Systèmes Énergétiques Embarqués

TP2 – Orientation d'un panneau solaire –

Électif NE - Séances 3 et 4

Moussat Ken / Vannavong Kevin

1 - Sommaire

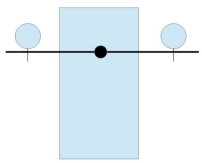
2 - Problème: Orientation d'un panneau solaire

On réalise d'abord le montage. On utilise deux ponts diviseurs de tension afin d'obtenir la tension aux bornes des deux LDR accrochées au moteur (la raison est donnée en détail dans le premier rapport de NE). Par exemple, la valeur de L1 correspond à (VCC*RLDR)/(R+RLDR) où R est l'autre résistance du pont diviseur de tension. Le schéma est le suivant:

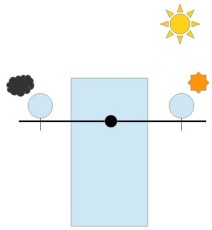
SCHÉMA HERE

2-1)Orientation des photo-résistances

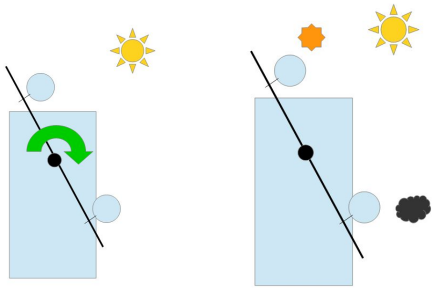
Voici une schématisation du problème:



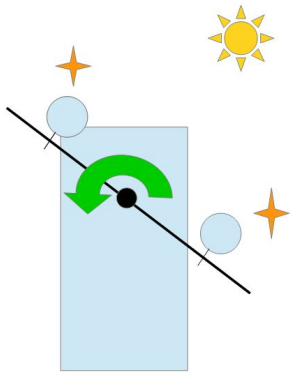
Les deux LDR sont symbolisées par les ronds et la barre noire constitue le rotor du servomoteur. Le système est à son état initial. LDR1 est à droite, LDR2 à gauche.



Supposons qu'un dispositif éclaire le système avec un angle non orthogonal au plan du support des LDR. Dans le cas du schéma ci-dessus, LDR1 est plus éclairée que LDR2. Prenons le cas de LDR1. La luminosité allonge le fil de cuivre du composant augmentant sa résistance. D'après la formule de L1 donnée plus haut, dans le cas présent L1 augmente. Comme diff = L1-L2 où Li est la valeur analogique de la LDRi lue sur son pin analogique, diff est négatif. On fait alors tourner le servomoteur dans le sens de LDR1.



Dans ce cas, le rotor est allé trop loin. LDR2 se retrouve plus éclairée que LDR1. Diff devient positif. La situation actuelle est l'inverse de la précédente. On fait tourner le rotor dans le sens de LDR2 pour rééquilibrer l'exposition des deux composants. Si l'éclairage est bien le même, L1 = L2, diff = 0, le servomoteur n'est plus activé.



Pour utiliser le servomoteur, on inclut d'abord le document <Servo.h> puis on crée un objet de type Servo. En début de programme, les différentes variables sont déclarées et initialisées.

Code:

```
#include <Servo.h>

Servo servo;

int LDR1 = A0;          //pins analogiques récupérant la tension des LDR
int LDR2 = A1;
int L1 = 0;             //stockent les valeurs analogiques de A0 et A1 comprises entre 0 et 1023
int L2 = 0;
int diff=0;             //diff = L1 - L2

int angleServo = 90;    //permet de connaître la position du servomoteur et de la réutiliser
int seuilHaut=90;       //Les seuils compensent le bruit lorsqu'on veut faire tourner le servo dans un sens ou dans l'autre
int seuilBas=-90;       //Le deuxième seuil peut avoir une autre valeur que -seuilHaut, ce qui permet d'avoir par exemple plus de sensibilité dans un sens de rotation ou de compenser un offset si la valeur neutre de diff n'est pas zéro
```

Dans setup, on indique le pin d'entrée du servomoteur et on initialise la position de celui-ci.

Code:

```
void setup()
{
 Serial.begin(9600);    //initialisation de la communication série pour afficher les valeurs de L1, L2 et diff
 servo.attach(9);
 servo.write(angleServo);
}
```

Dans loop, on récupère en boucle les valeurs des tensions entre LDR1 et LDR2 puis on fait leur différence. Si l'éclairage est identique pour les deux, l'exposition à la lumière est optimale et diff = 0. En revanche, si la différence est non nulle, l'une des deux LDR est moins éclairée que l'autre et on fait tourner le servomoteur en conséquence pour rééquilibrer l'exposition à la lumière des deux photo-résistances. Il existe forcément du bruit impliquant une instabilité du servomoteur. Afin de le compenser, on s'assure que la différence de tension captée entre les deux LDR est suffisamment importante pour entraîner une rotation. Afin que cette rotation soit fluide et qu'on puisse contrôler son temps de réaction, on ajoute des delay et on incrémente ou décrémente la valeur de l'angle imposé au servomoteur suivant le sens qu'on souhaite lui donner.

Code:

```
void loop()
{
  //Lecture des tensions entre L1 et L2
 L1 = analogRead(LDR1);
 Serial.print("L1=");
 Serial.println(L1);
 L2 = analogRead(LDR2);
 Serial.print("L2=");
 Serial.println(L2);

 diff=L1-L2;
 Serial.print("Difference:");
 Serial.println(diff);

//équilibrage dans un sens
 if (diff > seuilHaut)        //Seuil haut est modifiable pour plus de précision
 {
        angleServo = angleServo + 1; //On met +1 au lieu de ++ pour symboliser qu'on peut à la place mettre +n avec n>1 pour augmenter la vitesse du déplacement, aux dépens de la précision
        if(angleServo>180)      //On fait attention à ce que l'angle soit inférieur à 180°
        {
                angleServo=180;
        }
        else{}
        servo.write(angleServo);
 }

//équilibrage dans l'autre sens
 else if (diff < seuilBas)
 {
        angleServo = angleServo - 1;
        if (angleServo<0)         //On vérifie que l'angle ne soit pas inférieur à 0°
        {
                angleServo=0;
        }
        else{}
        servo.write(angleServo);
 }
 delay(15);
}
```

Le montage et son fonctionnement a été validé à la fin de la première séance.

2-2) Interface de gestion web

Partie HTML

Formulaire HTML permettant d'envoyer la valeur souhaitée par l'utilisateur avec javascript..

```
<!DOCTYPE html>
<html>
<head>
<title> Arduino TP2 </title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <style type="text/css" media="screen">
      html, body {
        height: 100%;
        overflow: auto;
        margin: 0;
        padding: 0;
        font: 13px Arial, 'Helvetica Neue', sans-serif;
      }
      p {
        margin: 20px;
        padding: 0;
      }
      img {
        border: 0;
      }
      #flashContent {
        display: none;
      }
      #container_app {
        position: absolute;
        top: 0;
        left: -10000;
        width: 100%;
        height: 100%;
        outline: none;
      }
      #wrapper {
        height:100%;
        width:100%;
      }
    </style>

            <link type="text/css" href="login.css" rel="stylesheet" />

  <script type="text/javascript" src="zepto.min.js"></script>
  <script type="text/javascript">

                    function display(){
                    document.write("<span id=\"content\">Waiting for Arduino...</span>");
                    }


  </script>
</head>
<body>
          <form name="msgform" onSubmit="sendMsg()">
            <div id="loginForm">
```

```
  <p class="loginRow" >
    <span class="loginLabel">Valeur à envoyer:</span>
    <input name="msg" class="margeTextInput" type="text" />
   </p>
            <br>
   <p id="loginButtonRow">
     <input class="button blue" type="submit" value="Yun it!" />
   </p>
            </form>
            <script type="text/javascript">
                    function sendMsg() {
$.get('/arduino/msg/' + document.msgform.msg.value + '/',
   function(){
      alert("Success");
   }
);
return false;

}
</script>

            </body>
            </html>
```

Partie Arduino

La connexion sur le réseau Yùn est détaillée dans le premier rapport. Nous nous concentrerons dans ce rapport que sur le code permettant de communiquer avec le client depuis la carte Yùn.

Pour cela on inclut les librairies suivantes:

```
#include <Bridge.h>
#include <YunServer.h>
#include <YunClient.h>
```

Programme test

Avant de débuter le codage du programme demandé, nous ferons d'abord un programme test pour vérifier le fonctionnement de la communication serveur/client. Le programme final ne sera donc qu'une adaptation de la fusion entre le programme test et le programme codé dans la partie précédente.

Le test débute par la création d'un objet de classe YunServer et un objet string qui n'a pas d'utilité fondamentale mais permet de tester l'échange de données entre client et serveur. Dans setup, on lance la communication série pour afficher du texte sur le moniteur série de notre ordinateur et on démarre le bridge. Bridge.h est une interface pour l'exécution de commandes linux. Le serveur est mis en écoute pour intercepter d'éventuels clients.

Code:

```
YunServer server;
String msg;

void setup ()
{
 Serial.begin(9600);
 Bridge.begin();
 server.listenOnLocalhost();
 server.begin();
}
```

A chaque tour de loop, on cherche un nouveau client. Si un client est trouvé, on reçoit son message et exécute des instructions en fonction du message lu. Avant de terminer la boucle, on ferme la communication avec le client.

Code:

```
void loop ()
{
 //*********Read new message from the client**************
 YunClient client = server.accept(); //check new clients

 if(client)
{
        Serial.println("receiving");
        String command = client.readStringUntil('/');  //read the incoming data
        if (command == "msg")
        {
                msg = client.readStringUntil('/');          // read the incoming data
                Serial.println(msg);
        }
        client.stop();
 }
}
```

Ce petit programme test permet depuis l'interface web de faire afficher la carte Yùn un message sur le moniteur série si la commande "msg" est reçue.

Programme final

Revenons au sujet. Nous combinons les lignes précédentes à notre programme écrit en première partie. Pour plus d'efficacité, on ne mettra ici qu'une seule variable de seuil, le seuilHaut correspondant à sa valeur en positif et seuilBas à sa valeur en négatif.

Code:

```
#include <Servo.h>
#include <Bridge.h>
#include <YunServer.h>
#include <YunClient.h>

Servo servo;
YunServer server;

int LDR1 = A0;
int LDR2 = A1;
int L1 = 0;
int L2 = 0;
int diff=0;

int angleServo = 90;
int seuil=90;


void setup()
{
	Serial.begin(9600);
	Bridge.begin();
	server.listenOnLocalhost();
	server.begin();

	servo.attach(9);
	servo.write(angleServo);
}
```

Cette fois-ci, après avoir capté les valeurs L1 et L2, on ne les affichera pas sur le moniteur série, ce qui nous fera gagner beaucoup de temps d'exécution. On cherche à la place un client. En fonction de la commande reçue, le programme affiche sur le moniteur série les valeurs de L1 et L2, le sens de rotation du moteur ou modifie le seuil en fonction du code envoyé.

Le programme poursuit comme avant en inclinant le plan des LDR de +1 ou -1.

Code:

```
void loop()
{
	L1 = analogRead(LDR1);
	L2 = analogRead(LDR2);
	diff=L1-L2;

	YunClient client = server.accept();
	if (client)
	{
		String command = client.readString();
```

```
        switch(command)
        {
                case “valeur”:
                        client.print(“L1=”);
                        client.println(L1);
                        client.print(“L2=”);
                        client.println(L2);
                        break;

                case “sens”:
                        if (diff<0)
                                client.println("rotation sens horaire");
                        else
                                client.println("rotation sens anti-horaire");
                        break;

                case “tout”:
                        client.print(“L1=”);
                        client.println(L1);
                        client.print(“L2=”);
                        client.println(L2);
                        if (diff<0)
                                client.println("rotation sens horaire");
                        else
                                client.println("rotation sens anti-horaire");
                        break;

                default:
                        command.trim();
                        seuil=command.toInt();
                        break;
        }
        client.stop();
}

//Reprise du code initial
if (diff > seuil)
{
        angleServo = angleServo + 1;
        if(angleServo>180)
        {
                angleServo=180;
        }
        else{}
        servo.write(angleServo);
```

```
        }
        else if (diff < -seuil)
        {
                angleServo = angleServo - 1;
                if (angleServo<0)
                {
                        angleServo=0;
                }
                else{}
                servo.write(angleServo);
        }
        delay(15);
}
```