**COURS ARDUINO\_COURS 2**

[**https://openclassrooms.com/fr/courses/3290206-perfectionnez-vous-dans-la-programmation-arduino/3290213-decouvrez-les-principes-mecaniques-des-robots-autonomes**](https://openclassrooms.com/fr/courses/3290206-perfectionnez-vous-dans-la-programmation-arduino/3290213-decouvrez-les-principes-mecaniques-des-robots-autonomes)

**PARTIE 1**

**PARTIE 2**

**6\_ MONITEUR SERIE EN ENTREE-SORTIE**

Principe : le même à chaque envoi de données vers Arduino (depuis ordinateur, Internet, autre arduino...)

* **Le moniteur série pour envoyer des données**

Interface : **photo6\_1** légende

1\_ titre de la fenêtre = port de connexion utilisé / 2\_zone de saisie / 3\_ bouton "envoyer"

4\_ zone d'affichage (par Serial.print()  ou  Serial.println()) / 5\_case à cocher = arrêt défilement

6\_ menu déroulant (4 possibilités) / 7\_ menu déroulant = paramétrer les bauds (vitesse de transmission)

Test de vitesse : ajuster programme et interface

|  |
| --- |
| void setup() {  Serial.begin(115200);  }  void loop() {  Serial.println("test");  } |

* **Un peu de théorie**

**vocabulaire**

ensemble codes = **casse** en typographie (minuscules= en bas du tiroir (bas de casse) )

**caractère =** forme lettre ou ponctuation qui sera imprimé / **graisse** = épaisseur du trait d'un caractère

**style** = forme donnée à chaque caractère (helvétique, comic sans ms...) / **corps** = taille de la lettre (h/l)

**variation =** faire varier la lettre (italique, +/- grasse, serif ou non, … / **fonte =** groupe même style-graisse-corps

**police =** tous les caractères d'une même fonte, dont le style est coordonné.

**lettre en pixels photo6\_2** caractères affichés sur une matrice de 7x5 (aujourd’hui bcp plus)

Principe : transposition du caractère sur un affichage à point

Table ASCII (American Standard Code for Information Interchange) = utilisée par Arduino pour le moniteur

* « premier » standard codage des caractères = **code ASCII**, 1971, permet de coder 2^7=128 caractères
* code ASCII fait correspondre un code à un caractère imprimable (à l'écran par exemple).

Référence : [tableau des codes ASCII](https://www.arduino.cc/en/Reference/ASCIIchart) de l'Arduino

* les 33 premiers codes pas imprimables = caractères de commande.
* pas de lettres accentuées car code ASCII est d'origine américaine

**affichage d’un caractère simple sur le moniteur :**  lettre “A”= code “65”

|  |
| --- |
| void setup() {  Serial.begin(9600);  Serial.println();  Serial.println(65);  Serial.println('A');  Serial.println(char(65));  Serial.println(int('A'));  }  void loop() {    } |

Résultat : affichage sur le moniteur = 65 A A 65

Explication : on peut déclarer un caractère de deux façons :  char caractere='A';  ou  char caractere=65;

* Serial.println(65) : affiche le nombre 65 = 2 caractères (6 et 5)
* Serial.println('A') : affiche la lettre A : **guillemets simples indiquent qu'il s'agit d'un type caractère**
* Serial.println(char(65)) : affiche la lettre A = **char() force le typage en caractère**
* Serial.println(int('A')) : affiche 65 = instruction  int() force le typage vers un nombre entier

Exercice : programme qui affiche le tableau des codes ASCII et des caractères correspondants, entre 32 et 127.

**afficher des caractères spéciaux sur le moniteur** caractères spéciaux utilisés fréquemment

| Code ASCII | Fonction | Code lettre |
| --- | --- | --- |
| 0 | NULL | \0 |
| 9 | Tabulation | \t |
| 10 | Fin de ligne | \n |
| 13 | retour chariot | \r |

1. Écritures possibles :

* code ASCII avec la fonction  char() :  Serial.println( char(9)) = tabulation.
* directement code lettre :  Serial.println('\t") = tabulation (caractère d’échappement  \ avant la lettre)

Remarque : retour à la ligne sur le moniteur = Serial.print('\n'); /  Serial.print('\r'); selon les machines

Exercice : afficher table ASCII avec un tab entre code et caractère avec caractères spéciaux (et pas println()).

**afficher un mot sur le moniteur**

Un mot = successions de caractères stockés dans un tableau de type  char (appelé chaîne de caractères)

Déclarations possibles :

* char chaine[15]; // chaîne de 14 caractères + 1 place pour le caractère null (14+1=15)
* char chaine[8]={'B','o','n','j','o','u','r'}; // chaîne de 7 caractères en laissant la dernière comme null (7+1=8)
* char chaine[8]={'B','o','n','j','o','u','r','\0'}; // 8 caractères dont la valeur nulle à la fin
* char chaine[]="Bonjour"; // directement la chaîne qui contiendra 7 caractères plus le caractère null
* char chaine[8]="Bonjour"; // explicitement la chaîne et réserve la dernière place pour le caractère null
* char chaine[15]="Bonjour"; // une chaîne de 7 caractères et autres cases contiendront caractère null

Explications : **lorsqu'on gère des chaînes de caractères, on gère en fait des tableaux de type  char**

* caractère  NULL,  char(0) ou  '\0'  = caractère qui signale que la chaine est finie (à la fin de chaque chaîne)
* guillemets simples pour définir un caractère <> guillemets doubles pour définir une chaîne
* La chaîne de caractère en anglais est appelée string.

**parcourir une chaîne de caractères =** parcourir le tableau jusqu’au caractère  NULL

|  |
| --- |
| char mot[]="Arduino"; // on initialise la chaîne de caractères  void setup() {  Serial.begin(9600); // début communication série  Serial.println(); // on saute une ligne  int l=0; // variable de position dans la chaîne de caractère  while (mot[l]){ // tant que la lettre n'est pas le caractère NULL  Serial.print(mot[l]); // on écrit la lettre  Serial.print(','); // on écrit une virgule  l++; // on avance d'une position dans la chaîne  }  }  void loop() {    } |

Remarque : on peut utiliser une boucle for aussi avec la longueur de la chaine

**remplacez un caractère dans une chaîne**

Programme = remplace tous les espaces par des tirets

|  |
| --- |
| char mot[]="Jean de la Fontaine"; // une chaîne de caractère avec des espaces  void setup() {  Serial.begin(9600); // on initialise la communication série  Serial.println(); // on saute une ligne  int l=0; // variable pour parcourir la chaîne  while (mot[l]){ // tant qu'on n'est pas au bout de la chaîne (caractère NULL)  if (mot[l]==' ') // on teste si le caractère est un espace  mot[l]='-'; // si oui, on le remplace par un tiret  l++; // on passe au caractère suivant  }  Serial.println(mot); // on écrit la chaîne résultante  }  void loop() {    } |

Exercice : programme qui inverse les lettres d'un mot mais caractère  NULL , qui doit rester à la fin

* **Envoyez un message à l'Arduino**

Rappel : Serial.print() et  Serial.println() envoient la chaîne caractère par caractère.

Principe : Arduino lit un message un caractère à la fois (boucle  while  jusqu'à atteindre la fin de la chaîne)

1. Arduino doit être à l'écoute d'une éventuelle arrivée d'information

Serial.available();. = retourne le nombre de caractères en attente d'être reçus.

1. Si caractères (type  char) en attente, utiliser Serial.read(); = lit 1er caractère de la chaîne en attente

= on peut stocker ce caractère, l'écrire, le traiter.

**Programme** : renvoyer ce que vous avez écrit

|  |
| --- |
| void setup() {  Serial.begin(9600); //début de la communication série  }  void loop() {  while (Serial.available()) { // tant que caractères en attente d'être lus  char c = Serial.read(); // on lit le charactère  Serial.print(c); // on l'écrit sur le moniteur  delay(10); // petit temps de pause  }  } |

Test : ouvrir le moniteur et tapez un texte en haut, puis cliquez sur envoyer.

Résultat : affiche mot / 2ème mot collé = "envoyer" n'envoie pas de caractère de fin de ligne et de retour chariot.

Remarque : envoi de  '\n'  n'a aucun effet car le moniteur va envoyer le caractère  '\'  et le caractère  'n'.

**Programme** : affiche le caractère et à côté le code ASCII qui correspond

|  |
| --- |
| void setup() {  Serial.begin(9600); // initialisation de la communication  }  void loop() {  while (Serial.available()) { // tant que des caractères sont en attente  char c = Serial.read(); // on lit le caractère  Serial.print(c); // on l'affiche  Serial.print('\t'); // on affiche une tabulation  Serial.println(int(c)); //on affiche le code correspondant  delay(10); //petit temps de pause  }  } |

Résultat : apparition des codes 10 ou 13 = caractères de fin de ligne (10) ou de retour chariot (13).

Remarque : petits temps de pause servent à clarifier la communication sinon risque perdre quelques caractères

**7\_ CODER-DECODER DES INFORMATIONS Matériel nécessaire : 5 LED et 5 résistances 220Ω**

* **créez un code** afficher la LED choisie sur unmontage avec 5 LED

Idée 0 : choisir des mots-clés : allumer / éteindre / clignoter

**Idée 1 : coder instructions => a= allumer / e=éteindre / c=clignoter, n pour numéro de LED**

ex1: "c2"="fait clignoter la LED 2" / " c23"="fait clignoter la diode 2 et 3" /

ex2 : "c23e4a15" = " fait clignoter la LED 2 et la LED 3, éteins la LED 4 et allume la LED 1 et la LED 5".

**Idée 2 : coder directement l'état des diodes => "accea" # "c23e4a15"**

= chaque lettre correspond à l’état de la diode correspondante

* **décodez un train d'informations**

Arduino recoit ce code par port série (port USB) : Serial.available()  indiquer données attente

* **décoder la bonne quantité d'informations : pour éviter des erreurs de saisie ou d’envoi**
  + **seulement 5 caractères** car 5 LEDs
  + **purgé l’excédent** (lu pour vider la mémoire d'attente, mais pas stocké)
  + moyen de vérifier bonne réception (*Exp* : a=1, e=2, c=4 donc "accea" vaut 1+4+4+2+1=12)

Programme : reçoit données du moniteur série et transforme en code de vérification :

|  |
| --- |
| char reception[5]; //tableau pour la réception des données (5 valeurs)  void setup() {  Serial.begin(9600); //on initialise la communication série  }  void loop() {  while (Serial.available()) { //tant que des données sont en attente  for (int p=0;p<5;p++){ //on en lit 5  char c=Serial.read(); // lecture  reception[p]=c; //écriture dans le tableau  delay(10); //petite attente  }  while (Serial.available()) //s'il reste des données  Serial.read(); //on les lit sans les stocker  decodage(reception); // on appelle la fonction de décodage  }  }  //fonction pour décoder la réception  void decodage(char c[5]){  for (int p=0;p<5;p++){ //on parcourt le tableau  Serial.print("LED "); //on écrit le mot LED  Serial.print(p+1); //on écrit le numéro de la LED (position dans taleau +1)  Serial.print(" "); //on fait un espace  if (c[p]=='a') // on teste si le code est 'a'  Serial.println("allumee"); //si oui on écrit "allumee"  else if (c[p]=='e') // sinon on teste si le code est 'e'  Serial.println("eteinte"); //si oui on écrit "éteinte"  else if (c[p]=='c') //sinon on teste si le code est 'c'  Serial.println("clignote"); //si oui on écrit "clignote'"  else //sinon c'est que ça ne correspond à rien  Serial.println("erreur de code"); // on inscrit donc "erreur de code"  }  } |

Remarque : définition de la fonction indique en paramètre un tableau de taille 5 de variables type  char

**objet String =** manipuler les chaînes de caractères de façon plus naturelle et intuitive [l'objet String](https://www.arduino.cc/en/Reference/StringObject)

Intérêt : pas besoin de créer des fonctions de lecture de caractères mais utiliser celles de l'objet

Ex1 : String maChaine="Ceci est une chaîne de caractères";

Ex2 : String longueChaine=String(maChaine + " avec une phrase ajoutée !");

Fonction string : String(quelqueChose);  transforme le "quelqueChose" en format  String

Ex : String chaineNombre=String(123); => variable  chaineNombre  tableau 4 caractères : '1', '2', '3' et '\0'

Autres fonctions utiles : toLowerCase() ,  toUpperCase() ,  trim()  et  length()

|  |
| --- |
| void setup() {  Serial.begin(9600);  //on crée une chaîne avec des espaces, des majuscules et des minuscules  String maChaine=" Bonjour le monde Actuel ! \n\r";  String resultat=""; //on crée une chaîne qui va contenir les résultats  Serial.println(); //saute une ligne    resultat=maChaine;//copie maChaine dans resultat  resultat.toLowerCase(); //fonction qui passe tout en minuscules  Serial.println(resultat); //affiche le résultat  resultat=maChaine;//copie maChaine dans resultat  resultat.toUpperCase(); //fonction qui passe tout en majuscules  Serial.println(resultat); //affiche le résultat  resultat=maChaine; //copie maChaine dans resultat  resultat.trim(); //fonction qui enlève tous les espaces superflus en début et fin de chaîne(caractères 9, 10,11, 12, 13 et 32)  Serial.println(resultat);  int taille=maChaine.length(); //fonction qui renvoie le nombre de caractères de la chaîne dans l'objet String  Serial.println(taille);    }  void loop() {  } |

Remarque : bibliothèque Serial utilise objet  String avec Serial.print()  et  Serial.println()

Autres fonctions utiles pour analyser un train d'informations

* maChaine.startsWith("exemple");

teste si l'objet  String commence par caractères chaîne entre parenthèses / renvoie 1 si oui et 0 si non

* maChaine.endsWith("exemple");

teste si l'objet  String se termine par les caractères de la chaîne entre parenthèses.

* maChaine.substring(début,fin);

envoie la chaîne de caractères (en objet  String) entre la position "début" et "fin" de la chaîne. "fin"

si fon est omis, dans ce cas, la fonction renvoie tout le reste de la chaîne depuis la position "début"

Programme : analyse de notre code sur 5 caractères :

|  |
| --- |
| String reception = ""; //création de l'objet String reception  void setup() {  Serial.begin(9600); //on initialise la communication série  }  void loop() {  if (Serial.available()) {  while (Serial.available()) { //tant que des données sont en attente  char c = Serial.read(); // l’Arduino lit le caractère  reception += String(c); //et l’ajoute à l’objet reception  delay(10); //petite attente  }  reception.trim(); //on enlève le superflu en début et fin de chaine  reception = reception.substring(0, 5); // ne prend que les caractère de 0 à 4 (soit 5 caractères)  decodage(reception); // on appelle la fonction de décodage  reception = "";  }  }  //fonction pour décoder la réception  void decodage(String s) {  Serial.println("\* Etat des LED \*");//titre  for (int p = 0; p < 5; p++) { //on parcourt le tableau  Serial.print("LED "); //on écrit le mot LED  Serial.print(p + 1); //on écrit le numéro de la LED (position dans taleau +1)  Serial.print(" "); //on fait un espace  // on utilise la fonction switch qui fonctionne aussi avec des caractères (puisque ce sont des nombres !)  switch (s.charAt(p)) { // fonction string.charAt(pos) renvoie le caractère à la position "pos" de l'objet String  case 'a':// on teste si le code est 'a'  Serial.println("allumee"); //si oui on écrit "allumee"  break;  case 'e': // sinon on teste si le code est 'e'  Serial.println("eteinte"); //si oui on écrit "éteinte"  break;  case 'c': //sinon on teste si le code est 'c'  Serial.println("clignote"); //si oui on écrit "clignote'"  break;  default://sinon c'est que ça ne correspond à rien  Serial.println("erreur de code"); // on inscrit donc "erreur de code"  }  }  Serial.println();// saut de ligne  } |

Remarque : maChaine.charAt(p);  renvoie caractère (type char) situé position  p  de maChaine.

* **Pilotez des LED avec un programme de codage/décodage**

**montage photo7\_1** pins 2 à 6 pour LED de 1 à 5 / laisser pins 0 et 1 libres utilisés pour communication série

**principes programmation**

1 fonction par étape du programme (pas de paramètre)

1. lecture-traduction données envoyées et maj tableau comportements LEDs (allumées, éteintes, clignotantes)
2. temps écoulé et changement éventuel du tableau d'état des LED en fonction du tableau comportements
3. allumage des LED en fonction du tableau d'états

2 tableaux distincts

* **tableau des comportements des LED** : stocke si LED doit s'allumer, s'éteindre ou clignoter (3valeurs=>char)
* **tableau des états des LED** : stocke si LED, à un instant donné, est allumée ou éteinte (2 valeurs=>booléen)

(1LED clignotante passe de allumée à éteinte à chaque écoulement du temps défini)

Variable globale : stockage du temps et une variable qui définit le temps de clignotement.

|  |
| --- |
| int pins[5] = {2, 3, 4, 5, 6};  //tableau qui stocke les pins des LED  char comportements[5] = {0, 0, 0, 0, 0};  //tableau des comportements. 0 pour éteint, 1 pour allumé, 2 pour clignotant  boolean etats[5] = {0, 0, 0, 0, 0};  // tableau des états. 0 pour éteint, 1 pour allumé  unsigned long tempsDepart; // variable pour stocker le temps à un instant donné  int duree = 200; // variable pour stocker la durée de clignotement en ms  void setup() {  Serial.begin (9600);  // initialisation des LED et mise à LOW  for (int l = 0; l < 5; l++) {  pinMode(pins[l], OUTPUT);  digitalWrite(pins[l], LOW);  }  //titre de présentation (c'est toujours plus sympa !)  Serial.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  Serial.println("\* Gestion de LED \*");  Serial.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  Serial.println("> start");//message de début de programme  tempsDepart = millis(); //initialisation du temps de départ sur le temps actuel  }  void loop() {  decodage(); //appel de la fonction de décodage  timer(); // appel de la fonction de gestion du temps  allumage(); // appel de la fonction d'allumage des LED  }  //------------------------fonctions  void decodage(){  if (Serial.available()){ // teste si il reste des données en attente  //code pour le décodage  //la mise à jour du tableau des comportements  //la mise à jour du tableau des états  //et l'affichage sur le moniteur pour contrôle  }  }  void timer(){  unsigned long tempsActuel=millis();  if (tempsActuel-tempsDepart>=duree){ //teste si le temps est écoulé  //code pour modifier le tableau d'état des LED  //pour chaque LED qui a un comportement clignotant  }  }  void allumage(){  // code pour allumer chaque LED  // en fonction du tableau d'état des LED  } |

**Programme :**

|  |
| --- |
| int pins[5] = {2, 3, 4, 5, 6}; //tableau qui stocke les pins des LED  char comportements[5] = {0, 0, 0, 0, 0};  //tableau des comportements. 0 pour éteint, 1 pour allumé, 2 pour clignotant  boolean etats[5] = {0, 0, 0, 0, 0};  // tableau des états. 0 pour éteint, 1 pour allumé  unsigned long tempsDepart; // variable pour stocker le temps à un instant donné  int duree = 200; // variable pour stocker la durée de clignotement en ms  void setup() {  Serial.begin (9600); // initialisation de la communication série  // initialisation des LED et mise à LOW  for (int l = 0; l < 5; l++) {  pinMode(pins[l], OUTPUT);  digitalWrite(pins[l], LOW);  }  //titre de présentation (c'est toujours plus sympa !)  Serial.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  Serial.println("\* Gestion de LED \*");  Serial.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  Serial.println("> start");//message de début de programme  tempsDepart = millis(); //initialisation du temps de départ sur le temps actuel  }  void loop() {  decodage(); //appel de la fonction de décodage  timer(); // appel de la fonction de gestion du temps  allumage(); // appel de la fonction d'allumage des LED  }  //------------------------fonctions  void decodage() {  if (Serial.available()) {// teste s'il reste des données en attente  String reception = "";  while (Serial.available()) { //tant que des données sont en attente  char c = Serial.read(); // lecture  reception += String(c); //on ajoute à l'objet reception  delay(10); //petite attente  }  reception.trim(); //on enlève le superflu en début et fin de chaîne  reception = reception.substring(0, 5); // ne prend que les caractères de 0 à 4 Serial.print ("> Etat : ");  //boucle de test par caractère  for (int c = 0; c < 5; c++) {  switch (reception.charAt(c)) {  case 'a':  Serial.print("\tON");  comportements[c] = 1; //mise à jour du tableau des comportements  break;  case 'e':  Serial.print("\tOFF");  comportements[c] = 0; //mise à jour du tableau des comportements  break;  case 'c':  Serial.print("\tCLI");  comportements[c] = 2; //mise à jour du tableau des comportements  break;  default :  comportements[c] = 0; //mise à jour du tableau des comportements par défaut à 0  Serial.print("\t???");  }  }  Serial.println(); // saut de ligne  // boucle de maj du tableau des états en fonction du tableau des comportements  for (int l = 0; l < 5; l++) {  if (!comportements[l]) {  etats[l] = 0;  }  else {  etats[l] = 1; // on allume aussi les lED clignotantes  }  }  }  }  void timer() {  unsigned long tempsActuel = millis(); //récupération du temps Arduino  if (tempsActuel - tempsDepart >= duree) { //test si temps écoulé  for (int l = 0; l < 5; l++) {  if (comportements[l] == 2) { //test si comportement clignotant  etats[l] = !etats[l]; //passe de 0 à 1 ou de 1 à 0  }  }  tempsDepart = tempsActuel; //initialisation du temps de départ  }  }  void allumage() {  for (int l = 0; l < 5; l++) {  digitalWrite(pins[l], etats[l]); //allumage LEDs en fct du tableau des Etats  }  } |

Remarque :programme de ce type peut permettre de piloter des LED, mais aussi moteurs, servos

**Évolutions possibles**

* ajouter un code pour faire varier la vitesse de clignotement depuis le moniteur.
* faire varier la vitesse de clignotement grâce à un potentiomètre.
* préparer des séquences d'allumages qui peuvent être appelées depuis le moniteur par un codage

(de droite à gauche, de gauche à droite, tout clignote, on allume une, puis deux, puis trois...)

(tout en gardant la possibilité de commander chaque diode).

* créer une version du jeu MOTUS à 5 lettres (pas mal de boulot mais enrichissant)
  + Arduino choisit un mot au hasard dans un tableau
  + Joueur proposez un mot de 5 lettres
  + diodes s'allument si une lettre au bon endroit / s'éteint si pas dans le mot / clignote mal placée

**8\_ MATRICE DE LEDs** matériel : 9 LED et trois résistances 220Ω

Rappel : pilotage des LEDs par pins numériques … mais limités à 14. +6 avec analogique.

Max 20 ! Et encore en ne faisant rien d’autre (afficher infos série, capter infos par CAN, recevoir état bouton...

Matériel : matrice de LED 4x4, 4 lignes, 4 colonnes **photo8\_1**

Objectif : commander 16 LEDs avec seulement 8 pins (1 par ligne et 1 par colonne)

Formule : nb pins = nbLignes + nbColonnes / nb LED pilotables = nbLignes \* nbColonnes

| Nombre de LED à piloter | Nombre de pins nécessaires |
| --- | --- |
| 1 | 2 |
| 4 | 4 |
| 9 | 6 |
| 16 | 8 |
| 25 | 10 |

**Montez une matrice de 3x3 LED en carré**

Schéma de connexion : **photo8\_2**

* LED connectées : par lignes de 3 / par colonne de 3 / chaque LED appartient à 1 paire ligne/colonne unique.
* 3 résistances sont placées à l'entrée des lignes (pour protéger les LED).
* Le courant qui peut allumer une LED va circuler d'une ligne vers une colonne.

**Allumez une seule LED**

Principe :

* placer toutes les sorties (2 à 7) en mode  en position LOW= rien ne s'allume (le courant ne circule nulle part)
* si sortie 2 en HIGH , allume 3 LED = courant circuler entre cette sortie et les autres en respectant 2 règles :
* Le courant ne revient pas sur une branche du circuit où il circule déjà en sens inverse.
* Le courant ne peut circuler dans une LED  que dans un seul sens.

Rappel : courant ne circule que si différence de potentiel : HIGH  vers  LOW mais pas LOW->LOW, ni HIGH->HIGH

* agir 2 pins (le 2 et le 5) LED en bas à gauche.

Procédure :

* pins 2, 3 et 4 à LOW (on pourrait tout mettre en HIGH, mais on économise le courant).
* pins 5, 6, 7 à HIGH (rien ne s'allume car le courant ne peut pas traverser une LED dans le mauvais sens).
* place le pin 2 à HIGH (rien ne se passe encore car pas de circulation de HIGH vers HIGH).
* place le pin 5 à LOW (le courant passe du pin 2 vers le pin 5, ce qui allume la diode en bas à gauche).
* pour éteindre, place pin 2 à LOW et pin 5 à HIGH (pour préparer l'éventuel allumage d'une autre LED).

Programme : 2=ligne 0 / pin 3=ligne 1 / pin 4=ligne 2 / pin 5=colonne 0 / pin 6=colonne 1 / pin 7=colonne 2

|  |
| --- |
| void setup() {  //initialisation des pins en OUTPUT et mise à l'état LOW  for (int l=2;l<8;l++){  pinMode(l,OUTPUT);  digitalWrite(l,LOW);  }  //on passe les pins de colonne à l'état HIGH  for (int l=5;l<8;l++){  digitalWrite(l,HIGH);  }  }  void loop() {  // on allume la LED ligne 0 colonne 0  digitalWrite(2,HIGH);  digitalWrite(5,LOW);  delay(500);  // on l'éteint  digitalWrite(2,LOW);  digitalWrite(5,HIGH);  delay(500);  } |

*exercice* : faire clignoter la LED en bas à gauche (sans utiliser delay()) à l'aide de la fonction millis())

|  |
| --- |
| boolean etat = 1; // boléen pour l'état de la LED  unsigned long tpsDep = millis(); // variable pour le temps de départ  void setup() {  //initialisation des pins en OUTPUT et mise à LOW  for (int l = 2; l < 8; l++) {  pinMode(l, OUTPUT);  digitalWrite(l, LOW);  }  //on passe les pins de colonne à HIGH  for (int l = 5; l < 8; l++) {  digitalWrite(l, HIGH);  }  }  void loop() {  unsigned long tpsAct = millis(); // variable pour le temps actuel  if (tpsAct - tpsDep > 500) { // si le temps dépasse 500ms  etat = !etat; //on inverse l'état de la LED  tpsDep = tpsAct; //on réinitialise le temps de départ  digitalWrite(2, etat); //on place le pin ligne de la LED comme etat  digitalWrite(5, !etat); //on place le pin colonne de la LED à l'opposé d'état  }  } |

**Allumez plusieurs LED :**

Physiologie :persistance rétinienne : sorte d'anti-capacité de notre œil.

= quand lumière perçue, image (point lumineux) reste un court moment comme incrusté sur l’œil.

Rappel : une LED qui clignote trop vite est perçue comme restant allumée.

Principe : allumer (ou non), puis éteindre chaque LED les unes après les autres … très rapidement

= même si une LED reste éteinte pendant qu'on traite les autres -> œil nous trompe et voit LED allumée

Programme : allumer LEDs bas-gauche + haut-droite

* toutes les autres éteintes.
* connexion en matrice ne peut allumer qu'une LED à la fois => allumer-éteindre rapidement avant la suivante
* créer un fonction  affiche() qui gèrera l'état des pins.

|  |
| --- |
| void setup() {  //initialisation des pins en OUTPUT et mise à l'état LOW  for (int l = 2; l < 8; l++) {  pinMode(l, OUTPUT);  digitalWrite(l, LOW);  }  //on passe les pins de colonne à l'état HIGH  for (int l = 5; l < 8; l++) {  digitalWrite(l, HIGH);  }  }  void loop() {  affiche();  }  // fonction d'affichage  void affiche(){  //allumage de la LED en bas à gauche  digitalWrite(2,HIGH);  digitalWrite(5,LOW);  delay(2); // petit délai pour l'oeil  //on l'éteint  digitalWrite(2,LOW);  digitalWrite(5,HIGH);  //allumage de la LED en haut à droite  digitalWrite(4,HIGH);  digitalWrite(7,LOW);  delay(2); // petit délai pour l'oeil  //on l'éteint  digitalWrite(4,LOW);  digitalWrite(7,HIGH);  } |

Remarque :

* delay(2) : pour que la lumière imprime la rétine + pour laisser le temps à la LED de s'allumer

= temps entre allumage et extinction très court = temps minimum pour briller correctement : faire des tests

* mais long coder allumage toutes les LED => tableau d'état à parcourir très vite chaque allumage/extinction

exemple : pour LED bas-gauche + haut-droite :

|  |
| --- |
| boolean matrice[3][3]={  0,0,1,  0,0,0,  1,0,0  }; |

Mais besoin de définir clairement les pins de lignes et de colonnes

|  |
| --- |
| byte ligne[3]={4,3,2};  byte colonne[3]={5,6,7}; |

Conseil : privilégier une forme de tableau lisible pour vous + faire correspondre au montage.

Remarque : le type 'byte' (on peut utiliser aussi le type 'int') prend moins de place mémoire

Synthèse : maintenant 3 tableaux

* parcourir le tableau matrice avec 2 variables l (ligne) et c (colonne) = positions dans chaque tableau de pins
* suffit ensuite d'allumer (ou non) selon le tableau matrice, puis éteindre pour passer à la LED suivante

|  |
| --- |
| boolean matrice[3][3]={  0,0,1,  0,0,0,  1,0,0  };  byte ligne[3]={4,3,2};  byte colonne[3]={5,6,7};  void setup() {  //initialisation des pins en OUTPUT et mise à l'état LOW  for (int l = 2; l < 8; l++) {  pinMode(l, OUTPUT);  digitalWrite(l, LOW);  }  //on passe les pins de colonne à l'état HIGH  for (int l = 5; l < 8; l++) {  digitalWrite(l, HIGH);  }  }  void loop() {  affiche(); //appel de la fonction d'affichage  }  // fonction d'affichage  void affiche(){  for (byte l=0;l<3;l++){//on parcourt les lignes  for (byte c=0;c<3;c++){//puis chaque case de la ligne (colonne)  digitalWrite(ligne[l],matrice[l][c]); //on allume ou non en fonction du tableau matrice  digitalWrite(colonne[c],LOW);  delay(2); // attente pour l'oeil  digitalWrite(ligne[l],LOW); //on éteint  digitalWrite(colonne[c],HIGH); //et on place à l'état HIGH pour la suivante  }  }  } |

Remarque :

* fonction  digitalWrite() appelée même si le tableau  matrice montre un état à  0
* appel 4\*9=36 fois à la fonction  digitalWrite() à chaque appel de la fonction  affichage()

Amélioration :

|  |
| --- |
| boolean matrice[3][3] = {  0, 0, 1,  0, 0, 0,  1, 0, 0  };  byte ligne[3] = {4, 3, 2};  byte colonne[3] = {5, 6, 7};  void setup() {  //initialisation des pins en OUTPUT et mise à l'état LOW  for (int l = 2; l < 8; l++) {  pinMode(l, OUTPUT);  if (l > 1 && l < 4)  digitalWrite(l, LOW);  else  digitalWrite(l, HIGH);  }  }  void loop() {  affiche(); //appel de la fonction d'affichage  }  // fonction d'affichage  void affiche() {  for (byte l = 0; l < 3; l++) { //on parcourt les lignes  digitalWrite(ligne[l], HIGH); //on prépare la ligne à l'affichage  for (byte c = 0; c < 3; c++) { //on parcourt chaque case de la ligne (colonne)  if (matrice[l][c]) { //on teste si affichage dans matrice  digitalWrite(colonne[c], LOW); //on allume  delay(2); //avec un delai  digitalWrite(colonne[c], HIGH); //on éteint  }  }  digitalWrite(ligne[l], LOW); //on arrête la ligne  }  } |

Commentaires : digitalWrite() appelée que 6 fois (2\*par ligne) + 2 fois/case allumée (4 fois ici) = 10x total

(allumer toutes les LED : modifier tableau matrice => appel fonction  digitalWrite()  6+9\*2=24 fois

**Exercice : gérez la matrice de LED avec un tableau à simple entrée**

Principe :

* ne plus parcourir un tableau à double entrée mais un tableau de 9 cases (donc à simple entrée)

= chaque case correspond à une LED du montage = première case haut-gauche, dernière bas- droite

012345678 = ligne1 LED0 LED1 LED2 / ligne 2 LED3 LED4 LED5 / ligne 3 LED6 LED7 LED8

* stockage des états plus sur un tableau de booléens = char matrice[10]="eeaeeeaee"; e=éteint, a= allumé

ne pas oublier de déclarer un caractère de plus pour le caractère  NULL.

* fonction  affiche() parcourt cette chaîne => n’allume que si caractère de position donnée = "a"
* garde tableaux  ligne[]  et  colonne[]  pour le repérage des pins

Rappel maths :

* si "valeur" entier (char,int), x un autre entier => "valeur/x" entier tronqué à la valeur inférieure (7/3=2)
* si "valeur" entier (char,int), x un autre entier => "valeur%n" reste de la division euclidienne (7%3=1)

Intérêt : permet de repérer ligne&colonne LED dans tableau matrice

ex : caractère en position 7 du tableau en ligne (cases 012345678) :

* numéro ligne dans matrice LED est 7/3=2 (3 caractères par ligne) = ligne 2
* numéro colonne dans la matrice de LED est 7%3=1 = colonne 1

**Créez des séquences d'affichage**

Programme : afficher une matrice différente toutes les secondes (gestion du temps + utilisation des tableaux)

Exemple : séquence allume chaque LED successivement en donnant l'impression qu'une lumière "dessine" un S

|  |
| --- |
| //tableau des matrices  char matrices[10][10] = {  "aeeeeeeee",  "eaeeeeeee",  "eeaeeeeee",  "eeeeeaeee",  "eeeeaeeee",  "eeeaeeeee",  "eeeeeeaee",  "eeeeeeeae",  "eeeeeeeea",  "eeeeaeeee"  };  byte pos = 0; //variable de position dans le tableaud des matrices  int vitesse=1000; // variable pour la vitesse en ms entre chaque matrice  char matriceAct[10]="eeeeeeeee"; //tableau de matrice en cours  unsigned long tpsDep = millis(); //initialisation du temps  byte ligne[3] = {4, 3, 2}; //stockage des pins ligne  byte colonne[3] = {5, 6, 7}; // stockage des pins colonne  void setup() {  //initialisation des pins en OUTPUT et mise à l'état LOW ou HIGH  for (byte l = 2; l < 8; l++) {  pinMode(l, OUTPUT);  if (l > 1 && l < 4)  digitalWrite(l, LOW);  else  digitalWrite(l, HIGH);  }  }  void loop() {  unsigned long tpsAct = millis();  if (tpsAct - tpsDep > vitesse) { //teste si temps écoulé  copie(pos); //appel de la fonction de mise à jour de la matrice en cours  pos = (pos + 1) % 10; //avancée curseur de position : retour à 0 si > 10  tpsDep = tpsAct; //réinitialisation du temps  }  affiche(); //appel de la fonction d'affichage  }  // fonction d'affichage  void affiche() {  for (byte p = 0; p < 9; p++) { //parcours du tableau matrice  if (matriceAct[p] == 'a') { //test si 'a'  //allumage  digitalWrite(ligne[p / 3], HIGH);  digitalWrite(colonne[p % 3], LOW);  delay(2);  //exctinction  digitalWrite(colonne[p % 3], HIGH);  digitalWrite(ligne[p / 3], LOW);  }  }  }  //fonction de copie de tableau en fonction de la position.  void copie(byte n) {  for (byte p = 0; p < 10; p++) {  matriceAct[p] = matrices[n][p];  }  } |

Remarque : pos=(pos+1)%10 = incrémenter une valeur & faire passer à 0 si elle dépasse une limite (10%10=0 !)

**Un alphabet matrice 3x3**

Programme : épeler un mot (écrit en minuscules) contenu dans un tableau de  char.

Principe :

* créer 26 lettres de l'alphabet dans tableau de matrices (=26 matrices)
* parcourir tableau du mot à épeler
* appelle la bonne matrice en fonction de la lettre

Astuce : une lettre=un code & lettres suivent donc codes aussi => si 'a'=97(ASCII) donc retrancher 97 si a=0.

|  |
| --- |
| //tableau des matrices  char matrices[26][10] = {  "aaaaeaaea", //a  "aeeaaeaae", //b  "aaaaeeaaa", //c  "eeaeaaeaa", //d  "aaaaaeaaa", //e  "aaaaaeaee", //f  "aaeaeaaaa", //g  "aeaaaaaea", //h  "eaeeaeeae", //i  "eaeeaeaae", //j  "aeaaaeaea", //k  "aeeaeeaaa", //l  "aaaaaaaea", //m  "aeeaeaaea", //n  "aaaaeaaaa", //o  "aaeaaeaee", //p  "eaaeaaeea", //q  "aaeaaeaea", //r  "eaaeaeaae", //s  "aaaeaeeae", //t  "aeaaeaaaa", //u  "aeaaeaeae", //v  "aeaaaaaaa", //w  "aeaeaeaea", //x  "aeaeaeeae", //y  "aaeeaeeaa" //z  };  char mot[10]="salut"; //mot à épeler  int pos = 0; //variable de position dans le mot  int vitesse=500; // variable pour la vitesse en ms entre chaque matrice  char matriceAct[10]="eeeeeeeee"; //tableau de matrice en cours  unsigned long tpsDep = millis(); //initialisation du temps  byte ligne[3] = {4, 3, 2}; //stockage des pins ligne  byte colonne[3] = {5, 6, 7}; // stockage des pins colonne  void setup() {  //initialisation des pins en OUTPUT et mise à l'état LOW ou HIGH  for (byte l = 2; l < 8; l++) {  pinMode(l, OUTPUT);  if (l > 1 && l < 4)  digitalWrite(l, LOW);  else  digitalWrite(l, HIGH);  }  }  void loop() {  unsigned long tpsAct = millis();  if (tpsAct - tpsDep > vitesse) { //teste si temps écoulé  char lettre=mot[pos]; //on lit la lettre du mot  copie(byte(lettre)-97); //appel de la fonction de mise à jour de la matrice en cours  pos = (pos + 1) % 10; //avancée du curseur de position dans le mot  tpsDep = tpsAct; //réinitialisation du temps  }  affiche(); //appel de la fonction d'affichage  }  // fonction d'affichage  void affiche() {  for (byte p = 0; p < 9; p++) { //parcours du tableau matrice  if (matriceAct[p] == 'a') { //test si 'a'  //allumage  digitalWrite(ligne[p / 3], HIGH);  digitalWrite(colonne[p % 3], LOW);  delay(2);  //exctinction  digitalWrite(colonne[p % 3], HIGH);  digitalWrite(ligne[p / 3], LOW);  }  }  }  //fonction de copie de tableau en fonction du code de la lettre.  void copie(byte n) {  for (byte p = 0; p < 10; p++) {  matriceAct[p] = matrices[n][p];  }  } |

Remarques : plus simple de faire alphabet sur une matrice 4x4 mais il existe des matrices toutes faites ;-)

**Pour aller plus loin :**

**2\_ digitalWrite() ou registre ?**

*registre* = adresse mémoire pourstockage informations de gestion de courants (entrées, sorties, LOW, HIGH...)

**Il existe en fait une méthode pour allumer directement les LED concernées en passant par les registres**

= affichage + rapide (et simultané), mais code moins lisible (voir [registres](https://www.arduino.cc/en/Reference/PortManipulation))

**2\_ Matrice de LEDs** : exemples [monochromatique](https://www.adafruit.com/products/870) et [trichromatique](https://www.adafruit.com/products/902)

Même même principe mais 1 puce permet de gérer l'affichage et Arduino commande la puce par protocole I2C

* puces I2C telles que la Pcf8574 ou la Mcp23017
* matrices peuvent être mises côte à côte ou dans d'autres configurations

Usage : affichages pour du texte ou des petits jeux du genre mini-Mario ou Tetris.

**9\_ MATRICE DE BOUTONS**

**10\_ ECRAN LCD**

**11\_ TELECOMMANDE INFRA-ROUGE**

**Utilisez le pilotage infrarouge =** Lire et interpréter informations envoyées par une télécommande IR

Matériel :

* capteur infrarouge (TSOP38238 ou TSOP1738) (voir cours 1 [montage du TSOP38238](https://openclassrooms.com/courses/programmez-vos-premiers-montages-avec-arduino/les-capteurs-electroniques#r-3309360))
* LED+résistance 220Ω
* télécommande infrarouge (celle de votre téléviseur, votre chaîne hifi, votre appareil photo...)

**Capteur infrarouge** capteurs de type TSOP permettent de capter un signal infrarouge

Attention : certaine fréquence nécessaire = 38 kHz pour les deux TSOP

Connexions : TSOP ont 3 broches : ground / +5V / pin numérique en mode input (vérifier sur datasheet)

**Exemple** : TSOP38238 : 1 = pin numérique pour la lecture / 2 = ground / 3 = +5V **photob\_1**

**Tester si le capteur est fonctionnel**

Montage simple (sans programmation) **photob\_2**

* lorsqu'un TSOP reçoit signal infrarouge, passe sortie à LOW
* connecter une LED (visible à l'œil nu <> LED IR) s'allume lorsque TSOP reçoit signal

Test : si LED clignote avec télécommande télé = TSOP correctement connecté

**Décoder un signal infrarouge**

Problème : il existe des normes constructeurs de communications IR = pas les mêmes codes pour même action

=> travailler avec la même télécommande

Montage précédent : si connecte sortie (OUT) TSOP à pin numérique en mode INPUT = image du code envoyé

* télécommande envoie un signal pulsé à environ 38KHz (succession états hauts et bas sur période 26 µs)
* signal pulsé envoyé pendant un temps donné puis s'arrête = suite signaux émis&arrêts pendant un temps précis

= forme un code (qui peut être plus ou moins long en fonction des télécommandes).

* **besoin de lire chaque temps où le signal est pulsé et chaque temps où il ne l'est pas**

avantage capteur de type TSOP = filtre à 38Khz => tenter de capter un signal haut (et non de signaux haut-bas) <> une photodiode capte tous les signaux émis !

**Programme 1 = succession signal pulsé puis repos (mais on ne connait pas les temps)**

|  |
| --- |
| const int pin = 8; // pin de lecture du TSOP  const int nbMax = 32; // nombre de lectures  int tableau[nbMax][2]; // tableau pour stocker les valeurs hautes et basses  void setup() {  pinMode(pin, INPUT); // mode INPUT pour le pin de lecture  Serial.begin(9600); //initialisation communication série  }  void loop() {  if (digitalRead(pin)) { //si état haut capté  lecture(); // appel de la fonction de lecture du code  affichage(); // appel de la fonction d'affichage du résultat  }  }  //fonction de lecture du code  void lecture() {  for (int t = 0; t < nbMax; t++) { //boucle pour le nombre de lectures  while (digitalRead(pin)) { // tant que état pulsé  tableau[t][0]++; // on incrémente la valeur du tableau  }  while (!digitalRead(pin)) { // puis tant que état non pulsé  tableau[t][1]++; // on incrémente l'autre valeur du tableau  }  }  }  // fonction d'affichage du résultat  void affichage() {  for (int i = 0; i < 2; i++) { // boucle pour état Haut, puis état Bas  for (int t = 0; t < nbMax; t++) { //lecture des valeurs  Serial.print(tableau[t][i]); //affichage  Serial.print ("\t"); //tabulation pour présentation  }  Serial.println(); //saut de ligne  }  } |

**Programme 2 : compare sur 10 mesures : temps exécution digitalRead() vs temps lecture registre du pin 8**

Besoin de calculer :

* temps réel écoulé entre chaque lecture (et donc chaque incrémentation) ;
* vitesse de lecture du pin (donc la vitesse que la fonction  digitalRead(pin)  prend pour s'exécuter !
  + digitalRead(pin) plusieurs instructions qui chacune demande un certain temps d’exécution
  + lire l'état d'un pin =utiliser une méthode plus proche du langage machine = lecture des registres

Registres : chaque information Arduino contenue dans cases mémoires

* chacune avec des adresses (une position dans la mémoire de microprocesseur) définies et fixes.
* faire appel à un registre (lecture ou écriture) = aller directement lire ou écrire dans cette case mémoire

= programme +rapide mais –lisible

|  |
| --- |
| const int pin = 8; // constante de pin lu  void setup() {  Serial.begin(9600); //initialisation communication série  pinMode(pin, INPUT); // mise en mode input du pin  }  void loop() {  calcule(); //appel de la fonction calcule  delay(500); // délai pour l'affichage  }  void calcule() {  int tableau[10][2]; //tableau de récupération des valeurs  for (int t = 0; t < 10; t++) { //boule pour 10 mesures  unsigned long tpsDep, tpsFin; // variables pour les temps  tpsDep = micros(); // temps de départ  digitalRead(pin); // fonction digitaRead()  tpsFin = micros(); // temps de fin  tableau[t][0] = tpsFin - tpsDep; // on met le résultat dans le tableau  tpsDep = micros(); // temps départ  boolean n = PINB & B00000001; // lecture du registre  tpsFin = micros(); // temps de fin  tableau[t][1] = tpsFin - tpsDep; // on met le résultat dans le tableau  }  //affichage du tableau des résultats  Serial.println("\*\*\*\*\*\*");  for (int i = 0; i < 2; i++) {  for (int t = 0; t < 10; t++) {  Serial.print(tableau[t][i]);  Serial.print("\t");  }  Serial.println();  }  Serial.println();  } |

Résultats : tps digitalRead() : 8 8 8 8 4 4 8 8 8 8 / registre : 4 0 4 4 4 4 4 0 0 4 #2 plus rapide

fonction micros() gourmande en temps & mesures que tranche de 4µs = résultats modulo 4

**Programme 3 : Utilisation lecture directe du registre en pin 8 + temporisation de lecture sur 25µs (ajustable)**

|  |
| --- |
| const int pin = 8; // pin de lecture du TSOP  const int nbMax = 64; // nombre de lectures  const int temporisation=25;  unsigned int tableau[nbMax][2]; // tableau stocke valeurs hautes et basses  void setup() {  pinMode(pin, INPUT); // mode INPUT pour le pin de lecture  Serial.begin(9600); //initialisation communication série.  }  void loop() {  if (digitalRead(pin)) { //si état haut capté  lecture(); // appel de la fonction de lecture du code  affichage(); // appel de la fonction d'affichage du résultat  }  }  //fonction de lecture du code  void lecture() {  for (int t = 0; t < nbMax; t++) { //boucle pour le nombre de lectures  while (PINB & B00000001) { // tant que état pulsé  tableau[t][0]++; // on incrémente la valeur du tableau  delayMicroseconds(temporisation);  }  while (!(PINB & B00000001)) { // puis tant que état non pulsé  tableau[t][1]++; // on incrémente l'autre valeur du tableau  delayMicroseconds(temporisation);  }  }  delay(200);  }  // fonction d'affichage du résultat  void affichage() {  Serial.println("\*");  for (int i = 0; i < 2; i++) { // boucle pour état Haut, puis état Bas  for (int t = 0; t < nbMax; t++) { //lecture des valeurs  Serial.print(tableau[t][i]\*temporisation); //affichage en microsecondes  Serial.print ("\t"); //tabulation pour présentation  tableau[t][i]=0; //effacement de la valeur pour prochaine lecture  }  Serial.println(); //saut de ligne  }  } |

Résultat : lecture un peu + fiable des codes télécommande.

Exploitation des codes ?

1. programme pour lire et reconnaître les boutons (+affiche la succession de temps à prendre en compte)
2. programme pour utiliser ces codes pour piloter un servo-moteur

**Détecter, stocker et reconnaître un signal infrarouge** A partir du programme précédent :

* décodage du signal télécommande et stockage dans un tableau
* comparaison du code du signal reçu par pin 8 aux codes des commandes déjà trouvées (20% marge d’erreur)
* si le code correspond à une commande déjà trouvée = on indique quelle touche appuyée
* sinon on ajoute une commande et on propose de la nommer (facultatif, mais plus joli).
* le code affiche aussi la suite des temps trouvés pour pouvoir les exploiter avec un simple copier/coller.

|  |
| --- |
| const int pin = 8; // pin de lecture du TSOP  const int nbMax = 32; // nombre maximum de lectures  const int nbMaxCom = 5; // nombre maximum de commandes stockées  String noms[nbMaxCom] = {""}; // tableau stocke noms des commandes (sur 5 char)  const int temporisation = 20; // temporisation de lecture des signaux pulsés  unsigned int tableau[nbMax \* 2] = {0}; // tableau pour stocker les temps lus  unsigned int commandes[nbMaxCom][nbMax \* 2] = {0};  // tableau pour stocker les temps des commandes reconnues  int nbCommande = 0; // numéro de commande en cours  void setup() {  pinMode(pin, INPUT); // mode INPUT pour le pin de lecture  Serial.begin(9600); //initialisation communication série.  }  void loop() {  Serial.println("Go..."); //signal l'attente d'un appui  while (PINB & B00000001);//attend un signal haut en boucle  lecture(); // appel de la fonction de lecture du code  affichage(); // appel de la fonction d'affichage  delay(1000); // attente d'une seconde  }  //fonction de lecture du code  void lecture() {  for (int t = 0; t < nbMax \* 2; t += 2) { //boucle nb lectures de 2 en 2  while (PINB & B00000001) { // tant que état pulsé  tableau[t] += temporisation; // on incrémente la valeur du tableau  delayMicroseconds(temporisation); //attente  }  while (!(PINB & B00000001)) { // puis tant que état non pulsé  tableau[t + 1] += temporisation; // on incrémente l'autre valeur du tableau  delayMicroseconds(temporisation);  }  }  }  // fonction d'affichage du résultat  void affichage() {  // on crée un affichage copiable sur la console  Serial.print("const unsigned int code[");  Serial.print(nbMax \* 2 + 1, DEC);  Serial.print("]={");  for (int t = 0; t < nbMax \* 2; t += 2) {  Serial.print(tableau[t], DEC);  Serial.print(", ");  Serial.print(tableau[t + 1], DEC);  Serial.print(", ");  }  Serial.println("0}");  compare();//on appelle la fonction de comparaison  //on efface le tableau pour la prochaine lecture  for (int t = 0; t < nbMax \* 2; t++) {  tableau[t] = 0;  }  }  //fonction de comparaison  int compare () {  boolean trouve = 0; // drapeau de commande trouvée  for (int c = 0; c < nbMaxCom; c++) { //boucle sur les commandes  int nbCor = 0; // variable de validation de comparaison  for (int t = 0; t < nbMax \* 2; t++) { // on parcourt les résultats  int tmoins = tableau[t] - tableau[t] \* 20 / 100; // valeur-20%  int tplus = tableau[t] + tableau[t] \* 20 / 100; // valeur+20%  if (commandes[c][t] > tmoins && commandes[c][t] < tplus && t > 0) {  //si dans la fourchette (sauf première valeur)  nbCor++; // on valide la comparaison  }  }  if (nbCor == nbMax \* 2 - 1) { //si tout est validé (sauf première valeur)  // On affiche la touche correspondante  Serial.print("Correspondance avec commande ");  Serial.print(c);  Serial.print(" = ");  Serial.println(noms[c]);  trouve = 1; // on indique que la commande est déjà existante  }  }  if (!trouve && (nbCommande < nbMaxCom)) {  //si commande non existante et encore possible d'en ajouter  //on crée la commande en copiant le tableau des résultats  for (int t = 0; t < nbMax \* 2; t++) {  commandes[nbCommande][t] = tableau[t];  }  //on affiche qu'une nouvelle commande est trouvée  Serial.println("ajout commande");  Serial.println("Donner un nom (5 char max) :");  //et on saisit son nom  while (!Serial.available())  noms[nbCommande] = "";  for (int t = 0; t < 5; t++) {  char ch = Serial.read();  if (ch > 32)  noms[nbCommande] += String(ch);  delay(10);  }  //on vide le cache de réception  while (Serial.available()) {  Serial.read();  delay(10);  }  nbCommande++; //on incrémente la commande en cours  }  Serial.println();  } |

Attention :

* ne fonctionne qu'avec le pin 8 en lecture. Pour le modifier, bien connaitre les registres.
* nb lectures & nb commandes enregistrables limité car le stockage de tous ces nombres prend de la place
* OK avec un TSOP 38238 mais certaines télécommandes ne sont pas reconnues avec un TSOP1738

**TP : Piloter un servo-moteur avec une télécommande infrarouge et un Arduino**

Principe : piloter un servo-moteur, grâce à 5 touches de votre télécommande .

* utiliser bibliothèque Servo
* touches : 1=servo à 0° / 2=179° / 3=90° / 4=vers 179° avec pas de 1° / 5=vers 0° avec un pas de 1°

Conseils :

* choisir une télécommande qui répond au programme précédent.
* copier/coller les codes correspondant aux touches que vous allez utiliser d’après programme précédent
* pour reconnaissance du code, s’inspirer de la fonction  compare() du programme précédent.
* préférer créer des fonctions

**TP : Correction**

|  |
| --- |
| #include "Servo.h" // on ajoute la bibliothèque Servo  const int pinServo = 2; // constante de pin pour le servomoteur  Servo monServo; // on définit l'objet Servo nommé monServo  //tableau qui stocke les codes : 5 codes de 65 valeurs  // /!\ Attention il correspond aux valeurs de ma télécommande  //Pour la vôtre, il faut utiliser le programme "Décodeur infrarouge"  //et créer ce tableau !  const unsigned int codes[5][65] = {  0, 8620, 4240, 580, 1580, 560, 1580, 580, 500, 560, 500, 560, 1580, 580, 1580, 560, 500, 580, 480, 580, 500, 580, 480, 580, 500, 560, 500, 580, 480, 580, 500, 560, 500, 560, 500, 580, 500, 560, 500, 580, 1560, 580, 1580, 580, 1580, 560, 500, 560, 500, 580, 500, 560, 1580, 580, 1580, 560, 500, 560, 500, 580, 500, 560, 1580, 580, 0,  0, 8580, 4280, 540, 1620, 540, 1600, 540, 540, 540, 520, 540, 1600, 560, 1600, 540, 520, 540, 540, 540, 520, 540, 520, 560, 520, 540, 520, 540, 520, 560, 520, 540, 520, 540, 520, 560, 520, 540, 520, 540, 1620, 540, 1600, 540, 540, 540, 520, 540, 520, 560, 1600, 540, 1600, 540, 1620, 540, 520, 540, 540, 520, 1620, 540, 1600, 560, 0,  0, 8620, 4240, 580, 1580, 560, 1580, 580, 500, 560, 500, 560, 1580, 580, 1580, 580, 480, 580, 480, 580, 500, 580, 480, 580, 500, 560, 500, 560, 500, 580, 500, 560, 500, 580, 480, 580, 1580, 580, 480, 580, 1580, 580, 1560, 580, 500, 560, 500, 560, 500, 580, 1580, 560, 500, 580, 1580, 560, 500, 580, 480, 580, 1580, 580, 1580, 560, 0,  0, 8620, 4240, 560, 1580, 580, 1580, 560, 500, 580, 480, 580, 1580, 580, 1580, 560, 500, 580, 480, 580, 500, 560, 500, 560, 500, 580, 500, 560, 500, 560, 500, 580, 500, 560, 500, 580, 1580, 560, 500, 560, 1600, 560, 1580, 580, 1560, 580, 500, 580, 480, 580, 500, 560, 500, 560, 1580, 580, 500, 560, 500, 580, 500, 560, 1580, 580, 0,  0, 8620, 4240, 580, 1580, 560, 1580, 580, 500, 560, 500, 560, 1580, 580, 1580, 560, 500, 580, 500, 560, 500, 560, 500, 580, 500, 560, 500, 580, 480, 580, 500, 560, 500, 560, 520, 560, 500, 560, 1580, 580, 1580, 560, 1580, 580, 1580, 560, 500, 580, 500, 560, 500, 580, 1560, 580, 500, 580, 480, 580, 480, 580, 500, 560, 1580, 580, 0  };  //Tableau des noms pour chaque code  String noms[5] = {"90 deg", "179 deg", "0 deg", "Vers 179 deg", "Vers 0 deg"};  const int pin = 8; // pin de lecture du TSOP  const int nbMax = 32; // nombre maximum de lectures  const int temporisation = 20; // temporisation de lecture des signaux pulsés  unsigned int tableau[nbMax \* 2] = {0}; // tableau pour stocker les temps lus  int posServo = 90; // variable de position du servomoteur  int ajServo = 0; // variable de déplacement du servomoteur  int nbCode = 0; // numéro de code en cours  unsigned long tpsDep = millis(); // initialisation du temps  void setup() {  monServo.attach(pinServo); // attache l'objet monServo à pinServo  pinMode(pin, INPUT); // mode INPUT pour le pin de lecture  Serial.begin(9600); //initialisation communication série.  }  void loop() {  while (PINB & B00000001) // tant que aucun signal  bougeServo(); // on gère le servomoteur  //sinon  lecture(); // appel de la fonction de lecture du code  }  //fonction de lecture du code  void lecture() {  for (int t = 0; t < nbMax \* 2; t += 2) { //boucle pour nb de lectures de 2 en 2  while (PINB & B00000001) { // tant que état pulsé  tableau[t] += temporisation; // on incrémente la valeur du tableau  delayMicroseconds(temporisation); //attente  }  while (!(PINB & B00000001)) { // puis tant que état non pulsé  tableau[t + 1] += temporisation; // on incrémente l'autre valeur du tableau  delayMicroseconds(temporisation);  }  }  delay(500); //délai pour ne plus rien recevoir  compare(); // appel de la fonction de comparaison  positionServo(); // appel de la fonction de positionnement du servomoteur  }  //fonction de comparaison  int compare () {  boolean trouve = 0; // drapeau de commande trouvée  for (int c = 0; c < 5; c++) { //boucle sur les codes  int nbCor = 0; // variable de validation de comparaison  for (int t = 0; t < nbMax \* 2; t++) { // on parcourt les résultats  int tmoins = tableau[t] - tableau[t] \* 20 / 100; // valeur-20%  int tplus = tableau[t] + tableau[t] \* 20 / 100; // valeur+20%  if (codes[c][t] > tmoins && codes[c][t] < tplus && t > 0) {  //si dans la fourchette (sauf première valeur)  nbCor++; // on valide la comparaison  }  }  if (nbCor == nbMax \* 2 - 1) { //si tout est validé (sauf première valeur)  // On affiche la touche correspondante  Serial.print("Commande ");  Serial.print(c);  Serial.print(" => ");  Serial.println(noms[c]);  nbCode = c; // on indique le code en cours  trouve = 1; // on indique que la commande a été trouvée  }  }  if (!trouve) { //si commande non trouvée  Serial.println("Commande inconnue...");  }  //on vide le tableau de lecture  for (int t = 0; t < nbMax \* 2; t++) {  tableau[t] = 0;  }  }  //Fonction de positionnement du servomoteur  void positionServo() {  switch (nbCode) { //en fonction du code  case 0: // on place le servo à 90°  ajServo = 0;  posServo = 90;  break;  case 1: // on place le servo à 179°  ajServo = 0;  posServo = 179;  break;  case 2: // on place le servo à 0°  ajServo = 0;  posServo = 0;  break;  case 3: // on met la variable de déplacement à 1°  ajServo = +1;  break;  case 4: // on met la variable de déplacement à -1°  ajServo = -1;  break;  }  }  //fonction de déplacement du servomoteur  void bougeServo() {  unsigned long tpsAct = millis(); //temps actuel  if (tpsAct - tpsDep > 50) { //si 50ms passées  posServo += ajServo; // on déplace le servo  tpsDep = tpsAct; //on réinitialise le temps  }  if (posServo <= 0 || posServo >= 179) // si servo hors limite  ajServo = 0; // on stoppe le déplacement  monServo.write(posServo); // on envoie la position du servomoteur  } |

Remarque : il existe une bibliothèque IRremote

* consulter [cette présentation de la bibliothèque IRemote](https://electroniqueamateur.blogspot.fr/2013/01/clonage-dune-telecommande-infrarouge.html) pour cloner télécommande IE
* informations sur le site d'Adafruit sur ce [tutoriel](https://learn.adafruit.com/ir-sensor)

**12\_ SHIELD ETHERNET ET LE LECTEUR DE CARTE SD**

Matériel : [shield Ethernet Arduino](https://www.arduino.cc/en/Main/ArduinoEthernetShield), + carte micro-SD

**Le shield Ethernet** shield internet = carte qui s'enfiche sur toutes les bornes de l'Arduino (ex : shield Ethernet)

Remarques : **photob\_1 / photob\_2**

* il existe des cartes Arduino ethernet
* le shield ethernet propose un lecteur de carte SD mais pas la carte micro-SD (256Mo suffisent)

Shield Internet ?

* basé sur puce  Wiznet W5100 qui échanges de données réseau : 16 Ko mémoire, 4 connexions (UDP,TCP)
* communique grâce au bus SPI (Serial Peripheral Interface) protocole d'échange données en full-duplex

= envoyer informations sous forme de 1 et 0 appelés signaux logiques.

* bus SPI utilise 4 signaux logiques :
* SCLK — Serial Clock, Horloge (généré par le maître)
* MOSI — Master Output, Slave Input (généré par le maître)
* MISO — Master Input, Slave Output (généré par l'esclave)
* SS — Slave Select, Actif à l'état bas (généré par le maître)
* utilise les pins 10, 11, 12 et 13 de l’Arduino. / même bus avec carte SD + le pin 4 de l'Arduino.

= restent donc pins 0 et 1 moniteur série + pins analogiques (A0 à A5) et 7 pins numériques.

Prise RJ45 **photob\_3** : 8 contacts / montée en mode croisé ou droit (utilisé pour le shield)

UDP et TCPshield Arduino travaille en TCP comme en UDP

Protocoles = règles d'échange comprenant codes début/fin, type d'infos, mode échange, codage

**encapsulation des données =** envoi de données avant/après (parfois pendant) échange en plus du message

* **protocole UDP : # carte postale :** 
  + émission sans prévenir récepteur et sans vérifier réception
  + pas d’information sur émetteur sauf IP donc ne peut répondre après lecture
* **Le protocole TCP : # téléphone :** liaison fixe entre l'émetteur et le récepteur = active dans les 2 sens
  + récepteur peut vérifier si données reçues sont bien celles émises avec check mathématique
  + sinon peut demander nouvel envoi à l'émetteur

Exemples : protocole http / FTP / POP3 et IMAP / SMTP

Remarque : Arduino=micro-ordinateur = peu performant (>30’ pour envoyer photo hires)

IP Internet Protocol : réseaux : 2 sortes : **réseau local=**Intranet/LAN ou **réseau mondial=**Internet/WAN

* adresses IP locales= échanges sur votre réseau interne uniquement
* adresses IP mondiales = données par fournisseurs à notre box (seule accessible du WAN)

**lecteur de carte micro-SD photob\_4**

2 limites d’Arduino : manque de mémoire et que de la RAM / besoin mémoire pour gérer le réseau

**=> ajout d’une carte SD au shield ethernet +** bibliothèque associée bibliothèque SD (incluse par défaut)

connexion lecteur SD-Arduino en mode SPI sur 4 pins réservés  : 11=MOSI / 12=MISO / 13=CLK / 4=SS (ou CS)

format : carte à formater en FAT ou FAT32

**Programmez la communication entre l’Arduino et la carte micro-SD**

Exemples d’utilisation : voir dans IDE : Fichiers/Exemples/SD dans les menus.

Includes : 2 bibliothèques : carte SD + communication SPI : #include <SPI.h> #include <SD.h>

**Programme 1 : écrire 10x sur SD nb aléatoire pendant 10’’ +lire SD et afficher le tableau moniteur série**

Fichier : données dans un fichier = objet type File monFichier.txt (.ard pour changer) ATTENTION :fn 8 caractères

Initialisation : connexion carteSD=Arduino : dans le setup(), SD.begin(4); sur le pin 4

Attention :si utilise pas pin 10 par défaut pour com SPI, le laisser en mode OUTPUT et ne pas l'utiliser

Conseil : vérifier la connexion est bien établie avec la carte if (!SD.begin(4)){// si oui }

|  |
| --- |
| #include <SPI.h>  #include <SD.h>  File monFichier;  void setup(){  Serial.begin(9600); //débute la communication avec le moniteur série  Serial.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\nInitialisation...");  if (!SD.begin(4)){//teste la communication avec la carte(pin 4)  Serial.println("Communication impossible");  return; //stoppe le programme  };  Serial.println("Communication ok !");  }  void loop() {  } |

Random : utiliser un seed (graine) randomSeed(analogRead(A0));

Ouverture/écriture fichier : monFichier = SD.open("alea.ard",FILE\_WRITE) ;

SD.open(nomfichier, option) , FILE\_WRITE= lecture/écriture avec le curseur en fin de fichier.

Conseils : verifier existence du fichier et ne pas oublier de fermer le fichier : monFichier.close();

|  |
| --- |
| #include <SPI.h>  #include <SD.h>  File monFichier;  boolean messageOK=0;  void setup(){  Serial.begin(9600); //débute la communication avec le moniteur série  Serial.println("\*\*\*\*\*\*\*\*\*\nNombres hasardeux\n\*\*\*\*\*\*\*\*\*");  message("Initialisation");  if (!SD.begin(4)){//teste la communication avec la carte(pin 4)  message("Communication impossible");  return; //stoppe le programme  };  message("Communication ok !");  randomSeed(analogRead(A0)); //initialise nombre aléatoire  message("Ouverture du fichier");  if (!(monFichier = SD.open("alea.ard",FILE\_WRITE))){  message("Erreur de fichier");  return; //stoppe le programme  }  message("Fichier ouvert");  monFichier.close();  message("Fichier clos");  }  void loop() {  }  void message(String s){  if (messageOK){  Serial.println(s);  }  } |

**Programme 2 : boucle 0 à 9 qui pond un nombre aléatoire écrit dans le fichier à chaque tour et ferme fichier**

Écriture fichier : fonctions write(),  print() ou  println() précédés du nom de l'objet  File et d'un point :

* write() = écrit données de type byte ou char ou chaîne de caractères.
* print() = écrit données de type char, byte, int, long ou string. (un nombre est décomposé en caractères)
* println() : comme print, mais ajoute caractère de saut de ligne (\n ou 10) et retour chariot (\r ou 13).

|  |
| --- |
| #include <SPI.h>  #include <SD.h>  File monFichier;  boolean messageOK=1;  void setup(){  Serial.begin(9600); //débute la communication avec le moniteur série  Serial.println("\*\*\*\*\*\*\*\*\*\nNombres hasardeux\n\*\*\*\*\*\*\*\*\*");  message("Initialisation");  if (!SD.begin(4)){//teste la communication avec la carte(pin 4)  message("Communication impossible");  return; //stoppe le programme  };  message("Communication ok !");  randomSeed(analogRead(A0)); //initialise nombre aléatoire  message("Ouverture du fichier");  if (!(monFichier = SD.open("alea.ard",FILE\_WRITE))){ //tente d'ouvrir le fichier  message("Erreur de fichier");  return; //stoppe le programme  }  message("Fichier ouvert");  for (int n=0;n<10;n++){ // boucle d'écriture  int nAlea=random(0,10000); //tirage d'un nombre entre 0 et 10000  message("Ecriture de "+String(nAlea));  monFichier.print(nAlea); //écriture dans le fichier  }  monFichier.close();  message("Fichier clos");  }  void loop() {  }  void message(String s){  if (messageOK){  Serial.println(s);  }  } |

**Programme3 : ajout séparateur + recrée le fichier à chaque fois**

Remarque : si on regarde sur un lecteur de carte, tous les chiffres collés = prévoir mise en forme => ajout « : »

Effacer fichier s’il existe SD.exists(), SD.remove()

|  |
| --- |
| #include <SPI.h>  #include <SD.h>  File monFichier;  boolean messageOK=1;  void setup(){  Serial.begin(9600); //débute la communication avec le moniteur série  Serial.println("\*\*\*\*\*\*\*\*\*\nNombres hasardeux\n\*\*\*\*\*\*\*\*\*");  message("Initialisation");  if (!SD.begin(4)){//teste la communication avec la carte(pin 4)  message("Communication impossible");  return; //stoppe le programme  };  message("Communication ok !");  message("Teste si fichier existant");  if (SD.exists("alea.ard")){  message("Destruction fichier");  SD.remove("alea.ard");  }  randomSeed(analogRead(A0)); //initialise nombre aléatoire  message("Ouverture du fichier");  if (!(monFichier = SD.open("alea.ard",FILE\_WRITE))){ //tente d'ouvrir le fichier  message("Erreur de fichier");  return; //stoppe le programme  }  message("Fichier ouvert");  for (int n=0;n<10;n++){ // boucle d'écriture  int nAlea=random(0,10000); //tirage du nombre entre 0 et 10000  message("Ecriture de "+String(nAlea));  monFichier.print(nAlea); //écriture dans le fichier  monFichier.write(':');  }  monFichier.close();  message("Fichier clos");  }  void loop() {  }  void message(String s){  if (messageOK){  Serial.println(s);  }  } |

**Lire un fichier**

Soit charger tout le fichier, soit se déplacer dans le fichier caractère par caractère = bien pour la mémoire

Principe :

* ouvre fichier en lecture avec le curseur au début / prépare un tableau de caractères
* lit les caractères un par un en les ajoutant au tableau jusqu'à rencontrer les ":" de séparation
* affiche notre tableau de caractère (en ajoutant le '\0' qui permet de finir la chaîne
* recommence la lecture ainsi jusqu'à la fin du fichier.

Fonction : read() curseur automatique 1er caractère, mode lecture avec la constante  FILE\_READ

|  |
| --- |
| #include <SPI.h>  #include <SD.h>  File monFichier;  boolean messageOK = 1;  void setup() {  Serial.begin(9600); //débute la communication avec le moniteur série  Serial.println("\*\*\*\*\*\*\*\*\*\nNombres hasardeux\n\*\*\*\*\*\*\*\*\*");  message("Initialisation");  if (!SD.begin(4)) { //teste la communication avec la carte(pin 4)  message("Communication impossible");  return; //stoppe le programme  };  message("Communication ok !");  message("Teste si fichier existant");  if (SD.exists("alea.ard")) {  message("Destruction fichier");  SD.remove("alea.ard");  }  randomSeed(analogRead(A0)); //initialise nombre aléatoire  message("Ouverture du fichier");  if (!(monFichier = SD.open("alea.ard", FILE\_WRITE))) { //tente d'ouvrir le fichier  message("Erreur de fichier");  return; //stoppe le programme  }  message("Fichier ouvert");    //phase d'écriture  for (int n = 0; n < 10; n++) { // boucle d'écriture  int nAlea = random(0, 10000); //tirage du nombre entre 0 et 10000  message("Ecriture de " + String(nAlea));  monFichier.print(nAlea); //écriture dans le fichier  monFichier.write(':');  }  monFichier.close();  message("Fichier clos");    //phase de lecture  message("Ouverture du fichier en lecture");  if (!(monFichier = SD.open("alea.ard", FILE\_READ))) { //tente d'ouvrir le fichier  message("Erreur de fichier");  return; //stoppe le programme  }  message("Fichier ouvert");  char c = 0; //variable de lecture  int pos = 0; //position dans la chaîne de caractère  char tab[6] = {0}; //tableau de la chaîne de caractère  while (c != -1) {  c = monFichier.read(); //on lit un caractère  if (c == ':') { //si : on affiche  tab[pos] = '\0'; //ajout du caractère de fin de chaîne  Serial.println(tab); //affichage sur le moniteur  pos = 0; //remise à zéro de la position  }  else { // sinon  tab[pos] = c; //on ajoute le caractère à la chaîne  pos++; //on incrémente le curseur dans la chaîne  }  }  monFichier.close(); //on ferme le fichier  message("Fichier clos");  }  void loop() {  }  void message(String s) {  if (messageOK) {  Serial.println(s);  }  } |

**TP : Liste de courses** programme qui va gérer une liste de courses avec quelques fonctionnalités

* liste est saisie à l'aide du moniteur série / ajoute à la liste
* si on tape : "liste"=l’affiche / "efface"=la détruit / "compte"=affiche nb éléments
* prévoir un mode débuggage activé avec la commande "bugON" et désactivé avec "bugOFF"
* stockerez la liste sur la carte SD dans un fichier "liste.ard".

|  |
| --- |
| #include <SPI.h>  #include <SD.h>  File monFichier;  boolean messageOK = 1;  void setup() {  Serial.begin(9600); //débute la communication avec le moniteur série  Serial.println("\*\*\*\*\*\*\*\*\*\nListe de course\n\*\*\*\*\*\*\*\*\*");  bug(1); //appelle la fonction bug  initialisation(); //appelle la fonction d'initialisation  }  void loop() {  if (Serial.available()) { //teste s'il y a une saisie en attente  analyse(); //lance l'analyse de la saisie  }  }  //---------------------------\* fonctions\* ----------------------  //fonction d'initialisation de la communication avec la carte SD  void initialisation() {  message("Initialisation");  if (!SD.begin(4)) { //teste la communication avec la carte(pin 4)  message("Communication impossible");  return;  };  message("Communication ok !");  }  //fonction de modification de l'affichage des messages  void bug(boolean b) {  messageOK = b;  if (messageOK) {  Serial.println("Mode bug ON");  return;  }  Serial.println("Mode bug OFF");  }  //fonction d'analyse de la saisie  void analyse() {  String chaine = ""; // création d'un String vide  //lecture de la saisie  while (Serial.available()) { //tant que caractères en attente.  delay(10); //petit délai de lecture  char c = Serial.read(); //on lit le message  //empêche la saisie d'un #  if (c == '#') {  Serial.println("<!> ne pas saisir de #, merci");  chaine = ""; //on vide la chaine  while (Serial.available()) //on vide la saisie  Serial.read();  return; //on retourne au programme  }  if (c != 10 && c != 13) { //nettoyage de la chaine  chaine += c; //on ajoute le caractère  }  }  //test de la saisie  if (chaine == "liste") { //si liste demandée  lister();  return;  }  else if (chaine == "efface") { //si effacement demandé  effacer();  return;  }  else if (chaine == "compte") { // si comptage demandé  compter();  return;  }  else if (chaine == "bugON") { // si affichage des message demandé  bug(1);  return;  }  else if (chaine == "bugOFF") { // si non-affichage des messages demandé  bug(0);  return;  }  //si aucun code spécial  ajouter(chaine); // ajout d'un item  }  //fonction d'affichage de la liste  void lister() {  message("lister");  if (ouvrir(0)) { //appelle la fonction pour ouvrir  Serial.println("\* liste \*");  char c = 0; //variable de lecture  int pos = 2; //position dans la chaîne de caractère  char tab[102] = {0}; //tableau de la chaîne de caractère  tab[0] = '-'; //mise en page  tab[1] = ' '; //avec un tiret  while (c != -1) {  delay(10);  c = monFichier.read(); //on lit un caractère  if (c == '#') { //si # on affiche  tab[pos] = '\0'; //ajout du caractère de fin de chaîne  Serial.println(tab); //affichage sur le moniteur  pos = 2; //remise à zéro de la position  }  else { // sinon  tab[pos] = c; //on ajoute le caractère à la chaîne  pos++; //on incrémente le curseur dans la chaîne  }  }  monFichier.close(); //on ferme le fichier  message("Fichier clos");  return;  }  }  //fonction d'effacement de la liste  void effacer() {  message("Effacement de la liste");  message("Teste si fichier existant");  if (SD.exists("liste.ard")) {  message("Destruction fichier");  SD.remove("liste.ard");  Serial.println("Effacement OK");  return;  }  message("Fichier inexistant");  }  //fonction de comptage des items  void compter() {  message("compter");  if (ouvrir(0)) {  Serial.print("La liste comporte ");  int nb = 0;  char c;  while (c != -1) {  c = monFichier.read(); //on lit un caractère  if (c == '#') {  nb++;  }  }  Serial.print (nb);  Serial.print (" item");  if (nb > 1)  Serial.print("s");  Serial.println();  monFichier.close(); //on ferme le fichier  message("Fichier clos");  return;  }  }  //fonction d'ajout d'un Item  void ajouter(String ch) {  message ("Ajout de <" + ch + ">");  if (ouvrir(1)) {  ch.trim();  monFichier.print(ch); //écriture dans le fichier  monFichier.write('#');  monFichier.close();  message("Fichier clos");  Serial.println("Ajout de <" + ch + "> fait");  return;  }  }  //fonction d'ouverture de fichier  boolean ouvrir(int mode) {  message("Ouverture du fichier");  if (mode)  monFichier = SD.open("liste.ard", FILE\_WRITE);  else  monFichier = SD.open("liste.ard", FILE\_READ);  if (!(monFichier)) { //tentative d'ouverture du fichier  message("Erreur de fichier");  return 0;  }  message("Fichier ouvert");  return 1;  }  //fonction d'affichage des messages  void message(String s) {  if (messageOK) {  Serial.println("\t>"+s);  }  } |

Remarque :

* fonction pour ouvrir les fichiers pour éviter répétition
* conçu pour ajouter d’autres fonctionnalités dans la loop()

**13\_ Récupérez les informations de votre carte par réseau local**

Difficulté principale : tout ne dépend plus uniquement d’Arduino, bcp configurations de réseaux locaux.

Configuration router : souvent nécessaire pour ouvrir la communication à votre Arduino en interne.

**Connectez la carte Arduino sur le réseau**

Connexion : entre prise RJ45 Arduino vers box ou prises murales = clignotement vert = connexion établie

Adresse : la carte Arduino ou le shield possèdent une **MAC** (pour Media Access Control).

**Serveur web Arduino**

réseau interne = réseau de machines qui peuvent communiquer entre elles sans être connectées sur le web.

* chaque machine dispose 1 adresse MAC + 1 adresse IP (hypothèse : 192.168.1.x, port : 80 (http))
* navigateurs compatible avec arduino  : Safari, Firefox, Google Chrome

**Le début du programme pour créer un serveur Web Arduino**

Bibliothèques : Ethernet.h + SPI.h car Arduino communique avec le shield avec le protocole SPI

|  |
| --- |
| #include <SPI.h> //bibliothèqe pour SPI  #include <Ethernet.h> //bibliothèque pour Ethernet  byte mac[] = {0x90, 0xA2, 0xDA, 0x0F, 0xDF, 0xAB}; // tableau adresse MAC  byte ip[] = {192, 168, 1, 123}; //tableau pour l'adresse IP  void setup() {  Serial.begin (9600); //initialisation de communication série  Ethernet.begin (mac, ip); //initialisatio de la communication Ethernet  Serial.print("\nLe serveur est sur l'adresse : ");  Serial.println(Ethernet.localIP()); //on affiche l'adresse IP de la connexion  }  void loop() {  } |

Remarques :

* adresses MAC et IP sont dans des tableaux de variables de typebyte.
* Objet de type Ethernet crée avec ces 2 adresses
* on peut aussi déclarer adresse IP avec la fonction IPAddress ip(192,168,1,123);

Résultat : message sur le moniteur série qui confirme adresse indiquée

(mais tout ce qui est communication réseau est parfois capricieux en particulier le réseau local lui-même)

**Serveur** **machine (ordinateur, Arduino...) qui écoute sur un port donné le réseau en attente de clients**

Principe : si un client arrive (en se connectant à adresse serveur), il l'accueille et lui envoie série d'informations

Exemple : serveur web : envoie souvent page HTML que le client afficher sur son navigateur

Arduino comme serveur : créer un objet global serveur qui écoute sur le port le port 80 pour l'http

Ecoute : avec begin() qui attend un éventuel client

**Programme : initialise l'objet serveur et démarre l'écoute**

|  |
| --- |
| #include <SPI.h> //bibliothèqe pour SPI  #include <Ethernet.h> //bibliothèque pour Ethernet  byte mac[] = {0x90, 0xA2, 0xDA, 0x0F, 0xDF, 0xAB}; //adresse mac de votre carte  byte ip[] = {192, 168, 1, 123}; //adresse IP  EthernetServer serveur(80); // déclare l'objet serveur au port d'écoute 80  void setup() {  Serial.begin (9600); //initialisation de communication série  Ethernet.begin (mac, ip); //initialisatio de la communication Ethernet  Serial.print("\nLe serveur est sur l'adresse : ");  Serial.println(Ethernet.localIP()); //on affiche l'adresse IP de la connexion  serveur.begin(); // démarre l'écoute  }  void loop() {  if (serveur.available()){ //si client connecté  Serial.println("Client"); //on le dit...  }  } |

**Client** machine qui se connecte à un serveur pour récupérer les informations qu'il peut donner.

Un client peut envoyer des données qui seront analysées par le serveur, qui renverra à son tour un résultat.

**Navigateur web** programme qui transforme les échanges de données entre machines en résultats visibles

Langages : HTML, PHP, ASP, JavaScript

Connexion : taper adresse arduino 192.168.1.123 (sans mettre http:// devant car réseau local)

Résultat espéré : rien sur navigateur (normal rien de programmé), « Client » sur moniteur série Arduino

**Faites répondre le serveur au client** plusieurs étapes

* attendre serveur reçoive client avec la fonction available();
* créer client avec EthernetClient() qui récupère le client en attente ;
* si client connecté, on lit ses informations et on les affiche sur le moniteur ;
* une fois toutes les informations lues (donc un saut de ligne suivi d'un retour chariot), on répond au client ;
* On déconnecte le client.

|  |
| --- |
| #include <SPI.h> //bibliothèqe pour SPI  #include <Ethernet.h> //bibliothèque pour Ethernet  byte mac[] = {0x90, 0xA2, 0xDA, 0x0F, 0xDF, 0xAB}; //adresse mac de votre carte  byte ip[] = {192, 168, 1, 123}; //adresse IP  EthernetServer serveur(80); // déclare l'objet serveur au port d'écoute 80  void setup() {  Serial.begin (9600); //initialisation de la communication série  Ethernet.begin (mac, ip); //initialisation de la communication Ethernet  Serial.print("\nLe serveur est sur l'adresse : ");  Serial.println(Ethernet.localIP()); //on affiche l'adresse IP de la connexion  serveur.begin(); // démarre l'écoute  }  void loop() {  EthernetClient client = serveur.available(); //on écoute le port  if (client) { //si client connecté  Serial.println("Client en ligne\n"); //on le dit...  if (client.connected()) { // si le client est en connecté  while (client.available()) { // tant qu'il a des infos à transmettre  char c=client.read(); // on lit le caractère  Serial.write(c);// on l'écrit sur le moniteur série  delay(1); //délai de lecture  }  //réponse au client  client.println("<!DOCTYPE HTML>"); // informe navigateur type de document  client.println("<html>Bonjour OpenClassRooms !<br></html>"); //code html  client.stop(); //on déconnecte le client  Serial.println("Fin de communication avec le client");  }  }  } |

Résultats:

|  |  |
| --- | --- |
| moniteur série | navigateur |
| *Le serveur est sur l'adresse : 192.168.1.123* *Client en ligne* *GET / HTTP/1.1* *Host: 192.168.1.123* *Accept-Encoding: gzip, deflate* *Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8* *User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_10\_4) AppleWebKit/600.7.12 (KHTML, like Gecko) Version/8.0.7 Safari/600.7.12* *Accept-Language: fr-fr* *Cache-Control: max-age=0* *Connection: keep-alive*  *Fin de communication avec le client* | "Bonjour OpenClassRooms !" |

**TP : Affichez l’état des pins analogiques de l’Arduino sur une page web**

Modifier programme de serveur pour affiche une page web :

* title / h1 / hr /
* affichage de la lecture des 6 pins analogiques (construire ces lignes en dynamique)

= affichage genre : "pin A0 : " suivi de la lecture du pin ; et finir par hr

**TP : Correction :** (avecaffichage sur le moniteur série, mais il n'est pas obligatoire)

|  |
| --- |
| #include <SPI.h> //bibliothèqe pour SPI  #include <Ethernet.h> //bibliothèque pour Ethernet  byte mac[] = {0x90, 0xA2, 0xDA, 0x0F, 0xDF, 0xAB}; //adresse mac de votre carte  byte ip[] = {192, 168, 1, 123}; //adresse IP  EthernetServer serveur(80); // déclare l'objet serveur au port d'écoute 80  void setup() {  Serial.begin (9600); //initialisation de communication série  Ethernet.begin (mac, ip); //initialisation de la communication Ethernet  Serial.print("\nLe serveur est sur l'adresse : ");  Serial.println(Ethernet.localIP()); //on affiche l'adresse IP de la connexion  serveur.begin(); // démarre l'écoute  }  void loop() {  EthernetClient client = serveur.available(); //on écoute le port  if (client) { //si client connecté  Serial.println("Client en ligne\n"); //on le dit...  if (client.connected()) { // si le client est en connecté  while (client.available()) { // tant qu'il a des infos à transmettre  char c=client.read(); // on lit le caractère  Serial.write(c);// on l'écrit sur le moniteur série  delay(1); //delai de lecture  }  //réponse au client  client.println("<!DOCTYPE HTML>"); // informe le navigateur du type de document à afficher  client.println("<html>"); //début du code html  client.println("<head>"); //entête  client.println("<title>Relevés analogiques</title>"); //titre de la fenêtre  client.println("</head>");//fin d'entête  client.println("<body>"); //corps  client.println("<h1>Etat des pins analogiques</h1>"); //titre en grosse lettres  client.println("<hr>"); //ligne horizontale  for (int p=0;p<6;p++){ // boucle pour parcourir les pins  client.print("pin A"); // affichage  client.print(p); //numéro de pin  client.print(" : "); // affichage  client.print(analogRead(p)); //valeur donnée par le CAN  client.print("<br>"); //saut de ligne  }  client.println("<hr>"); //ligne horizontale  client.println("</body>"); //fin du corps  client.println("</html>"); //fin du code html  client.stop(); //on déconnecte le client  Serial.println("Fin de communication avec le client");  }  }  } |

Résultats : en actualisant (rafraîchissant) la page, vous verrez les valeurs changer.

Remarque : HTML"mange" pas mal d'espace mémoire Arduino d’où besoin de carde mini-SD

**Programme pour créer un fichier avec du texte :** permet de saisir du texte ligne par ligne => créer HTML

/\*Programme de création de fichier simple à stocker sur la carte mini-SD du shield ethernet

\* Ce programme peut servir à créer rapidement des entêtes pour des pages html

\* le mot clef "efface" supprime le fichier s'il existe \*/

|  |
| --- |
| #include <SPI.h> // bibliothèque pour com SPI  #include <SD.h> // bibliothèque pour carte SD  char nom[] = "entete.ard"; //à modifier en fonction des besoins  File monFichier; // objet file  void setup() {  Serial.begin(9600); // début communication moniteur  Serial.println("\* Start \*"); //mise en page  if (!SD.begin(4)) { // test et démarrage connexion carte SD  Serial.println("Pb avec la carte");  return;  };  //pour info  Serial.print ("Nom de fichier : ");  Serial.println(nom);  }  void loop() {  Serial.println("Saisissez votre texte :");  while (!Serial.available()); //attente de saisie  String ligne = ""; //chaîne de récupération  while (Serial.available()) { // tant que caractères à lire  char c = Serial.read(); // lecture du caractère  if (c != 10 && c != 13) { //on enlève les fins de ligne  ligne += c; // on construit la ligne  }  delay(10); //attente pour communication  }  if (ligne == "efface") { // test mot-clé  //effacement du fichier  if (SD.exists(nom)) {  SD.remove(nom);  Serial.println("Effacement ok !");  }  else {  Serial.println("Pas de fichier");  }  }  else {  //écriture du fichier  monFichier = SD.open(nom, FILE\_WRITE);  monFichier.println(ligne);  monFichier.close();  }  //affichage du contenu du fichier  Serial.println("Le ficher contient :");  monFichier = SD.open(nom, FILE\_READ);  char c = 0;  while (c != -1) {  c = monFichier.read();  Serial.print(c);  }  Serial.println();  monFichier.close();  } |

Usage : mot-clé "efface" permet de supprimer le fichier pour recommencer, besoin de saisir le nom fichier

HTML

|  |
| --- |
| <!DOCTYPE HTML>  <html>  <head>  <title>Lecture analogique</title>  </head>  <body>  <h1>Etat des pins analogiques</h1>  <hr>  <br> |

**Programme pour construire la page web à partir du fichier** programme modifié avec lecture sur carte SD

|  |
| --- |
| #include <SPI.h> //bibliothèqe pour SPI  #include <Ethernet.h> //bibliothèque pour Ethernet  #include <SD.h> // bibliothèque pour la carte SD  byte mac[] = {0x90, 0xA2, 0xDA, 0x0F, 0xDF, 0xAB}; //adresse mac de votre carte  byte ip[] = {192, 168, 1, 123}; //adresse IP  EthernetServer serveur(80); // déclare l'objet serveur au port d'écoute 80  void setup() {  Serial.begin (9600); //initialisation de communication série  if (!SD.begin(4)) { // test et démarrage connexion carte SD  Serial.println("Pb avec la carte");  return;  };  Ethernet.begin (mac, ip); //initialisation de la communication Ethernet  Serial.print("\nLe serveur est sur l'adresse : ");  Serial.println(Ethernet.localIP()); //on affiche l'adresse IP de la connexion  serveur.begin(); // démarre l'écoute  }  void loop() {  EthernetClient client = serveur.available(); //on écoute le port  if (client) { //si client connecté  Serial.println("Client en ligne\n"); //on le dit...  if (client.connected()) { // si le client est en connecté  while (client.available()) { // tant qu'il a des infos à transmettre  char c = client.read(); // on lit le caractère  Serial.write(c);// on l'écrit sur le moniteur série  delay(1); //delai de lecture  }  //réponse au client  entete(client);  for (int p = 0; p < 6; p++) { // boucle pour parcourir les pins  client.print("pin A"); // affichage  client.print(p); //numéro de pin  client.print(" : "); // affichage  client.print(analogRead(p)); //valeur donnée par le CAN  client.print("<br>"); //saut de ligne  }  client.println("<br><hr></body></html>"); //ligne hr et fermeture balises  client.stop(); //on déconnecte le client  Serial.println("Fin de communication avec le client");  }  }  }  //fonction d'affichage de l'entête HTML  void entete(EthernetClient cl) {  File monFichier = SD.open("entete.ard", FILE\_READ);  char c = 0;  while (c != -1) {  c = monFichier.read();  if (c > 31 || c == 10) { //permet d'éviter des charactères non affichables  cl.write(c);  }  }  monFichier.close();  } |

**Remarque :**

* lecture du fichier dans une fonction
* pour envoyer l'info à l'objet client, paramètre d’attente d'un objet client (typeEthernetClient)

**Rafraîchissez une page web automatiquement**

Init : avant de construire la page, serveur doit envoyer au client avant <!DOCTYPE HTML> :

🡪 type de document, taux rafraîchissement éventuel, codage des caractères, type de maintien de la connexion

|  |
| --- |
| HTTP/1.1 200 OK  Content-Type: text/html; charset=ascii  Connection: close  Refresh: 5 |

Remarque : placé directement dans ma fonction d'en-tête, aussi possible dans fichier stocké sur la carte SD.

|  |
| --- |
| #include <SPI.h> //bibliothèqe pour SPI  #include <Ethernet.h> //bibliothèque pour Ethernet  #include <SD.h> // bibliothèque pour la carte SD  byte mac[] = {0x90, 0xA2, 0xDA, 0x0F, 0xDF, 0xAB}; //adresse mac de votre carte  byte ip[] = {192, 168, 1, 123}; //adresse IP  EthernetServer serveur(80); // déclare l'objet serveur au port d'écoute 80  void setup() {  Serial.begin (9600); //initialisation de communication série  if (!SD.begin(4)) { // test et démarrage connexion carte SD  Serial.println("Pb avec la carte");  return;  };  Ethernet.begin (mac, ip); //initialisation de la communication Ethernet  Serial.print("\nLe serveur est sur l'adresse : ");  Serial.println(Ethernet.localIP()); //on affiche l'adresse IP de la connexion  serveur.begin(); // démarre l'écoute  }  void loop() {  EthernetClient client = serveur.available(); //on écoute le port  if (client) { //si client connecté  Serial.println("Client en ligne\n"); //on le dit...  if (client.connected()) { // si le client est en connecté  while (client.available()) { // tant qu'il a des infos à transmettre  char c = client.read(); // on lit le caractère  Serial.write(c);// on l'écrit sur le moniteur série  delay(1); //délai de lecture  }  //réponse au client  entete(client);  for (int p = 0; p < 6; p++) { // boucle pour parcourir les pins  client.print("pin A"); // affichage  client.print(p); //numéro de pin  client.print(" : "); // affichage  client.print(analogRead(p)); //valeur donnée par le CAN  client.print("<br>"); //saut de ligne  }  client.println("<br><hr></body></html>"); //ligne horizontale et fermeture des balises  client.stop(); //on déconnecte le client  Serial.println("Fin de communication avec le client");  }  }  }  //fonction d'affichage de l'entête HTML  void entete(EthernetClient cl) {  cl.println("HTTP/1.1 200 OK"); // type du HTML  cl.println("Content-Type: text/html; charset=ascii ");  //type de fichier et encodage des caractères  cl.println("Connection: close");  // fermeture connexion quand toute réponse sera envoyée  cl.println("Refresh: 5");  // rafraîchissement automatique de la page toutes les 5 secondes  cl.println(); //saut de ligne obligatoire avant la suite  File monFichier = SD.open("entete.ard", FILE\_READ);  char c = 0;  while (c != -1) {  c = monFichier.read();  if (c > 31 || c == 10) { //permet d'éviter des caractères non affichables  cl.write(c);  Serial.write(c);  }  }  monFichier.close();  } |

**14\_ PILOTEZ VOTRE CARTE ARDUINO SUR LE RÉSEAU LOCAL**

**Méthode GET**

Principe : ajoute données l'URL en utilisant trois symboles : exemple : page.html**?**age**=**20**&**taille**=**180

|  |  |
| --- | --- |
|  | 2 liens : rafraichissement + envoi données en GET |
| <!DOCTYPE HTML>  <html>  <head>  <title>Essai</title>  </head>  <body>  <h1>Essai</h1>  Voici une page simple<br><br>  <a href=page.html?age=20&taille=180 target=\_self>Envoi</a>  <br><br>  <a href=page.html target=\_self>Encore</a>  </body>  </html> | #include <SPI.h> //bibliothèqe pour SPI  #include <Ethernet.h> //bibliothèque pour Ethernet  byte mac[] = {0x90, 0xA2, 0xDA, 0x0F, 0xDF, 0xAB}; //adresse mac de votre carte  byte ip[] = {192, 168, 1, 123}; //adresse IP  EthernetServer serveur(80); // déclare l'objet serveur au port d'écoute 80  void setup() {  Serial.begin (9600); //initialisation de communication série  Ethernet.begin (mac, ip); //initialisation de la communication Ethernet  Serial.print("\nLe serveur est sur l'adresse : ");  Serial.println(Ethernet.localIP()); //on affiche l'adresse IP de la connexion  serveur.begin(); // démarre l'écoute  }  void loop() {  EthernetClient client = serveur.available(); //on écoute le port  if (client) { //si client connecté  Serial.println("Client en ligne\n"); //on le dit...  if (client.connected()) { // si le client est en connecté  while (client.available()) { // tant qu'il a des infos à transmettre  char c = client.read(); // on lit le caractère  Serial.write(c);// on l'écrit sur le moniteur série  delay(1); //délai de lecture  }  //réponse au client  entete(client);  client.println("<a href=?valeur=100>Envoi</a><br>");  client.println("<a href=?>Refresh</a><br>");  client.println("<br><hr></body></html>"); //ligne horizontale et fermeture des balises  client.stop(); //on déconnecte le client  Serial.println("Fin de communication avec le client");  }  }  }  //fonction d'affichage de l'entête HTML  void entete(EthernetClient cl) {  cl.println("<!DOCTYPE HTML>");  cl.println("<html>");  cl.println("<head><title>Esssai</title></head>");  cl.println("<body><h1>Essai</h1><hr><br>");  } |

Résultat : si on clique sur ‘envoi’, affichage sur le moniteur série :

|  |
| --- |
| GET /?valeur=100 HTTP/1.1 Host: 192.168.1.123 Connection: keep-alive Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_10\_4) AppleWebKit/600.7.12 (KHTML, like Gecko) Version/8.0.7 Safari/600.7.12 Accept-Language: fr-fr Referer: http://192.168.1.123/ Accept-Encoding: gzip, deflate  Ce qui nous intéresse, c'est la première ligne :  GET /?valeur=100 HTTP/1.1 |

Principe lecture :

1\_ réception des données jusqu'à ?

2\_ créer une chaîne pour le nom paramètre avec tous les caractères jusqu'à =

3\_ créer une chaîne pour la valeur avec tous les caractères rencontrer un &  ou un espace

4\_ si c'est un &  on recommence, si c'est un espace, on a fini, le reste n'est pas utile.

|  |
| --- |
| #include <SPI.h> //bibliothèqe pour SPI  #include <Ethernet.h> //bibliothèque pour Ethernet  byte mac[] = {0x90, 0xA2, 0xDA, 0x0F, 0xDF, 0xAB}; //adresse mac de votre carte  byte ip[] = {192, 168, 1, 123}; //adresse IP  EthernetServer serveur(80); // déclare l'objet serveur au port d'écoute 80  void setup() {  Serial.begin (9600); //initialisation de communication série  Ethernet.begin (mac, ip); //initialisation de la communication Ethernet  Serial.print("\nLe serveur est sur l'adresse : ");  Serial.println(Ethernet.localIP()); //on affiche l'adresse IP de la connexion  serveur.begin(); // démarre l'écoute  }  void loop() {  EthernetClient client = serveur.available(); //on écoute le port  if (client) { //si client connecté  Serial.println("Client en ligne"); //on le dit...  if (client.connected()) { // si le client est en connecté  String affichage = GET(client); //appel de la fonction de décodage  //réponse au client  entete(client);  client.println(affichage);  client.println("<a href=?valeur=100&autreValeur=10&anniv=oui target=\_self>Envoi</a><br>");  client.println("<a href=? target=\_self>Refresh</a><br>");  client.println("<br><hr></body></html>"); //ligne horizontale et fermeture des balises  client.stop(); //on déconnecte le client  Serial.println("Fin de communication avec le client\n");  }  }  }  //fonction d'affichage de l'entête HTML  void entete(EthernetClient cl) {  cl.println("<!DOCTYPE HTML>");  cl.println("<html>");  cl.println("<head><title>Esssai</title></head>");  cl.println("<body><h1>Essai</h1><hr><br>");  }  //fonctin décodage GET  String GET(EthernetClient cl) {  int lecture = 0; // variable pour les étapes de décodage  String resultat = "<br>"; //initialisation de la chaine de réponse  String donnee = ""; //chaine pour stocker la lecture des données  while (cl.available()) { // tant qu'il a des infos à transmettre  char c = cl.read(); // on lit le caractère  if (lecture == 0 && c == '?') { //début de lecture des données donc d'un nom  lecture = 1;  donnee = "";  }  else if (lecture == 1 && c == '=') { //début de lecture d'une valeur  lecture = 2;  resultat += donnee + " : "; //on construit la chaîne de réponse  donnee = "";  }  else if (lecture == 2 && c == '&') { //nouveau nom  lecture = 1;  resultat += donnee + "<br>"; //on construit la chaîne de réponse  donnee = "";  }  else if ((lecture == 2 || lecture == 1) && c == ' ') { //fin de lecture  lecture = 3;  resultat += donnee + "<br><br>"; // on finit la chaîne réponse.  }  else if (lecture == 1 || lecture == 2) {//récupération des données de nom ou de valeur  donnee += c;  }  delay(1); //delai de lecture  }  return resultat; // retour le la chaîne de réponse  } |

**Allumez et éteignez des LED avec Arduino via le réseau interne**

Objectif :

* connecter 5 LED sur pins 2, 3, 5, 6 et 7 (4 pour la carte SD)
* créer une page web qui proposera 5 liens pour piloter une LED (allumer/ éteindre/clignoter (0,5 Hz)

Conseils :

* envoyer seulement le changement d'état d'une LED (le numéro de LED à changer)
* ce qui est lu est un caractère donc ‘1’ n’est pas le nombre 1 mais vaut 49. Le convertir en numérique.
* gérer le temps et permettre de faire d'autres tâches pendant attente des données (comme le clignotement)
* Ne pas oublier les résistances dans les montages

Programme :

|  |
| --- |
| #include <SPI.h> //bibliothèqe pour SPI  #include <Ethernet.h> //bibliothèque pour Ethernet  byte mac[] = {0x90, 0xA2, 0xDA, 0x0F, 0xDF, 0xAB}; //adresse mac de votre carte  byte ip[] = {192, 168, 1, 123}; //adresse IP  unsigned long tpsDep; //temps départ pour la gestion des LED  int pinLed[5] = {2, 3, 5, 6, 7}; //tableau des pins de LED  boolean etatLed[5] = {0, 0, 0, 0, 0}; //tableau des états des LED  int modeLed[5] = {0, 0, 0, 0, 0}; // tableau des modes des LED  EthernetServer serveur(80); // déclare l'objet serveur au port d'écoute 80  void setup() {  for (int l = 0; l < 5; l++) {  pinMode(pinLed[l], OUTPUT);  }  Serial.begin (9600); //initialisation de communication série  Ethernet.begin (mac, ip); //initialisation de la communication Ethernet  Serial.print("\*\n-> Le serveur est sur l'adresse : ");  Serial.println(Ethernet.localIP()); //on affiche l'adresse IP de la connexion  serveur.begin(); // démarre l'écoute  }  void loop() {  gestionClient(); // fonction qui gère toute la communication avec le client  gestionLed(); // fonction qui gère l'allumage des LED  }  //----------------------Fonctions----------------------  //fonction qui gère la communication avec le client  void gestionClient() {  EthernetClient client = serveur.available(); //on écoute le port  if (client) { //si client existe  Serial.println("Client en ligne"); //on le dit...  if (client.connected()) { // si le client est connecté  GET(client); //appel de la fonction de décodage  //réponse au client  entete(client); // fonction pour l'entête de la page HTML  corps(client); // fonction pour le corps  piedPage(client); // fonction pour le pied de page  Serial.println("Fin de communication avec le client\n");  client.stop(); //on déconnecte le client  }  }  }  //fonction de fabrication de l'entête HTML  void entete(EthernetClient cl) {  //infos pour le navigateur  cl.println("HTTP/1.1 200 OK"); // type du HTML  cl.println("Content-Type: text/html; charset=ascii");  //type de fichier et encodage des caractères  cl.println("Connection: close");  // fermeture de la connexion quand toute la réponse sera envoyée  cl.println();  //balises d'entête  cl.println("<!DOCTYPE HTML>");  cl.println("<html>");  cl.println("<head><title>Web-Commande de LED</title></head>");  cl.println("<body><h1>Web-Commande de LED</h1><hr><br>");  }  //fonction de fabrication du corps de page  void corps(EthernetClient cl) {  cl.println("<br>"); // saut de ligne  //boucle pour construire chaque ligne en fonction des LED  for (int l = 0; l < 5; l++) {  cl.print("<br>LED ");  cl.print(l);  cl.print(" ");  Serial.println(l);  switch (modeLed[l]) {  case 0:  cl.print("OFF ");  break;  case 1:  cl.print("ON ");  break;  case 2:  cl.print("CLI ");  break;  }  cl.print(" <a href=?");  //création du lien inutile de répéter l'adresse du site  cl.print(l);  cl.println(" target=\_self >Change</a><br>");  }  }  //fonction de fabrication du pied de page  void piedPage(EthernetClient cl) {  //balises de pied de page  cl.println("<br><hr>");  cl.println("</body>");  cl.println("</html>");  }  //fonctin décodage GET  void GET(EthernetClient cl) {  boolean lu = 0; //variable pour indiquer l'état de lecture  while (cl.available()) { // tant qu'il a des infos à transmettre  char c = cl.read(); // on lit le caractère  delay(1); //delai de lecture  if (c == '?' && lu == 0) { //si "?" repéré  c = cl.read(); //on lit le caractère suivant qui contient la donnée  int led = int(c) - 48; //on la transforme en nombre  modeLed[led] = (modeLed[led] + 1) % 3; //on change l'état de la LED  delay(10);  lu = 1; // on dit qu'on a lu l'info  }  }  }  //fonction d'allumage des LED par rapport au tableau de mode.  void gestionLed() {  unsigned long tpsAct = millis(); // récupération du temps actuel  if (tpsAct - tpsDep > 250) { //si délai dépassé  for (int l = 0; l < 5; l++) { // chaque LED  if (modeLed[l] == 0) { // si mode éteint  etatLed[l] = 0; // on éteind  }  else if (modeLed[l] == 1) {// si mode allumé  etatLed[l] = 1; //on allume  }  else if (modeLed[l] == 2) { //si mode clignotant  etatLed[l] = !etatLed[l]; //on inverse l'état  }  digitalWrite(pinLed[l], etatLed[l]); //on met le pin à jour  tpsDep = tpsAct; //on réinitialise le temps  }  }  } |

Remarques :

* fonctions qui construisent l'en-tête, le corps et le pied de page HTML
* pour l'envoi, ne crée pas ?variable=valeur mais juste le lien en mettant le numéro de la LED après le ?
* la fonction d'analyse du GET met à jour les tableaux qui sont ensuite exploités par fct d'allumage des LED

**15\_ PILOTEZ VOTRE ARDUINO SUR LE RÉSEAU MONDIAL**

Difficulté : paramétrage de votre box (votre accès Internet)

Rappel : adresse IP dans le programme Arduino = nombre séparés de virgules

**Adresse IP publique :** [mon-ip.com](http://www.mon-ip.com) / [votreip.free.fr](http://votreip.free.fr)

* **Paramétrez votre box pour l'Arduino**

Principe : paramétrer la box pour qu’une connexion l'adresse publique et un port donné dirige vers Arduino (srv)

0\_ se connecter sur la box

1\_ associer l'adresse de votre Arduino (192.168.1.123) à un nom

2\_ associer cette adresse avec l'adresse MAC (matérielle) de votre carte Arduino = la rendre statique

3\_ créer un relai entre l'adresse publique de votre box (et un port précis) et votre Arduino.

**Étape 1 : Associez l'adresse locale à un nom photoe\_1**

**Menu DNS** (Domain Name System) = = gestion ipV4

= propose d'associer une adresse IP à un nom par exemple « arduino »

NB : si adresse locale n'apparaît pas, la taper directement l'adresse en question ainsi que le nom.

**Étape 2 : Fixez l'adresse locale et l'associer l'adresse MAC photoe\_2**

**Menu DHCP** (Dynamic Host Configuration Protocol) pour associant l’adresse à machine (Arduino)

= associe donc une IP locale à une adresse MAC.

**Étape 3 : Reliez l'adresse interne avec l'adresse publique et un numéro de port photoe\_3**

**Menu NAT** (Network Address Translation) = relier adresse publique + n° port à adresse locale statique Arduino.

= nom de l'adresse liée (=Arduino) / protocole (=TCP) / port (entre 1024 et 65535) (=port précis 80)

Programmes : soit ceux chapitre précédent soit le suivant **(mettre son adresse MAC)**

|  |
| --- |
| #include <SPI.h> //bibliothèqe pour SPI  #include <Ethernet.h> //bibliothèque pour Ethernet  byte mac[] = {0x90, 0xA2, 0xDA, 0x0F, 0xDF, 0xAB}; //adresse mac de votre carte  byte ip[] = {192, 168, 1, 123}; //adresse IP  EthernetServer serveur(80); // déclare l'objet serveur au port d'écoute 80  void setup() {  Serial.begin (9600); //initialisation de communication série  Ethernet.begin (mac, ip); //initialisation de la communication Ethernet  Serial.print("\nLe serveur est sur l'adresse : ");  Serial.println(Ethernet.localIP()); //on affiche l'adresse IP de la connexion  serveur.begin(); // démarre l'écoute  }  void loop() {  EthernetClient client = serveur.available(); //on écoute le port  if (client) { //si client connecté  Serial.println("Client en ligne\n"); //on le dit...  if (client.connected()) { // si le client est en connecté  //réponse au client  entete(client);  client.println("Le monde nous est ouvert !<br>");  client.println("Vive l'Arduino !");  client.println("<br><hr></body></html>"); //hr et fermeture des balises  client.stop(); //on déconnecte le client  Serial.println("Fin de communication avec le client");  }  }  }  //fonction d'affichage de l'entête HTML  void entete(EthernetClient cl) {  cl.println("<!DOCTYPE HTML>");  cl.println("<html>");  cl.println("<head><title>Esssai</title></head>");  cl.println("<body><h1>Essai</h1><hr><br>");  } |

Usage : taper directement dans la barre du navigateur web : **adresseIP:port**

Résultat : affiche la page … sur le web mondial !

**Les tableaux HTML** balises <table>, <tr>, <td>.

|  |
| --- |
| <!DOCTYPE HTML>  <html>  <head>  <title>Tableau</title>  </head>  <body>  <table rules=all border=1px>  <tr>  <td>cellule 0.0</td>  <td>cellule 1.1</td>  </tr>  <tr>  <td>cellule 1.0</td>  <td>cellule 1.1</td>  </tr>  </table>  </body>  </html> |

**TP : la table de pythagore**

Rappel : table de Pythagore = tableau de résultats de produits des nombres de 0 à 9 **photoe\_4**

Asctuce : code pour créer automatiquement un tableau HTML à l'aide de boucles

Exemple : tableau 5 lignes x 4 colonnes

|  |
| --- |
| client.println("<table rules=all border=1px>"); //ouvre la balise tableau avec deux paramètres  for (int l=0;l<5,;l++){  client.println("<tr>"); //ouvre une balise ligne  for (int c=0;c<5;c++){  client.print("<td>");//ouvre une balise cellule  client.print("un truc");//écrit "un truc" dans la cellule  client.println("</td>");//ferme la balise cellule  }  client.println("</tr>");//ferme la balise ligne  }  client.println("</table>");//ferme la balise tableau |

Table de Pythagore :

* titre fenêtre "Table de Pythagore" / titre page "Table de Pythagore" / 1ere cellule haut-gauche "X"
* 1ère ligne nombres de 0 à 9 / 1ère colonne nombres de 0 à 9 ;
* chaque cellule contient résultat du produit du produit

**TP : Correction**

Rappel : IDE de l'Arduino ne vous indiquera pas de faute sur le code HTML = que des codes affichés

|  |
| --- |
| #include <SPI.h> //bibliothèqe pour SPI  #include <Ethernet.h> //bibliothèque pour Ethernet  byte mac[] = {0x90, 0xA2, 0xDA, 0x0F, 0xDF, 0xAB}; //adresse mac de votre carte  byte ip[] = {192, 168, 1, 123}; //adresse IP  EthernetServer serveur(80); // déclare l'objet serveur au port d'écoute 80  void setup() {  Serial.begin (9600); //initialisation de communication série  Ethernet.begin (mac, ip); //initialisation de la communication Ethernet  Serial.print("\nLe serveur est sur l'adresse : ");  Serial.println(Ethernet.localIP()); //on affiche l'adresse IP de la connexion  serveur.begin(); // démarre l'écoute  }  void loop() {  EthernetClient client = serveur.available(); //on écoute le port  if (client) { //si client connecté  Serial.println("Client en ligne\n"); //on le dit...  if (client.connected()) { // si le client est en connecté  //réponse au client  entete(client); // appel de la fonction pour créer l'entête de la page  corps(client); // appel de la fonction pour créer le corps de la page  client.println("<br><hr></body></html>"); //ligne horizontale et fermeture des balises  client.stop(); //on déconnecte le client  Serial.println("Fin de communication avec le client");  }  }  }  //------------------------------- Fonctions ------------------  //fonction d'affichage de l'entête HTML  void entete(EthernetClient cl) {  //infos pour le navigateur  cl.println("HTTP/1.1 200 OK"); // type du HTML  cl.println("Content-Type: text/html; charset=ascii"); //type de fichier et encodage des caractères  cl.println("Connection: close"); // fermeture de la connexion quand toute la réponse sera envoyée  cl.println();  //balises d'entête  cl.println("<!DOCTYPE HTML>");  cl.println("<html>");  cl.println("<head><title>Table de Pythagore</title></head>");  cl.println("<body><h1>Table de Pythagore</h1><hr><br>");  }  //fonction d'affichage du corps de la page HTML  void corps(EthernetClient cl) {  cl.println("<table rules=all border=1px>"); //ouverture de la balise tableau  for (int l = -1; l < 10; l++) { // boucle pour les lignes  cl.println("<tr>"); //ouverture d'une balise ligne  for (int c = -1; c < 10; c++) { // boucle pour les colonnes  cl.println("<td>"); // ouverture d'une balise cellule  if (c == -1 && l == -1) {// si case en haut à gauche  cl.print("X"); //on affiche un X dans la cellule  }  else if (c > -1 && l == -1) {// si ligne d'entête  cl.print(c); // on affiche le nombre d'entête de colonne dans la cellule  }  else if (c == -1 && l > -1) { // si colonne d'entête  cl.print(l); // on affiche le nombre d'entête de ligne dans la cellule  }  else { // sinon  cl.print(l \* c); // on affiche le produit ligne x colonne  }  cl.println("</td>"); // on ferme la balise de cellule  delay(1); //on attend un peu  }  cl.println("</tr>");// on ferme la balise de ligne  }  cl.println("</table>"); // on ferme la balise de tableau  } |

Remarque : ce n’est qu’un page

**TP : Modifier la table de Pythagore grâce à un formulaire HTML** rendre les pages interactives avec formulaires

Objectif : formulaire pour préciser les paramètres de table de multiplication et générer

Rappel : vu déjà comment envoyer données à l'Arduino depuis navigateur grâce à un lien hypertexte avec GET

Principe : utiliser des formulaires pour envoyer des chaines de caractères

|  |
| --- |
| <!DOCTYPE HTML>  <html>  <head>  <title>Essai de formulaire</title>  </head>  <body>  Un exemple de formulaire<br>  <form method='post' action=''>  <label for='mini'>Nombre minimum : </label>  <input type='text' name='mini' id='mini'/> <br>  <label for='maxi'>Nombre maximum : </label>  <input type='text' name='maxi' id='maxi'/> <br>  <input type='submit' value='Envoi' />  </form>  </body>  </html> |

Commentaires :

**balise <form>**

* '**method**' = méthode choisie pour envoi onnées
  + GET = données liées à votre URL (mais limitées à 255 caractères)
  + POST = données ne sont pas visibles dans l'URL mais peut envoyer de grandes quantités de données
* '**action**' : adresse à laquelle vous envoyez le formulaire = **adresseIP:port**

NB : en laissant action='' on obtient souvent un accès correct en réseau interne comme depuis l'extérieur

**Décodez des informations reçues par la méthode POST programme q**ui construit formulaire \_ tester en local

|  |
| --- |
| #include <SPI.h> //bibliothèqe pour SPI  #include <Ethernet.h> //bibliothèque pour Ethernet  byte mac[] = {0x90, 0xA2, 0xDA, 0x0F, 0xDF, 0xAB}; //adresse mac de votre carte  byte ip[] = {192, 168, 1, 123}; //adresse IP  EthernetServer serveur(80); // déclare l'objet serveur au port d'écoute 80  void setup() {  Serial.begin (9600); //initialisation de communication série  Ethernet.begin (mac, ip); //initialisation de la communication Ethernet  Serial.print("\nLe serveur est sur l'adresse : ");  Serial.println(Ethernet.localIP()); //on affiche l'adresse IP de la connexion  serveur.begin(); // démarre l'écoute  }  void loop() {  EthernetClient client = serveur.available(); //on écoute le port  if (client) { //si client connecté  Serial.println("Client en ligne\n"); //on le dit...  if (client.connected()) { // si le client est en connecté  while (client.available()) {  char c = client.read();  if (c != 13)  Serial.write(c);  }  Serial.println();  //réponse au client  entete(client); // appel de la fonction pour créer l'entête de la page  formulaire(client); // appel de la fonction pour créer le formulaire de la page  client.println("<br><hr></body></html>"); //ligne horizontale et fermeture des balises  client.stop(); //on déconnecte le client  Serial.println("Fin de communication avec le client");  }  }  }  //------------------------------- Fonctions ------------------  //fonction d'affichage de l'entête HTML  void entete(EthernetClient cl) {  //infos pour le navigateur  cl.println("HTTP/1.1 200 OK"); // type du HTML  cl.println("Content-Type: text/html; charset=ascii"); //type de fichier et encodage des caractères  cl.println("Connection: close"); // fermeture de la connexion quand toute la réponse sera envoyée  cl.println();  //balises d'entête  cl.println("<!DOCTYPE HTML>");  cl.println("<html>");  cl.println("<head><title>Table de Pythagore</title></head>");  cl.println("<body><h1>Table de Pythagore</h1><hr><br>");  }  //fonction d'affichage du formulaire HTML  void formulaire(EthernetClient cl) {  cl.println("<form method='post' action=''>");  cl.println("<label for='mini'>Nombre minimum : </label>");  cl.println("<input type='text' name='mini' id='mini' /> <br>");  cl.println("<label for='maxi'>Nombre maximum : </label>");  cl.println("<input type='text' name='maxi' id='maxi' /> <br>");  cl.println("<input type='submit' value='Envoi' />");  cl.println("</form>");  } |

Résultat : sur le moniteur série, affichage selon machine

|  |
| --- |
| POST / HTTP/1.1 Host: 192.168.1.123 Content-Type: application/x-www-form-urlencoded Origin: http://192.168.1.123 Content-Length: 13 Connection: keep-alive Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_10\_4) AppleWebKit/600.7.12 (KHTML, like Gecko) Version/8.0.7 Safari/600.7.12 Referer: http://192.168.1.123/ Accept-Language: fr-fr Accept-Encoding: gzip, deflate  mini=0&maxi=9 |

Asctuce : pour repérer les données, utilise balise input avec attribut type='hidden' et value=''

cl.println("<input type='hidden' name='x' id='x' value='' />");

= permet de repérer le premier & = indique le début des données

Procédure de codage : (limiter les noms de variables à 4 caractères

* Lire données jusqu'à & / lit 5 caractères suivants (mini=) / lit suite jusqu'au prochain &
* Tester s'il s'agit bien d'un nombre et transformer la suite de caractère en nombre

(on stocke les caractères dans une chaîne de type String)

* lire 5 caractères suivants (maxi=) / lire caractères jusqu'à là fin des données
* les 2 valeurs récupérées, vérifier certaines limites (<100 ne pas dépasser valeur maximale pour int)
* si tout est ok, mettre à jour variables / afficher table , sinon envoie un message d'erreur (ex : saisie)

**TP : correction de la table de Pythagore avec formulaire**

|  |
| --- |
| #include <SPI.h> //bibliothèqe pour SPI  #include <Ethernet.h> //bibliothèque pour Ethernet  byte mac[] = {0x90, 0xA2, 0xDA, 0x0F, 0xDF, 0xAB}; //adresse mac de votre carte  byte ip[] = {192, 168, 1, 123}; //adresse IP  int mini = 0; //variable pour la valeur inférieure  int maxi = 9; //varialbe pour la valeur supérieure  EthernetServer serveur(80); // déclare l'objet serveur au port d'écoute 80  void setup() {  Serial.begin (9600); //initialisation de communication série  Ethernet.begin (mac, ip); //initialisation de la communication Ethernet  Serial.print("\nLe serveur est sur l'adresse : ");  Serial.println(Ethernet.localIP()); //on affiche l'adresse IP de la connexion  serveur.begin(); // démarre l'écoute  }  void loop() {  EthernetClient client = serveur.available(); //on écoute le port  if (client) { //si client connecté  Serial.println("Client en ligne\n"); //on le dit...  if (client.connected()) { // si le client est en connecté  int r = decodage(client); //on appelle la fonction qui décode et on récupère le retour  //réponse au client  entete(client); // appel de la fonction pour créer l'entête de la page  if (!r) { //si erreur de décodage  client.println("<h1>Erreur de saisie</h1>"); //on l'indique  mini = 0; //on fixe le minimum  maxi = 9; //et le maximum  }  formulaire(client);// appel de la fonction pour créer le corps de la page  corps(client); //on construit le tableau  client.println("<br><hr></body></html>"); //ligne horizontale et fermeture des balises  client.stop(); //on déconnecte le client  Serial.println("Fin de communication avec le client");  }  }  }  //------------------------------- Fonctions ------------------  //fonction d'affichage de l'entête HTML  void entete(EthernetClient cl) {  //infos pour le navigateur  cl.println("HTTP/1.1 200 OK"); // type du HTML  cl.println("Content-Type: text/html; charset=ascii"); //type de fichier et encodage des caractères  cl.println("Connection: close"); // fermeture de la connexion quand toute la réponse sera envoyée  cl.println();  //balises d'entête  cl.println("<!DOCTYPE HTML>");  cl.println("<html>");  cl.println("<head><title>Table de Pythagore</title></head>");  cl.println("<body><h1>Table de Pythagore</h1><hr><br>");  }  //fonction d'affichage du formulaire HTML  void formulaire(EthernetClient cl) {  cl.println("<form method='post' action=''>");  cl.println("<input type='hidden' name='x' id='x' value='' />");  cl.println("<label for='mini'>Nombre minimum : </label>");  cl.println("<input type='text' name='mini' id='mini' /> <br>");  cl.println("<label for='maxi'>Nombre maximum : </label>");  cl.println("<input type='text' name='maxi' id='maxi' /> <br>");  cl.println("<input type='submit' value='Envoi' />");  cl.println("</form>");  }  //fonction de décodage des infos client  int decodage(EthernetClient cl) {  String valeur = ""; //chaîne pour la récupération des valeurs  char c = lecture(cl); // on lit la première donnée  while (c != '&') { //tant qu'on ne lit pas un &  c = lecture(cl); //on continue de lire  if (c == -1) return 0; //renvoie erreur si fin de fichier  }  //& trouvé  for (int t = 0; t < 5; t++) { //on lit les 5 caractères suivants  c = lecture(cl);  }  while (c != '&') { // tant qu'on n'est pas au bout de la saisie  c = lecture(cl);  if (c > 47 && c < 58) { //si il s'agit d'un nombre (voic tableau ascii)  valeur += c; //on l'ajoute à la chaîne valeur  }  //else return 0; //sinon on retourne une erreur  }  if (valeur == "") { // si rien n'est saisi  mini = 0; //on fixe le minimum à 0  }  else {  mini = valeur.toInt(); // on initialise la valeur minimum avec la saisie  }  if (mini < 0 || mini > 99) { //test si nombre saisi est correct  return 0;  }  valeur = ""; //on réinitialise la valeur  for (int t = 0; t < 5; t++) { //on lit les 5 caractères suivants  c = lecture(cl);  }  while (cl.available()) { //tant qu'il reste des caractères à lire  c = lecture(cl); //on les lit  if (c != 0) { //si le caractère existe  if (c > 47 && c < 58) { // si c'est un nombre  valeur += c; //on l'ajoute à la chaîne  }  else return 0; //sinon on retourne une erreur  }  }  if (valeur == "") { // si rien n'est saisi  maxi = 9; //on fixe le maximum à 9  }  else {  maxi = valeur.toInt(); // on initialise la valeur minimum avec la saisie  }  if (maxi <= mini || maxi > 99) { //test si nombre saisi est correct  return 0;  }  return 1; // on retourne que tout va bien  }  //fonction d'affichage du corps de la page HTML  void corps(EthernetClient cl) {  cl.println("<table rules=all border=1px>"); //ouverture de la balise tableau  for (int l = mini - 1; l < maxi + 1; l++) { // boucle pour les lignes  cl.println("<tr>"); //ouverture d'une balise ligne  for (int c = mini - 1; c < maxi + 1; c++) { // boucle pour les colonnes  cl.println("<td>"); // ouverture d'une balise cellule  if (c == mini - 1 && l == mini - 1) { // si case en haut à gauche  cl.print("X"); //on affiche un X dans la cellule  }  else if (c > mini - 1 && l == mini - 1) { // si ligne d'entête  cl.print(c); // on affiche le nombre d'entête de colonne dans la cellule  }  else if (c == mini - 1 && l > mini - 1) { // si colonne d'entête  cl.print(l); // on affiche le nombre d'entête de ligne dans la cellule  }  else { // sinon  cl.print(l \* c); // on affiche le produit ligne x colonne  }  cl.println("</td>"); // on ferme la balise de cellule  delay(1); //on attend un peu  }  cl.println("</tr>");// on ferme la balise de ligne  }  cl.println("</table>"); // on ferme la balise de tableau  }  //fonction de lecture des données client avec attente.  char lecture(EthernetClient cli) {  char ch = cli.read();  delay(1);  /\*  \* Si vous voulez afficher la lecture, décommenter ce qui suit  \* if (c!=13) // si pas retour chariot  \* Serial.write(c); //on écrit le caractère  \* else //sinon  \* Serial.write('|'); //on écrit un symbole pour le retour chariot  \*/  return ch; //on renvoie le caractère lu  } |

**Pour aller plus loin**

* Etudier bibliothèque Ethernet liée au shield Arduino qui permet bien d'autres choses.
* Il existe aussi bibliothèques liées au HTML : HttpClient ou Webduino.