Nom :…… Prénom :……

TP SIN

Programmation

Support : Carte Arduino et Shield Ethernet

Support : Carte arduino Méga et Shield Ethernet

Pré requis (l'élève doit savoir):

- Savoir utiliser un ordinateur

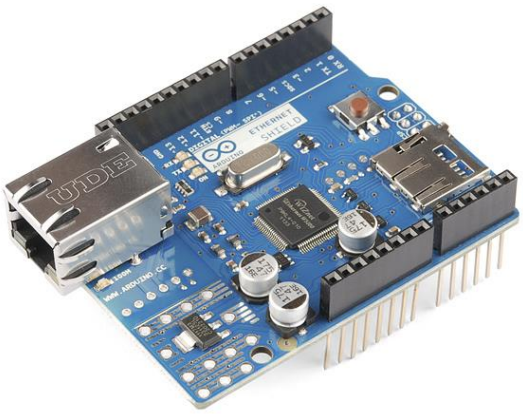
Programme

Objectif terminal :

L'élève doit être capable de commander un servo moteur depuis une page html

Matériels :

- Logiciel Arduino
- Carte Arduino méga
- Carte Ethernet Shield Arduino
- 3 boutons
- Servo moteur
- Carte shield Ethernet



Nom :…… Prénom :……

1. Travail demandé

- Cahier des charges :

  On veut commander l'allumage de Leds par une page html.

Remarque :

La balise <form> déclare une zone de formulaire dans laquelle les utilisateurs peuvent saisir des informations qui pourront par la suite être récupérées et traitées côté serveur. Il est possible d'utiliser plusieurs formulaires dans une même page sous condition de ne pas les imbriquer. Finalement, pour que le formulaire puisse être envoyé il faut qu'il soit soumis,
soit à l'aide de la balise input : <input type="submit" .../>
soit à l'aide du javascript en appliquant la fonction "submit()" au formulaire (en se basant sur son identifiant)

D'autre part, pour utiliser cette notion de formulaire, il est important de donner les valeurs adéquates aux principaux attributs de la balise <form> :

- name : nom du formulaire utilisé pour faire référence à celui-ci.
- action : indique l'adresse d'envoi des données du formulaire lors de sa soumission.
- method : indique comment seront transmises les données au serveur ("get" ou "post").

Les éléments utilisés à l'intérieur d'un formulaire pour échanger explicitement des informations sont les suivant :

- <button>
- <input> avec les différents types qui lui sont associés (submit, text, checkbox ...)
- <select>
- <textarea>

Les methodes get et post :
La méthode get transmet les informations présentes dans le formulaire via l'url (à la suite du "?"). Ces informations sont donc visibles dans la barre d'adresse du navigateur.
La méthode post envoie les données en plaçant celles-ci dans l'entête de la trame http, elles ne sont alors pas directement visibles.

- **La balise INPUT**

La balise *INPUT* est la balise essentielle des formulaires, car elle permet de créer un bon nombre d'éléments "interactifs". La syntaxe de cette balise est la suivante :

```
<INPUT type="Nom du champ" value="Valeur par défaut" name="Nom de l'élément">
```

L'attribut *name* est essentiel car il permettra au script CGI de connaître le champ associé à la paire nom/valeur, c'est-à-dire que le nom du champ sera suivi du caractère "=" puis de la valeur entrée par l'utilisateur, ou dans le cas contraire de la valeur par défaut repéré par l'attribut *value*.
L'attribut *type* permet de préciser le type d'élément que représente la balise *INPUT*, voici les valeurs que ce champ peut prendre :

- **checkbox**: il s'agit de *cases à cocher* pouvant admettre deux états : *checked* (coché) et *unchecked*(non coché). Lorsque la case est coché la paire nom/valeur est envoyée au CGI

Nom :…… Prénom :……

- **hidden**: il s'agit d'un *champ caché*. Ce champ non visible sur le formulaire permet de préciser un paramètre fixe qui sera envoyé au CGI sous forme de pair nom/valeur
- **file**: il s'agit d'un *champ* permettant à l'utilisateur de préciser l'emplacement d'un fichier qui sera envoyé avec le formulaire. Il faut dans ce cas préciser le type de données pouvant être envoyées grâce à l'attribut *ACCEPT* de la balise *FORM*
- **image**: il s'agit d'un *bouton de soumission personnalisé*, dont l'apparence est l'image situé à l'emplacement précisé par son attribut *SRC*
- **password**: il s'agit d'un *champ de saisie*, dans lequel les caractères saisis apparaissent sous forme d'astérisques afin de camoufler la saisie de l'utilisateur
- **radio**: il s'agit d'un *bouton* permettant un choix parmi plusieurs proposés (l'ensemble des boutons radios devant porter le même attribut *name*. La paire nom/valeur du bouton radio sélectionné sera envoyé au CGI. Un attribut *checked* pour un des boutons permet de préciser le bouton sélectionné par défaut
- **reset**: il s'agit d'un *bouton de remise à zéro* permettant uniquement de rétablir l'ensemble des éléments du formulaire à leurs valeurs par défaut
- **submit**: il s'agit du *bouton de soumission* permettant l'envoi du formulaire. Le texte du bouton peut être précisé grâce à l'attribut *value*
- **text**: il s'agit d'un *champ de saisie* permettant la saisie d'une ligne de texte. La taille du champ peut être définie à l'aide de l'attribut *size* et la taille maximale du texte saisi grâce à l'attribut*maxlength*

Propriétés de la balise Form :

Propriétés	Définition	Valeurs
action	Spécifie l'url de la page qui recevra et traitera les informations soumises	URL
accept-charset	Spécifie quels encodages de caractères sont acceptés par le formulaire.	charset UTF-8 ...
enctype	Indique de quelle façon doivent être encodées les données soumises	application/x-www-form-urlencoded text/plain multipart/form-data
method	Détermine la façon dont sont transférées les données	get post
name	Nom donné au formulaire	Texte
target	Permet de définir comment s'ouvre l'url définié par l'attribut action	_blank _parent _self _top

Exemple à tester :

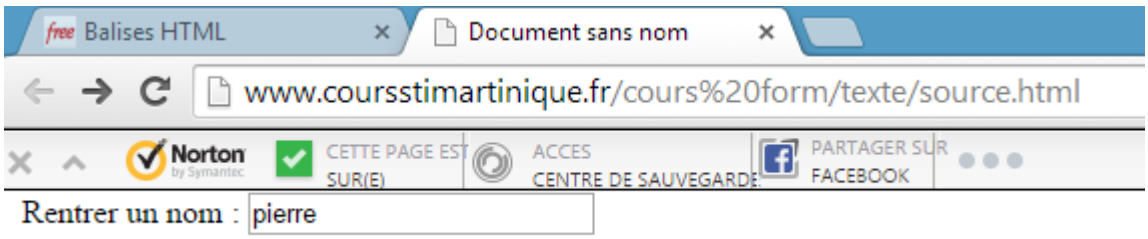
Texte :

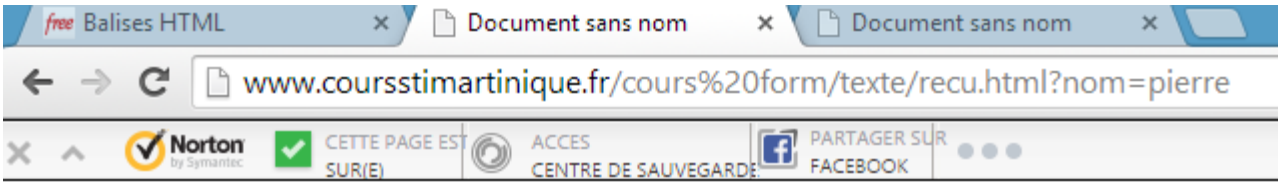
<html>

<head>

Nom :…… Prénom :……

```
<meta charset="utf-8">

<title>Test balise form</title>

</head>


<body>

<form action="http://www.coursstimartinique.fr/cours%20form/texte/recu.html" method="get" name="form1" id="form1">

 <label for="textfield">Rentrer un nom :</label>

 <input type="text" name="nom" id="nom">

</form>

</body>

</html>
```



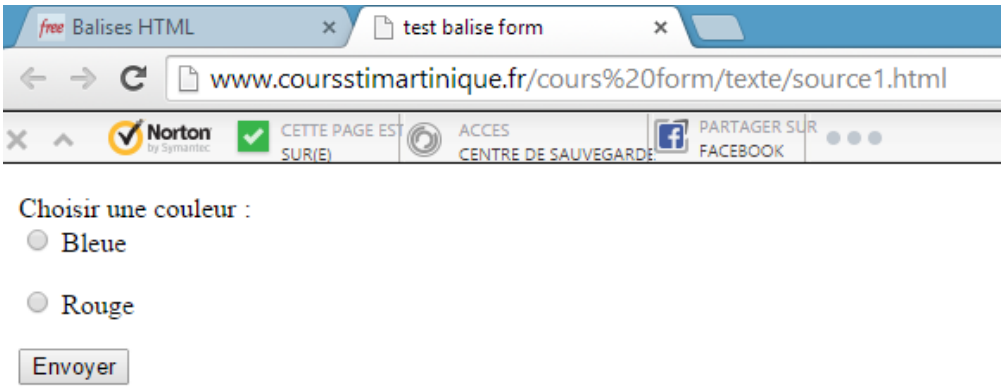


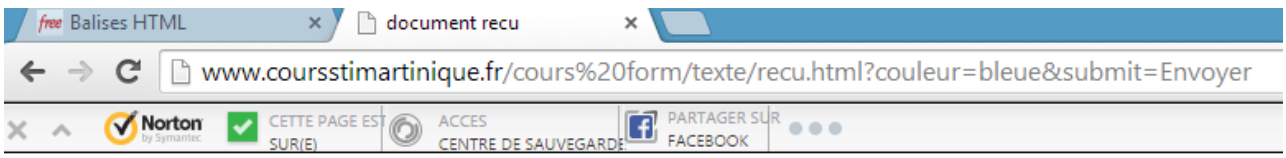
<u>Boutons d'option</u>

```
<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>test balise form</title>

</head>


<body>
```

Nom :…… Prénom :……

```
<FORM action="recu.html" method="get" NAME="form2">

<p>Choisir une couleur : <br>

 <INPUT NAME="couleur" TYPE=radio

VALUE="bleue">

 Bleue</INPUT>

</p>

<p>

 <INPUT NAME="couleur" TYPE=radio

VALUE="rouge">

 Rouge</p>

</INPUT>

<input type="submit" name="submit" id="submit" value="Envoyer">

</FORM>

</body>

</html>
```





- Tester le programme ci-dessous

```
#include <SPI.h>

#include <Client.h>

#include <Ethernet.h>
```

Nom :…… Prénom :……

```
#include <Server.h>

#include <Udp.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //adresse mac de la carte

byte ip[] = { 10,49,18,80};          // adresse IP à définir

byte gateway[] = { 10,49,18,1 };          // adresse IP du routeur à définir

byte subnet[] = { 255, 255, 255, 0 };               //masque sous réseau à définir

EthernetServer server(80);                              //server port

byte sampledata=50;          //some sample data - outputs 2 (ascii = 50 DEC)

int ledPin = 13;

String readString = String(30); //string for fetching data from address

boolean LEDON = false; //LED status flag

void setup(){

//start Ethernet

  Ethernet.begin(mac, ip, gateway, subnet);

//Set pin 13 to output

  pinMode(ledPin, OUTPUT);

   //enable serial datada print

  Serial.begin(9600);

}

void loop(){

// Create a client connection

EthernetClient client = server.available();

  if (client) {

    while (client.connected()) {

   if (client.available()) {

    char c = client.read();

     //read char by char HTTP request

    if (readString.length() < 100)

      {
```

```
      //store characters to string
      readString += c; //replaces readString.append(c);
     }
      //output chars to serial port
      Serial.print(c);
      //if HTTP request has ended
      if (c == '\n') {
       //dirty skip of "GET /favicon.ico HTTP/1.1"
       if (readString.indexOf("?") <0)
       {
//Pas de paramètre dans l'url
       }
       else
       //lets check if LED should be lighted
      if(readString.indexOf("L=11") >0)//replaces if(readString.contains("L=11"))
       {
         //led has to be turned ON
         digitalWrite(ledPin, HIGH);    // set the LED on
         LEDON = true;
       }else
       if(readString.indexOf("L=10") >0)//replaces if(readString.contains("L=10"))
       {
         //led has to be turned OFF
         digitalWrite(ledPin, LOW);    // set the LED OFF
         LEDON = false;
       }
              // now output HTML data starting with standart header
       client.println("HTTP/1.1 200 OK");
       client.println("Content-Type: text/html");
```

```
client.println();

//set background to yellow

client.print("<body style=background-color:yellow>");

//send first heading

        client.println("<hr />");

//output some sample data to browser

client.println("<font color='blue' size='5'>Sample data: ");

client.print(sampledata);//lets output some data

client.println("<br />");//some space between lines

client.println("<hr />");

//drawing simple table

        //printing some link

        //controlling led via checkbox

            if (LEDON)

{ client.println("<form method=get name=LED22><input type=checkbox name=L value=10
CHECKED>LED<br><input type=submit value=submit></form>");}

else

    { client.println("<form method=get name=LED22><input type=checkbox name=L value=11>LED<br><input
type=submit value=submit></form>");

client.println("<br />");}

        //printing LED status

client.print("<font size='5' color='blue'>LED22 status: ");

if (LEDON)

  { client.println("<font color='green' size='5'>ON");}


else

  { client.println("<font color='grey' size='5'>OFF");}

client.println("</body></html>");

//clearing string for next read
```

```
        readString="";

        //stopping client

        client.stop();

         }

        }

       }

     }

 }
```

Remarque :

L'objet String méthode indexOf () vous donne la possibilité de rechercher pour la première instance d'une valeur de caractère particulier dans une chaîne. Vous pouvez également regarder pour la première instance du caractère après une position donnée. La méthode lastIndexOf () vous permet de faire les mêmes choses de la fin d'une chaîne.

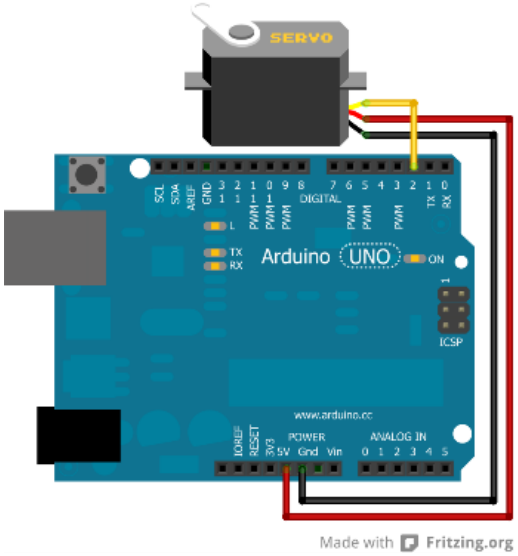
- Modifier le programme pour rajouter une lampe

- On veut commander un servo moteur depuis l'application internet, http://eskimon.fr/287-arduino-602-un-moteur-qui-de-la-tete-le-servo-moteur

CONNECTIQUE

Le servomoteur a besoin de trois fils de connexion pour fonctionner. Deux fils servent à son alimentation, le dernier étant celui qui reçoit le signal de commande :

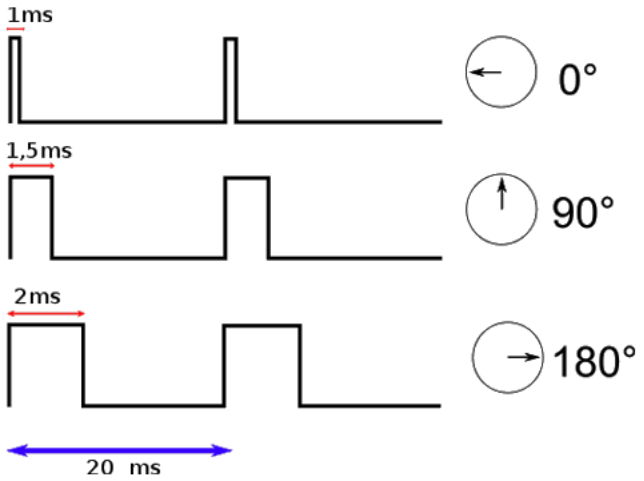
- **rouge** : pour l'alimentation positive (4.5V à 6V en général)
- **noir** ou **marron** : pour la masse (0V)
- **orange**, **jaune**, **blanc**, … : entrée du signal de commande



Nom :…… Prénom :……

Le signal de commande

La consigne envoyée au servomoteur n'est autre qu'un signal électronique de type PWM. Il dispose cependant de deux caractéristiques indispensables pour que le servo puisse comprendre ce qu'on lui demande. À savoir : une fréquence fixe de valeur 50Hz (comme celle du réseau électrique EDF) et d'une durée d'état HAUT elle aussi fixée à certaines limites.



Exemple programmation (Arduino Yun) :

- Arduino

Servo possèdant comme caractéristiques des durées de 1ms pour 0° et 2ms pour 180° et branché sur la broche 12 :

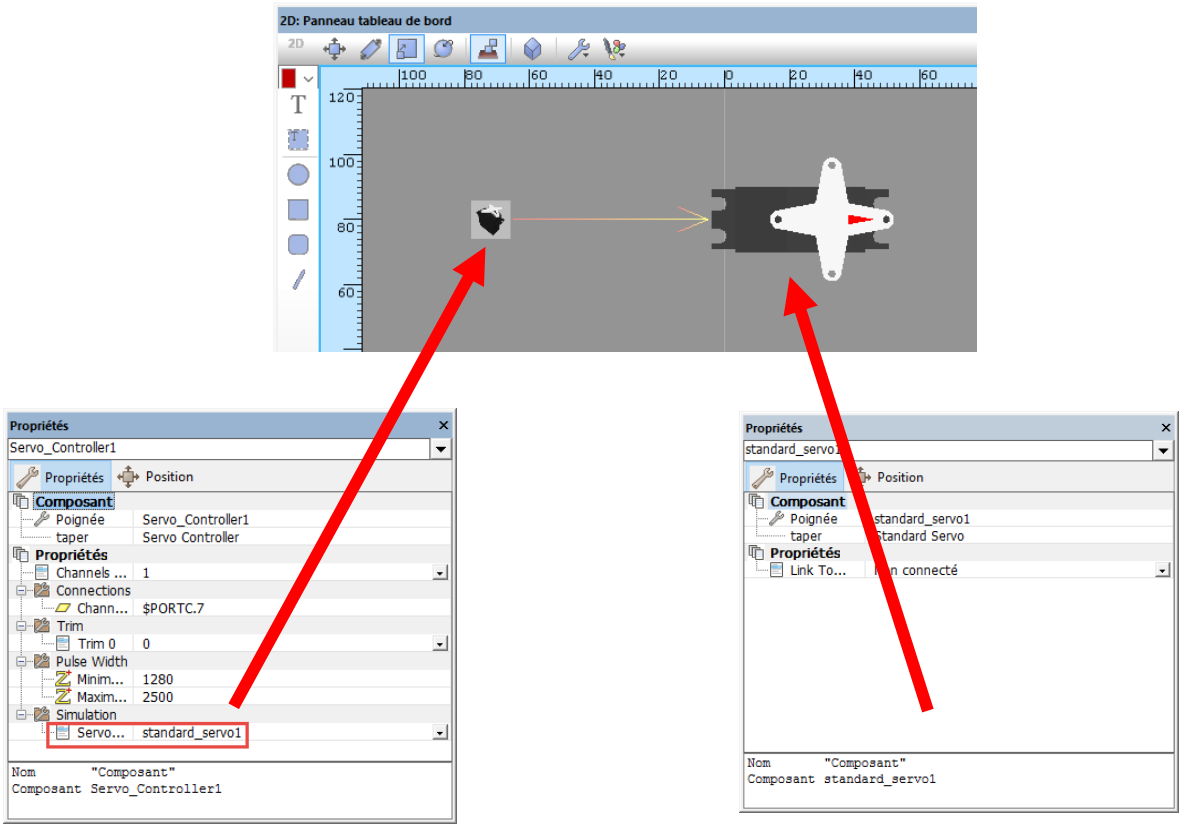
```
#include <Servo.h>

Servo monServo;

void setup()
{
  monServo.attach(12, 1000, 2000) ; // « 12 » est la borne de branchement du moteur sur broche Pin 12
}

void loop()
{
  monServo.write(0) ; // mise en position 0
delay(2000) ;
  monServo.write(180) ; // mise en position 180
delay(2000) ;
}
```

- Flowcode

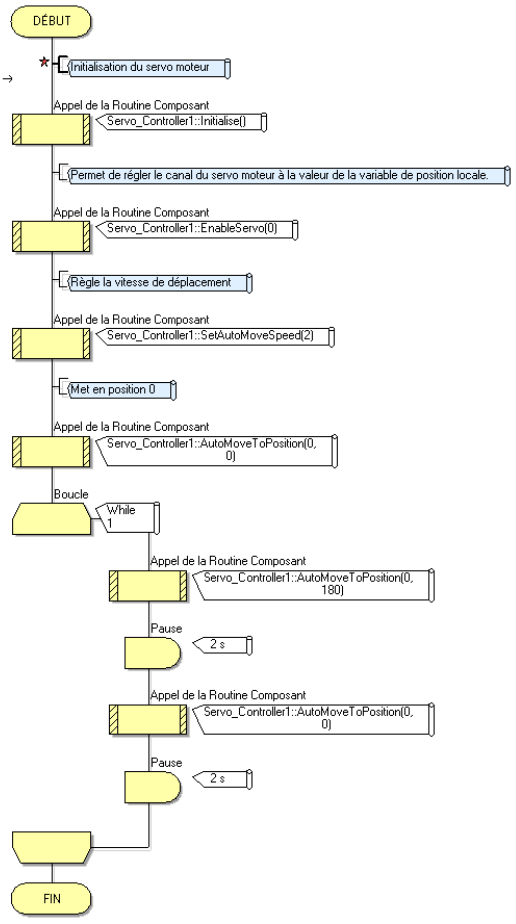
Explication macro : http://www.matrixtsl.com/wiki/index.php?title=Component:_Servo_Controller_(Mechatronics)

Panneau tableau de bord :

Nom :…… Prénom :……

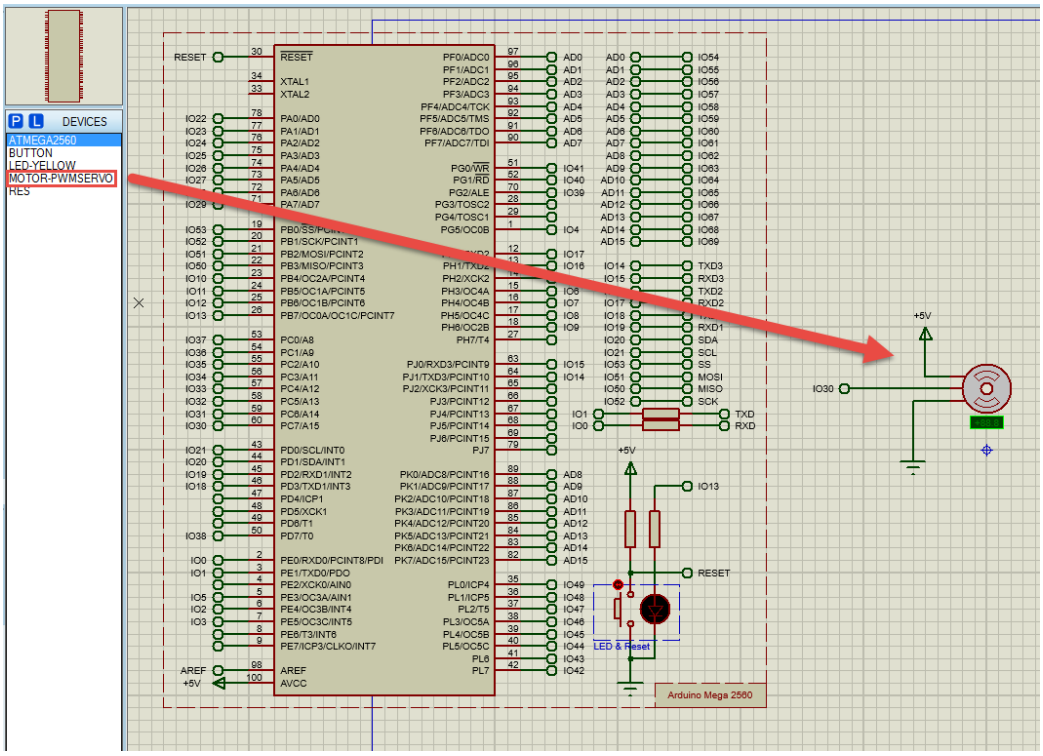


Programme :



Nom :…… Prénom :……

- Proteus (avec carte Arduino Méga)



- Réaliser l'application pour faire tourner le moteur sur trois angles (0,90,180) avec trois boutons
  - Programme Arduino sur carte
  - Programme Flowcode sur carte et Proteus
- Réaliser l'application html ci-dessous pour faire tourner le moteur avec Shield Ethernet

Attention aux guillemets avec Arduino (exemple): client.println("<meta http-equiv=\"refresh\" content=\"2\">");

Voir vidéo :

http://www.sti2dsinhyrome.fr/video%20tp%20shield%20ethernet%20servo%20moteur.html