Module SIN21
Mise à jour de l'IHM d'affichage des données météo

Temps : 3h

Objectifs :

- Modifier l'IHM d'affichage des données météo.
- Utilisation simple d'un environnement de développement en C++
- Analyser et modifier du code C++

Prérequis :

- langage C.
- opération de base sur les masquages et les décalages.
- maitrise de la conversion hexadécimal <-> binaire.

Documents/ressources nécessaires :

- le protocole de la station : **SIN21_WMR928Protocol.pdf**
- le projet de base utilisé dans la première activité du parcours
  - situé dans le répertoire **SIN21_projet_builder_logiciel_client_de_base_pour_stagiaires.**
- la vidéo sur la présentation à l'utilisation de l'environnement C++ Builder : **SIN21_video_cpp_builder.mp4**

1 Introduction

Le choix d'un environnement de développement et d'un langage dépend de nombreux paramètres. Il est difficile de dire qu'un langage est meilleur qu'un autre car tout dépend du contexte. On aurait pu développer le logiciel sur lequel nous allons travailler, en Java, en C#, en Basic, en C ou encore en Python qui sont tous des langages populaires. Il en est un qui sort tout de même du lot, c'est le C++ avec l'environnement **C++ Builder** de la société **Embarcadero** (racheté à **Borland**). Les gros avantages sont :

- Le C/C++ est un langage pour l'informatique industrielle.
- L'environnement C++ Builder permet de mettre au point une IHM très facilement.
- La gestion des évènements (clic de souris, appuie sur des touches, etc) est aussi très simple.
- Il n'est pas du tout nécessaire de connaitre sur le bout des doigts le C++ pour pouvoir coder, cependant une bonne base de C standard est requise (ce que les collègues électroniciens ont puisque les projets de bacs intègrent bien souvent un microcontrôleur).

Remarque 1 : *Pour la prise en main de l'environnement de développement ainsi qu'une introduction très succincte au C++, veuillez utiliser la vidéo idoine.*

Remarque 2 : *Pour l'utilisation de tableau en C/C++, voir l'annexe 1 ci-dessous*

Remarque 3 : *Pour les opérations de masquage de bits, voir l'annexe 2.*

2 Etude du logiciel client de base.

A ce stade nous possédons un logiciel permettant d'afficher la température, la pression et le taux d'humidité. Nous allons compléter l'IHM en affichant l'état de la batterie, le point de rosée et l'état de la météo.

2.1 Fonctionnement global

Le fonctionnement est assez simple. Il suffit de choisir le port série sur lequel l'information sera envoyée, puis de choisir si les données seront fixes (les valeurs seront fixées par les différents champs présents dans l'interface), ou bien variables (tirées de manière aléatoire et toujours changeantes).

2.2 Etude du protocole du WMR928

Le fichier **WMR928Protocol.pdf** permettant de prendre connaissance du protocole présente plusieurs versions pour plusieurs appareils. La colonne qui nous intéresse est notée **EXTBTH** (la dernière).

Questionnement

A partir du document :

1. *Repérez l'octet et le bit portant l'information de l'état de la batterie.*
2. *Repérez l'octet portant l'information sur le point de rosée. Quel code binaire est utilisé ? (binaire naturel, BCD,…).*
3. *Enfin, repérez l'octet puis les bits portant l'information de l'état du temps (vous avez leur signification en bas du document).*

2.3 Etude du traitement de la trame.

Dans le logiciel client, le traitement de l'information se fait dans un évènement généré par le **Timer** toutes les secondes (voir vidéo pour comprendre son fonctionnement). Le code est le suivant :

```
//Réception des octets (ne fait pas partie de l'étude)

/*******************************************************/
//Début du traitement des informations dans la trame reçue
//pour affichage
/*******************************************************/

//Traitement des informations pour affichage

//temperature
centaine = (trame[5] & 0x30) >> 4;  //
dizaine = (trame[5] & 0x0F);        //
unite = (trame[4] & 0xF0) >> 4;     // traitement

EditTemp->Text = AnsiString(centaine*100 + dizaine*10 + unite); // affichage
//Fin traitement température

//humidité
unite = (trame[6] & 0x0F);            //
dizaine = (trame[6] & 0xF0 ) >> 4;    // traitement

EditHumid->Text = AnsiString(dizaine*10 + unite);  // affichage
//Fin traitement humidité
```

```
//pression
pression = (int)trame[8];    //
if ( trame[9] & 1 )          //
       pression +=512;          // traitement

EditPression->Text = AnsiString(pression+600); // affichage
//Fin traitement pression

//*******************************************
//   A COMPLETER POUR AFFICHER LES AUTRES DONNÉES
//*******************************************
```

Rappel : l'indice d'un tableau en C/C++ commence à la valeur 0. La première case se désigne donc comme ceci : **tab[0]** (si votre tableau s'appelle **tab**).

Questionnement

En repérant les octets porteurs de l'information « température » sur le document du protocole, justifiez/expliquez les 3 lignes de traitement pour extraire l'information et la placer dans la variable ***température****. Vous utiliserez les questions ci-dessous.*

1ère ligne de code

```
centaine = trame[5] & 0x30 >> 4 ;
```

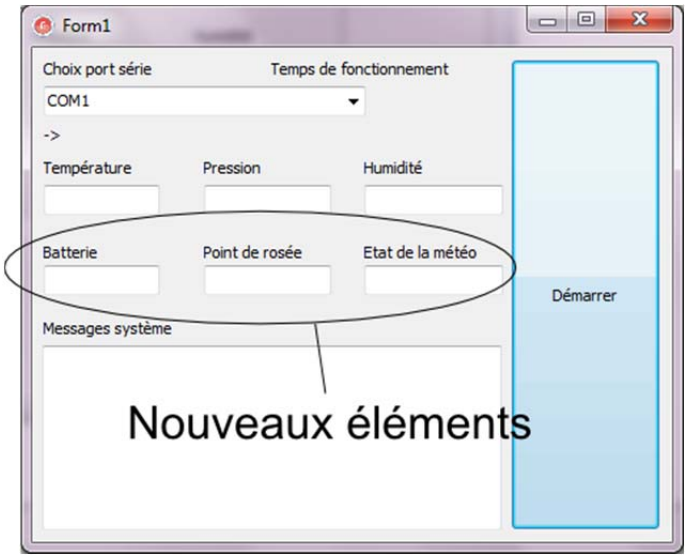
- **Trame[5] :** que représente cette case ? Que contient-elle ?
- **& 0x30** : quelle est l'opération logique réalisée avec l'opérateur & ? En transformant la valeur 0x30 en binaire, quels sont les bits qui resteront dans le résultat final ?
- **>> 4** : que réalise l'opérateur **>>** ? Justifiez le chiffre 4.

Reprenez le même travail pour les 2 autres lignes.

3 Modification du programme

Travail à faire

1. Dans un premier temps vous allez positionner sur la fiche les éléments nécessaires à l'affichage. Comme nous avons 3 informations différentes à afficher, nous utiliserons donc 3 nouveaux **TEdit** et 3 nouveaux **TLabel**. L'IHM pourra ressembler à ceci (mais ce n'est qu'un exemple. Tous les éléments sont redimensionnables et déplaçables) :



2. En s'inspirant du code déjà fourni, complétez la suite du code ci-dessus (utilisée par le Timer) pour que l'affichage des nouvelles données se fassent dans les composants que vous venez d'ajouter .

Annexe 1 – Les tableaux en C/C++

Les tableaux sont des structures de données de base. Ils permettent de déclarer un ensemble de cases de même nature sous un même nom de variable. La déclaration est la suivante :

<TYPE> <NOM_DU_TABLEAU>**[**<NOMBRE_DE_CASE>**]**;

Exemples :

- pour un tableau de char :
  - `unsigned char tableau[20]; //-> ceci déclare un tableau de 20 cases, chacune ayant une taille de 1 octet.`
- pour un tableau de int (entier) :
  - `unsigned int unAutreTableau[200]; //-> ceci déclare un tableau de 200 cases, chacune ayant une taille de 4 octets (un int ayant une taille de 4 octets en général).`

Une fois un tableau déclaré, la lecture et l'écriture dans les différentes cases se fait facilement de la manière suivante :

- lecture :
  - `var = tableau[3]; //ceci est un exemple`
  - `var = tableau[0]; //la première case a l'indice 0 !`
  - `var = tableau[i]; //on peut utiliser une autre variable pour désigner une case, cette variable étant de type entier (int).`
- écriture :
  - `unAutreTableau[0] = 3; //on écrit dans la première case.`
  - `unAutreTableau[i] = 7; //on écrit à la i-ème case.`

Annexe 2 – Les opérations sur bits

Plusieurs cas de base sont à considérer :

- l'extraction d'information : on souhaite connaitre la valeur d'un seul bit dans une variable (quelle qu'en soit la taille). Pour cela on utilisera l'opérateur permettant de réaliser un ET logique bit à bit. Par exemple pour connaitre la valeur du cinquième bit de la variable **var** on écrira : `resultat = var & 0x10;`
  - **explications :**
    - l'opération ET logique est donné par la présence de l'opérateur **&**.
    - si on cherche à éliminer les autres bits, il faut un masque binaire 0001000, qui vaut 0x10 en notation hexadécimale.
    - Le tout est rangé dans la variable **resultat**. La variable **var** n'est pas modifiée !
- si on veut extraire un ensemble de bits il suffit de prendre un masque plus large. Par exemple, si on cherche à garder les 4 derniers bits d'une variable on écrira : `variable = variable & 0xF0;` (masque binaire 11110000)
- On peut aussi décaler des bits dans une variable en utilisant l'opérateur "**>>**" ou bien "**<<**" . Par exemple, si on souhaite décaler les 4 derniers bits de 4 rangs vers la droite (ils deviennent ainsi les 4 bits de poids faible) il faut écrire : `var = var >> 4;`
- **Remarques :** attention au type des variables ! Le résultat du décalage n'est pas identique si la variable est signée ou pas !!