Développement durable en SIN

LOÏC JOSSE *

Le développement durable fait partie intégrante de nos enseignements en STI2D. Cependant, alors qu'il est facile d'y associer des ressources et applications sur les énergies oules matériaux, l'information est quant à elle souvent en reste. Or c'est un chapitre que les élèves de SIN ne peuvent ignorer, puisqu'il est évalué lors de l'oral de projet.

MOTS-CLÉS

lycée technologique, prébac, SIN, programmation, développement durable

Les notions de développement durable évaluées lors de la soutenance orale de projet sont les suivantes (*Bulletin officiel* n° 39 du 23 octobre 2014) :

– le choix des matériaux et/ou des matériels est justifié, des critères d'écoconception sont pris en compte ;

– la justification des paramètres de confort et/ou la réponse apportée par le système aux contraintes de préservation de la santé et du respect de la sécurité sont explicitées ;

– la relation entre une fonction, des solutions et leur impact environnemental ou sociétal est précisée ;

– le compromis technico-économique et/ou la prise en compte des normes et réglementations est expliqué.

Bien souvent et malheureusement, ces quatre critères ne sont pas développés par les candidats ayant pris la spécialité SIN. Un élève brillant n'ayant pas abordé ces points peut perdre entre 3 et 4 points sur 20 pour peu qu'il se soit focalisé sur la programmation, par exemple.

Quelques définition

Le développement durable est « un développement qui répond aux besoins du présent sans compromettre la capacité des générations futures à répondre aux leurs », citation de M^me^ Gro Harlem Brundtland, Premier ministre norvégien (1987).

En 1992, le Sommet de la Terre à Rio, tenu sous l'égide des Nations unies, officialise la notion de développement durable et celle de ses trois piliers (économie, environnement, social) : un développement économiquement efficace, socialement équitable et écologiquement soutenable.

* Enseignant SII option information et numérique, lycée Louis-le-Grand, Paris.

Enfin, selon l'Afnor, « l'écoconception consiste à intégrer l'environnement dès la conception d'un produit ou service, et lors de toutes les étapes de son cycle de vie ». Elle prend en compte toutes les étapes du cycle de vie du produit, depuis l'extraction des matières première jusqu'au traitement en fin de vie, en passant par la fabrication, le transport et l'utilisation. Pour résumer, nous devons concevoir des produits qui répondent à un besoin sociétal tout en réduisant les impacts environnementaux et les coûts.

Un groupe de projet SIN peut toujours aborder les notions d'alimentation électrique et faire une étude comparative : de quelle quantité d'énergie électrique a-t-on besoin ? Pendant combien de temps ? De quelle énergie dispose-t-on ? À quel prix ? etc. Il peut aussi envisager d'effectuer une étude sur le matériau utilisé comme support de la carte électronique s'il existe et proposer, là encore, une étude comparative.

Les deux items précédents sont, en règle générale, ceux abordés par défaut lors des oraux de cette spécialité et, en tant qu'examinateur, on souhaiterait aussi voir émerger d'autres axes de recherche.

Étude de cas

Imaginons un système dans lequel un serveur web indiquerait, entre autres, la température ambiante d'un local, tâche qui incombe à un élève. Nous ne sommes vraisemblablement pas en présence d'un sous-système très énergivore en courant électrique et n'ayant pas forcément besoin d'un boîtier fabriqué avec un matériau recyclable ou de faible bilan carbone. Pour l'élève, la nécessité de réaliser une partie de son exposé sur le développement durable devient alors bien contraignante.

Première solution

Pour réaliser son sous-système, l'élève choisit, par défaut, une carte Arduino Uno associée à une carte Ethernet, elle-même munie d'une carte SD. La température est mesurée à l'aide d'un capteur Grove (sur son support Grove Arduino) et l'alimentation électrique réalisée par un adaptateur secteur à tension de sortie régulée commutable entre 3 et 12 Vcc et positionnée sur 12 V 1. L'ensemble coûte environ 85 euros.

Le programme Arduino contient le serveur que l'on associe à une page index.htm située sur la carte SD du *shield* Ethernet. Le client se connecte au serveur via son adresse IP. La valeur de la température est alors envoyée au client qui voit sa page web rafraîchie toutes les cinq secondes. La mémoire de l'Arduino Uno étant limitée, la page index.htm intègre la mise en forme CSS ainsi que le code Javascript, mais ne contient pas d'image.

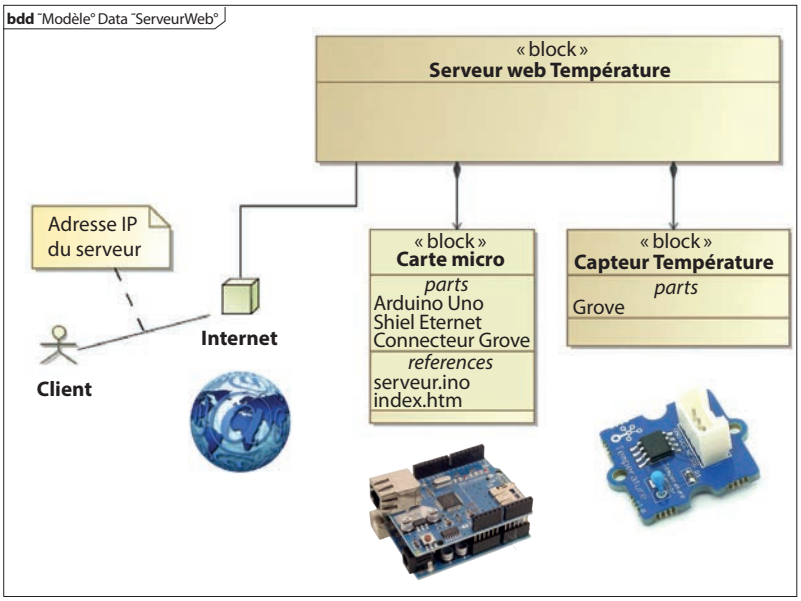
L'établissement du bilan énergétique et ici assez simple : la carte Arduino munie du *shield* Ethernet consomme 216 mA, le capteur de température 1 mA et l'ensemble reste en fonctionnement 24 h/24. Soit (217 × 24) = 5 208 mA·h/j.

Ce résultat est assez médiocre, car si l'on devait alimenter notre carte avec un accumulateur de 9 V/200 mA·h, nous ne disposerions que d'une autonomie de 55 minutes environ. Alors quels sont les moyens que nous pouvons mettre en œuvre pour diminuer la consommation électrique et réduire les coûts ?

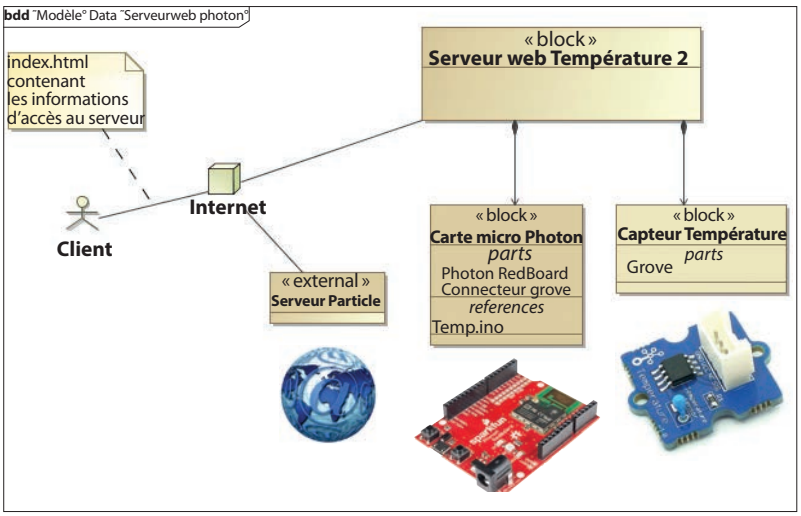
On pourrait commencer par changer de capteur, mais au vu de sa faible consommation par rapport à l'ensemble formé par les deux cartes électroniques, l'autonomie ne serait augmentée que de quelques secondes.

Sachant qu'il n'est pas nécessaire d'envoyer constamment la valeur de la température, nous pouvons mettre le système en veille, puis le réveiller toutes les huit secondes (temps maximum pour réveiller l'Arduino en utilisant le *Watchdog Timer* – voir Encadré). Malheureusement, en mode veille, notre système consomme encore 196 mA environ et toujours 217 mA en fonctionnement normal, mais que pendant une seconde. Le nouveau bilan énergétique devient : [(196 × 8 + 217) / 9] × 24 = 4 760 mA·h/j, ce qui nous ramène à une autonomie d'une heure approximativement avec notre accumulateur de 9 V/200 mA·h. Avec un adaptateur secteur 12 V, en multipliant par 365 jours, la consommation annuelle est à peu près de 1 737 kW·h. À 0,145 euro le kW, la dépense est de 251 € contre 275 € précédemment.

En fonctionnement normal, la carte Arduino nécessite à peu près 55 mA et la carte *shield* Ethernet 160 mA. Remplaçons alors ces deux cartes !



1 **Diagramme de bloc du premier montage**



2 **Diagramme de bloc du deuxième montage**

Deuxième solution

La photon RedBoard présentée dans le n° 202 de *Technologie* paraît appropriée 2. L'économie, par rapport à la solution précédente, est d'environ 12 euros.

Le contexte est alors légèrement modifié : la carte récupère la valeur de la température du local et l'envoie vers le serveur de Particle. Le client utilise une page HTML contenant les coordonnés d'accès au serveur pour lire cette valeur. Nous n'utilisons ici qu'une seule carte électronique, qui consomme 102 mA, soit 2 472 mA·h/j avec le capteur.

Nous avons divisé par deux et plus la consommation électrique, réduit le nombre de cartes (donc l'encombrement, mais aussi les soucis de connectique), ainsi que le prix d'achat.

Codes Arduino des deux montages

Côté logiciel, il fallait réaliser une page HTML contenant l'affichage de la température mise en valeur avec un thermomètre animé 3. Passons en revue les différents programmes à réaliser pour ces deux types de configuration.

Pour le premier montage, la mise en œuvre du serveur dans la carte Arduino n'est pas simple pour un élève de terminale ; heureusement, les constructeurs proposent des exemples à charger en bibliothèque. Nous nous sommes donc appuyés sur un de ces exemples pour réaliser le code 4.

Ce programme capte la valeur de la température (lignes 15 à 22) et la transmet au client qui en demande l'accès (lignes 46 et 47). Pour un meilleur visuel, cette valeur est accompagnée d'une enveloppe, la page index.htm (lignes 67 à 76), placée dans la carte SD.

Les commentaires vous permettront de comprendre son fonctionnement mais repérez, sur les lignes 64 à 66, le code HTML qui permet de passer la valeur de la température à la page web du client. On utilise une donnée de type texte cachée, dans une balise <p> identifiée « En1 ».

Sur la carte SD, nous plaçons le fichier index.htm 5.

Ce code HTML ne fait pas appel à des fichiers externes et contient donc la mise en forme CSS (lignes 7 à 29) et le code Javascript pour le thermomètre (lignes 30 à 75). On récupère la valeur de la température avec l'instruction « document.getElementById('En1').innerHTML; » (lignes 35 et 67). La page, et par conséquent la valeur de la température, est rafraîchie toutes les cinq secondes par la ligne 6.

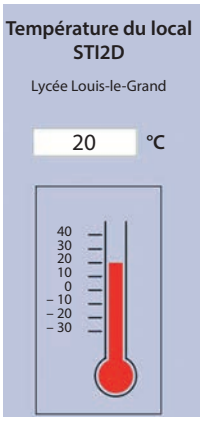
Pour le deuxième montage, il suffit de transférer le code 6 dans la carte. Ce code est bien plus simple, il ne fait que lire la valeur de la température (ligne 15), l'adapter et la renvoyer dans une variable stockée sur le *cloud* de Particle (ligne 18).

Pour la page HTML, nous reprendrons le fichier index.htm précédent auquel nous ferons quelques modifications :

– d'abord, pour communiquer avec le site Particle et récupérer la valeur de la température, nous utilisons un script Ajax. Nous devons insérer la ligne suivante, au début du fichier, après la balise <html> :
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js" type="text/javascript" charset="utf-8"></script> ;

– on supprime la ligne 6 ainsi que les lignes 20 à 23 qui ne sont plus nécessaires ;

– on remplace les fonctions Recup() et Charger() par les suivantes 7.



3 Page html reçue par le client

Placer l'Arduino en mode veille et le « réveiller » toutes les huit secondes

Le mode veille permet à la carte Arduino Uno de passer de 55 mA à 38 mA environ. Sur cette carte, ce sont surtout les composants autour du microcontrôleur qui sont énergivores.

En premier lieu, il faut insérer les bibliothèques nécessaires à « l'endormissement », à la gestion de l'énergie et à la répétition du « réveil » : <avr/sleep.h>, <avr/power.h> et <avr/wdt.h>.

Dans le Setup(), vous écrivez les lignes suivantes afin d'initialiser les registres internes liés au *Watchdog* pour réveiller le microcontrôleur toutes les huit secondes :

```
MCUSR = MCUSR & B11110111; // Mettre à 0 reset flag Watchdog
WDTCSR = WDTCSR | B00011000; // Valide le reset et les changements du Watchdog
WDTCSR = B00100001; // période à 8 s
WDTCSR = WDTCSR | B01000000; // Valide l'interruption Watchdog
```

Il faut encore ajouter deux fonctions, l'une pour sortir du mode veille et l'autre pour y entrer. La première fonction doit être très courte, voire vide comme notre exemple. Elle peut contenir des variables globales comme l'incrémentation d'un compteur mais pas de *delay :*

```
ISR(WDT_vect)
{
}
void enterSleep(void)
{
set_sleep_mode(SLEEP_MODE_PWR_DOWN);
  sleep_enable(); // Valide le mode veille
sleep_mode(); // Mode veille
// Retour de veille
sleep_disable(); // Invalide le mode veille après interruption
power_all_enable();
}
```

Une dernière ligne de code doit être placée en fin de programme Loop(), celle qui appelle la fonction de mise en veille :

```
enterSleep();
```

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

// adresse MAC située sous la carte
byte mac[] = { 0x90, 0xA2, 0xDA, 0x0E, 0xC4, 0x30 };
IPAddress ip(172, 16, 4, 252); // adresse IP attribuée
EthernetServer server(80);  // port 80
const int an0pin = A0;
const int B=3975;
float temperature;
float resistance;
float a;

//fonction acquisition de la température
float gettemp()
{
  a = analogRead(an0pin);
  resistance=(1023-a)*10000/a;
  temperature=1 / (log(resistance/10000)/B+1/298.15)-273.15;
  return temperature;
}

File webFile;

void setup()
{
    // initialiser la carte Ethernet
    Ethernet.begin(mac, ip);
    server.begin();

    Serial.begin(9600);

    // initialiser la carte SD
    Serial.println("Initialise carte SD...");
    if (!SD.begin(4))
    {
        Serial.println("ERROR - carte SD non initialisee!")
        return;
    }
    Serial.println("SUCCESS - carte SD initialisee.");
}
void loop()
{
    temperature = gettemp(); //Acquérir la température
    EthernetClient client = server.available();
    if (client)
    {

        Serial.println("client connecte");
        boolean currentLineIsBlank = true;
        while (client.connected())
        {
            if (client.available())
            {
                char c = client.read(); // lecture d'un caractère
                Serial.print(c);
                if (c == '\n' && currentLineIsBlank)
                {
                    // envoi d'un contenu HTML au client
                    client.println("HTTP/1.1 200 OK");
                    client.println("Content-Type: text/html");
                    client.println("Connection: close");
                    client.println();
                    client.print("<p id='En1' class='hidden'>");
                    client.print(temperature); //valeur de la température "cachée"
                    client.println("</P>");
                    webFile = SD.open("index.htm"); //ouverture de la page index.htm
                    if (webFile)
                    {
                        while(webFile.available())
                        {
                            client.write(webFile.read()); //envoi de la page au client
                        }
                        webFile.close();
                    }
                    break;
                }
                if (c == '\n') //détection de la fin de ligne
                {
                    currentLineIsBlank = true;
                }
                else if (c != '\r') //détection fin
                {
                    currentLineIsBlank = false;
                }
            }
        }
        delay(1);
        client.stop(); //fermer la connection
        Serial.println("client deconnecte");
    }
}
```

4 Code Arduino du premier montage

```
<html>
  <head>
    <meta charset="utf-8">
    <meta content="L. JOSSE" name="author">
    <title>Serveur Web Température</title>
    <meta http-equiv="refresh" content="5">
    <style>
        body
        {
            margin: 10mm;
            background-color: #ccccff;
            border:1px solid black;
            text-align:center;
        }
        h1
        {
            font-style:italic;
            text-align:center;
        }
        .hidden
        {
            display: none;
        }
        input
        {
            text-align:center;
            font-size:xx-large;
        }
    </style>
    <script type="text/javascript">
        function thermometre()
        {
            var c = document.getElementById("mon_canvas");
            var ctx = c.getContext("2d");
            var N = document.getElementById('En1').innerHTML;
            ctx.clearRect(0,0,180,330);
            //Le thermomètre
            ctx.lineWidth=4;
            ctx.beginPath();
            ctx.moveTo(100,50);
            ctx.lineTo(100,250);
            ctx.moveTo(130,50);
            ctx.lineTo(130,250);
            ctx.arc(115,275,30,Math.PI*1.7,Math.PI*1.35,false);
            // Les graduations
            ctx.lineWidth=2;
            var graduation = 0;
            var pas = 20;
            for(var i=0;i<8;i++)
            {
                ctx.moveTo(95,70+pas*i);
                ctx.lineTo(75,70+pas*i);
                graduation += pas/2;
                ctx.font="20px arial";
                ctx.fillText(graduation-40,30,215-pas*i);
            }
            ctx.stroke();
            //Le mercure
            ctx.beginPath();
            ctx.fillStyle = "red";
            ctx.arc(115,275,25,Math.PI*1.66,Math.PI*1.34,false);
            ctx.rect(105,253,20,0-2*N-103);
            ctx.fill();
        }
        function recup()
        {
            var valeur = document.getElementById('En1').innerHTML;
            document.getElementById('AN0').value = valeur;
        }
        function charger()
        {
            recup();
            thermometre();
        }
    </script>
  </head>
  <body onload="charger();">
    <div contenu>
        <h1>Température du local<br>STI2D</h1>Lycée Louis Le Grand Paris<br><br>
        <h1><input type="text" id="AN0" value="0" size="5"> °C</h1>
        <br>
        <canvas id="mon_canvas" width="180" height="330" style="border:1px solid black">
        Pas de canvas !</canvas>
        <br>
  </div>
  </body>
</html>
```

5 Contenu du fichier index.htm

```
#include <math.h>
const int an0pin = A0;
const int B=3975;
float temperature;
float resistance;
char resultstr[5];
float a;
void setup()
{
   pinMode(an0pin, INPUT);
   Particle.variable("analogvalue", resultstr, STRING);
}
void loop()
{
    a = analogRead(an0pin)/ 4;
    resistance=(1023-a)*10000/a;
    temperature=1 / (log(resistance / 10000) / B + 1 / 298.15) - 273.15;
    sprintf(resultstr, "%f",temperature);
    delay(100);
}
```

6 Code Arduino du deuxième montage (plus léger !)

```
62          function recup()
63          {
64              requestURL = "https://api.particle.io/v1/devices/N°carte/analogvalue?access_token=N°jeton";
65              $.getJSON(requestURL, function(json)
66              {
67                  document.getElementById("AN0").value = Math.round(json.result*100)/100;
68                  thermometre();
69               });
70          }
71          function charger()
72          {
73              thermometre();
74              intervalID = setInterval(recup, 5000);
75          }
```

7 Les fonctions à remplacer

Les lignes 64 et 65 permettent de récupérer la valeur de la température stockée sur le *cloud* de Particle, la ligne 67 arrondit cette valeur au centième. La ligne 74 permet de rafraîchir la valeur de la température en relançant la fonction Recup() toutes les cinq secondes.

La deuxième solution, plus simple à mettre en œuvre, permet également de créer une page web plus riche avec un meilleur confort visuel. Les différents codes HTML, CSS ou Javascript peuvent être séparés pour une lecture simplifiée et l'ajout d'images ne pose aucun problème.

Conclusion

Il existe une multitude de cartes électroniques qui, souvent, proposent des fonctionnalités similaires. En partant d'un matériel présent dans le lycée, le candidat peut nous proposer une analyse comparative ou nous montrer l'évolution, dans le sens du développement durable, de sa solution. Il pourra alors exposer les coûts ainsi que les consommations en s'appuyant sur des expérimentations.

Si, exceptionnellement, l'élève ne peut nous exposer que du *software*, il faudrait qu'il travaille sur l'efficacité des programmes et la complexité des algorithmes en proposant, là encore, de comparer des solutions. L'idée est d'obtenir un code simple, bien structuré, qui gagne en rapidité tout en limitant ses ressources en mémoire, même si celle-ci est de moins en moins chère. ■

POUR EN SAVOIR PLUS

F. Bordage, *Écoconception web : les 115 bonnes pratiques*, Paris, Eyrolles, 2015.

R. Lacoste, M. Robiolle, X. Vital, *L'Écoconception en électronique*, Paris, Dunod, 2011.

EN LIGNE

http://donalmorrissey.blogspot.fr/

Tous les liens sur http://eduscol.education.fr/sti/revue-technologie