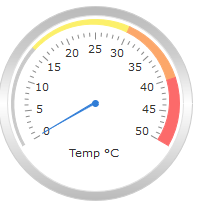
Content-Length: 22

Connection: keep-alive

Keep-Alive: timeout=20

Vary: Accept-Encoding

etc.



**Réponse**

<!DOCTYPE html>

<html lang="fr">

<head>

<title id='Description'>Meteo –

démo REST</title>$

etc.

## Les scénarios client-serveur

Lorsqu'une machine est connectée à un réseau TCP/IP, elle peut communiquer avec une autre machine en tant que :

**1 - Client envoyant des données à un serveur.**

**2 - Client recevant des données d’un serveur.**

**3 - Serveur fournissant des données à un client.**

Direction de la requête

Direction des données

**4 - Serveur acceptant des données d’un client.**

**Ces différents scénarios sont illustrés ci-dessous.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Scénario 1** | **Scénario 2** | **Scénario 3** | **Scénario 4** |
| **Serveur** | **Serveur** | **Client** | **Client** |
| C:\Users\Philippe\Desktop\pc.jpg | C:\Users\Philippe\Desktop\pc.jpg | C:\Users\Philippe\Desktop\pc.jpg | C:\Users\Philippe\Desktop\pc.jpg |
| **PUT**  ou  **POST** | **GET** | **GET** | **PUT**  ou  **POST** |
| C:\Users\Philippe\Desktop\Netduino plus2.jpg  **Client** | C:\Users\Philippe\Desktop\Netduino plus2.jpg  **Client** | C:\Users\Philippe\Desktop\Netduino plus2.jpg  **Serveur** | C:\Users\Philippe\Desktop\Netduino plus2.jpg  **Serveur** |
|  |  |  |  |

**À retenir : une machine cliente envoie des requêtes au serveur.**

**Les applications**

**- Surveillance à distance (Schémas 1 et 3)**

Dans ce type d’application, la carte à microcontrôleur **produit des données** (les valeurs des grandeurs physiques mesurées par des capteurs).

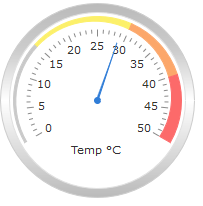
**- Commande à distance (Schémas 2 et 4)**

Dans ce type d’application, la carte à microcontrôleur **consomme** des données.

**À retenir** : le sens de la requête **indique si la machine** est un **client** ou un **serveur**. Le sens des données **indique si la machine** est un **producteur** ou un **consommateur**.

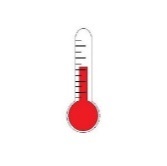
**L'exemple de la mise en situation**

En faisant abstraction du NAS, le cas étudié précédemment répond au **scénario 3**. La jauge permet de **surveiller** la valeur de la température ambiante à distance.



**GET**

température



**Navigateur [@IP1]**

**(Client HTTP)**



**Arduino Uno Wifi [@IP3]**

**(Serveur HTTP)**

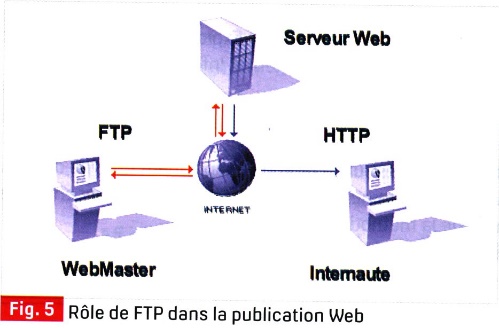
# Autre protocole : FTP (File Transfert Protocol)



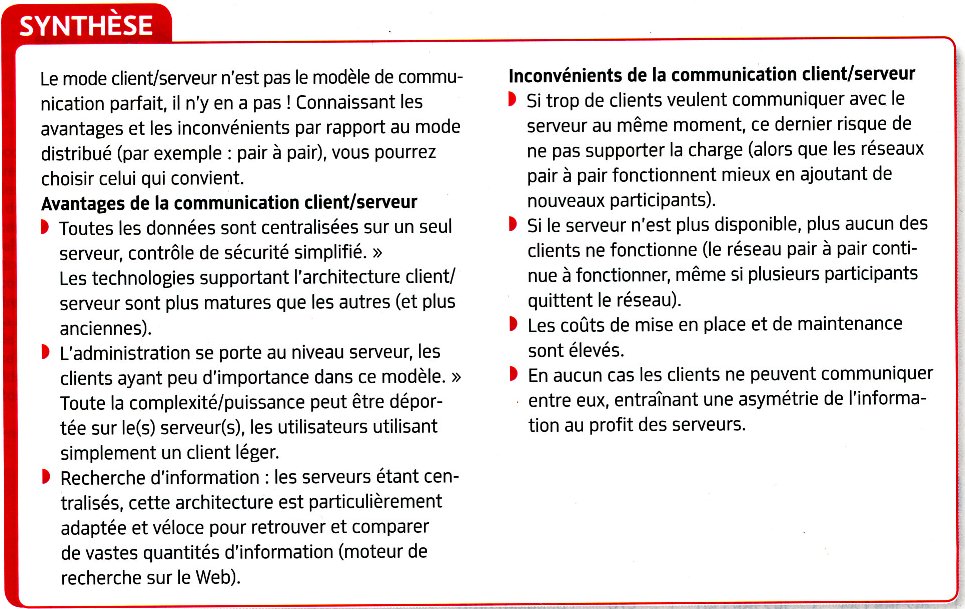
**Client FTP Filezilla**

Le protocole de transfert de fichiers, ou FTP est un protocole de communication destiné à l’échange de fichiers sur un réseau TCP/IP. Il permet de copier des fichiers vers un autre ordinateur du réseau, d’alimenter un site Web, ou encore de supprimer ou de modifier des fichiers sur cet ordinateur.

FTP obéit à un modèle client-serveur. C'est-à-dire qu’une des deux parties, le client, envoie des requêtes et le serveur répond.



# Synthèse



# Exercices

**Exercice 1**

Sachant que les machines de l'application présentée dans la mise en situation sont sur le réseau 192.168.40.0/24 et que la page index.html, contenant la jauge, est accessible sur le port 10003 du NAS, **dessinez** un schéma illustrant l'**étape 1.**

*Réponse*



**PC1**

**Client HTTP**

**192.168.40.1/24**



**NAS**

**Serveur HTTP**

**192.168.40.2/24**

**http://192.168.40.2:10003**

**Index.html**

**Exercice 2**

Complétez le diagramme de séquence ci-dessous pour qu'il corresponde au cas d'utilisation de l'exercice 1.

*Réponse*

**PC1**

**NAS**

**http://192.168.40.2:10003**

**Index.html**

**Connexion au serveur**

**Déconnexion du serveur**

**Exercice 3**

Le schéma d'une application illustrant le modèle client-serveur est donné ci-dessous.

**Devtools Mozilla** Analyse d'une requête **GET** de la réponse renvoyée.

(5) Merci **@IP2**



**Client HTTP**

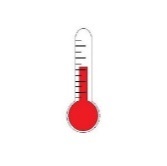
**PC1[@IP1]**

**Serveur HTTP**

**Arduino Uno Wifi [@IP3]**

**GET**





(2) Hé **@IP3** Donne-moi la valeur de la température !

(3) La voilà **@IP2** !

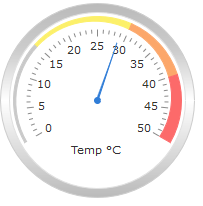
(1) Hé **@IP2**, "Demande à **@IP3** la valeur de la température" !

(4) Merci **@IP3**, la voilà **@IP1** !

**Client et serveur HTTP**

**NAS [@IP2]**

**GET**

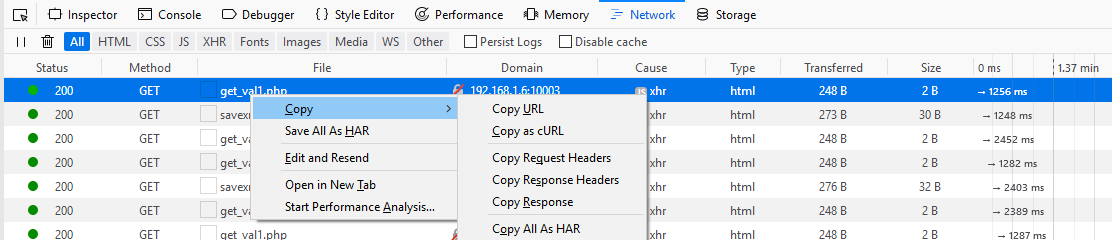


**valeur**

**valeur**



Le **moniteur réseau** du navigateur Mozilla **Firefox**, accessible par **F12** sous Windows, affiche toutes les requêtes. Une copie d'écran, illustrant le fonctionnement de l'application, a été réalisée ci-dessous.



Un clic droit sur une requête permet d'obtenir le détail de son contenu à l'aide du menu **Copy**.

**Copy\_URL**

http://192.168.1.6:10003/scripts/get\_val1.php

|  |  |
| --- | --- |
| **Copy Request Headers**  **Version**  **URL**  **Méthode**  **GET** /scripts/get\_val1.php **HTTP/1.1**  Host: 192.168.1.6:10003  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0  Accept: text/html, \*/\*; q=0.01  Accept-Language: en-US,en;q=0.5  Accept-Encoding: gzip, deflate  Referer: http://192.168.1.6:10003/  X-Requested-With: XMLHttpRequest  DNT: 1  Connection: keep-alive | **Copy Response Headers**  HTTP/1.1 200 OK  Server: nginx  Date: Thu, 08 Mar 2018 07:20:18 GMT  Content-Type: text/html; charset=UTF-8  Content-Length: 22  Connection: keep-alive  Keep-Alive: timeout=20  Vary: Accept-Encoding  Content-Encoding: gzip |
| **Copy Response**  27 |

**Travail demandé**

**Identifiez** ci-dessus : le texte de la requête, le texte de la réponse, la méthode utilisée, l'URL dans l'en-tête de la requête, la version du protocole HTTP, l'adresse de l'hôte de la requête, et le code de la réponse.

Quelle valeur de température est affichée par la jauge ?

**Exercice 4 : Activité de projet**

Complétez le schéma de l'annexe 1 après l'avoir reproduit dans votre dossier.

**Exercice 5 : Activité de projet**

Personnalisez le schéma et le code de l'annexe 2 pour qu'ils correspondent aux échanges de données de votre projet après les avoir reproduits dans votre dossier.

# Annexe 1 : schéma du réseau support des projets

**Adresse du réseau de test (Wifi)** : **192.168.200.0**

**Masque :** **255.255.255.0**



**Arduino Uno Wifi**

IP : \_\_\_\_\_\_Elève\_\_\_\_

**Routeur Synology RT1900AC**

IP : 192.168.200.1/24

**PC**

IP : \_\_\_\_\_\_\_\_Elève\_\_\_\_\_\_

**Serveur de test Synology DS112**

IP1 : 192.168.200.2/24

IP2 : 192.168.231.188/21

**Serveur Lycée**

IP1 : x.x.x.x/21

Ethernet

Ethernet

Ethernet

# Annexe 2 : code exécuté lors du transfert d'une donnée

Remarque : la page contenant le widget jauge a été chargé dans le navigateur.



**(Client HTTP)**

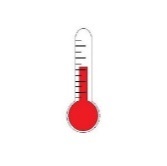
**PC1[@192.168.200.100/24]**

**(Serveur HTTP)**

**Arduino Uno Wifi [192.168.200.96/24]**

**GET [2]**





(2) Hé **@IP3** Donne-moi la valeur de la température !

(3) La voilà **@IP2** !

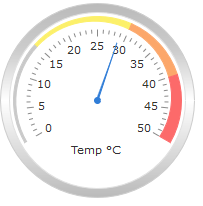
(1) Hé **@IP2**, "Demande à **@IP3** la valeur de la température" !

(4) La voilà **@IP1** !

**Client et serveur HTTP**

**NAS [192.168.200.2/24]**

**GET [1]**



**temp [3]**

**temp [4]**



 <?php

$service\_url = "**http://192.168.200.96/arduino/analog/temp**";

$curl = curl\_init($service\_url);

// Send cURL to Uno Wifi board

curl\_setopt($curl, CURLOPT\_IPRESOLVE, CURL\_IPRESOLVE\_V4);

**curl\_exec($curl); [2,4]**

curl\_close($curl);

?>

**// Code partiel**

**…**

void loop() { // Attente requête

while (Wifi.available()) {

process(Wifi);

}

delay(50);

}

void process(WifiData client) {

String command = client.readStringUntil('/');

if (command == "analog") {

analogCommand(client);

}

void analogCommand(WifiData client) {

String command = client.readStringUntil('\r');

if (command == "**temp**") { // **[3]**

temperature = analogRead(0);

client.println("HTTP/1.1 200 OK\n");

client.print(temperature);

client.print(EOL); }

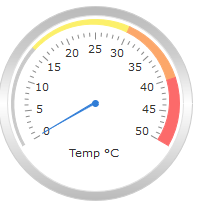
}

**testUnoWifi.ino**

**Get\_temp.php**

**gaugetemp.js**

$('**#gaugetemp**').jqxGauge({value:**temp**});



**[1]**

$("**#temp**").load('scripts/get\_temp.php');

var **temp** = $("#temp").html() ;