TP n°2 : Windows 10 IoT Standard

A faire sur Windows 10 avec votre session 3iL ou sur votre PC

1. - Prise en main de Windows 10 IoT Standard :

L'objectif va consister à créer un Windows 10 IoT Standard/Core et à tester le développement d'une application embarquée UWP fonctionnant avec cet OS sur carte Rasberry PI 3.

Vous devez disposer d'une Rasberry Pi3 avec une carte micro SD Windows 10 IoT core, d'un poste Windows 10 avec Visual Studio (session 3iL normale).

Windows 10 IoT Core pour Rasberry doit être installé sur la carte micro SD. Pour le faire il suffit d'aller sur le centre de téléchargement sur https://developer.microsoft.com/en-us/windows/iot/downloads et d'installer « Windows 10 IoT Core Dashboard ».
Après avoir lancé cet outil, sélectionnez « Broadcomm (Raspberry Pi 2 & 3) », « Windows 10 IoT Core (17763) », votre carte SD en USB, le nom système de votre OS, un mot de passe d'administrateur (admin par exemple), sans Wifi et faites « Télécharger et installer ».

Allez sur https://docs.microsoft.com/en-us/windows/iot-core/tutorials/quickstarter/PrototypeBoards, quels types de cartes électroniques sont utilisables avec Windows 10 IoT Core ? :

_____

Connectez physiquement votre Rasberry à un écran, à un câble ethernet et à un claivier et à une souris.
Connectez y votre carte SD et démarrez-la.

Démarrez votre système Windows IoT, il doit vous afficher l'adresse IP qu'il a récupéré en Dhcp.

Qu'affiche votre système ? :

_____

Avez-vous une interface utilisateur et des outils interactifs ? :

_____

Vous pouvez parcourir les outils mis à disposition. En cliquant sur « Paramètres » en bas à gauche, vous pouvez configurer la langue (Système) et les réseaux (ne configurez pas le Wifi). Vous avez aussi un « invite de commande » disponible.

Sur un autre poste, ouvrez un navigateur et connectez-vous à l'url : http://___.___.___.___:8080
(___.___.___.___ étant l'adresse Ip de votre Raspberry Pi)
Connecter vous avec le compte « adminsitrateur » et le mot de passe que vous avez saisi à la création de votre carte Windows IoT Standard.
Cette Url correspond à l'interface de configuration de votre Windows IoT.

Sur le tableau de bord de la première page, vous avez la possibilité de changer le nom ed la machine le mot de passe de l'administrateur, la « Time Zone » et la résolution de l'écran. Changer la « Time Zone » pour la bonne et testez les autres changements (pour le mot de passe attention aux claviers Qwerty/Azerty).

Allez sur « Apps/Apps Manager ». Vous voyez une liste d'applications.
A quoi doit correspondre l'application « IOTCoreDefaultApplication » ? :

_____

Allez sur « Apps/File Explorer ». Vous avez accés aux répertoires accessibles aux applications.
Dans quel sous-répertoires sont t-elles installées ? :

_____

Allez dans « Processes/Détails ». Que voyez-vous ? :

_____

Et enfin allez dans « Processes/Performance », que voyez-vous ? :

_____

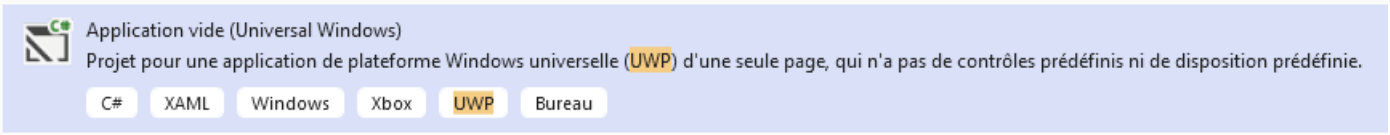
Allez dans « Processes /Run Command » pour accéder à l'interface PowerShell et tapez « ipconfig » et cliquez sur « Run ».
Tapez « dir », sur quel répertoire êtes-vous ? :

_____

Vous pouvez changer de nom de machine « SetComputerName », l'afficher avec « hostname », utiliser des commandes telles que ping, netstat, ipconfig, tracert, arp, faire des copies de fichier avec sfpcopy ou xcopy, arrêter ou redémarrer shutdown /s /t ou /r /t 0, …

2. – Création d'une application avec Visual Studio :

Démarrez Visual Studio et créez un nouveau projet C# / Application Vide (Universal Windows Plateform) (ou Blank Projet / Universal Windows Plateform).



Laissez par défaut les versions proposées par Visual Studio (au minimum la version build 17763).
Quelle est la particularité des Applications Universelles dites « UWP » ? :

_____

Il y a 2 fichiers relatifs à votre fenêtre principale. Quels sont-ils et quels types de code contiennent-ils ? :

_____

_____

Ouvrez votre page « MainPage.xaml » et ajoutez un bouton au centre de votre fenêtre avec un « TextBox » juste en dessous. Quel langage est utilisé pour décrire l'interface ? : _____

Vous pouvez remarquer que les contrôles Xaml ajoutés graphiquement sont en fait codés en script avec des propriétés définies dans balises en utilisant les syntaxes : *<balise …… /> ou <balise> … </balise>*
Exemples : *<TextBox … … … />* ou *<Grid> … … … </Grid>*

Que représente la balise *<Grid>* ? : _____

Pour bien répartir vos éléments graphiques créez 2 lignes juste après la balise « Grid » :

```
<Grid.RowDefinitions>
    <RowDefinition Height="1*" />
    <RowDefinition Height="1*" />
</Grid.RowDefinitions>
```

Ajoutez un bouton et un contrôle TextBox à votre fenêtre (DragDrop à partir de la barre d'outils des contrôle).
Le bouton est positionné dans la première ligne (ajoutez `Grid.Row="0"` dans sa balise), le TextBox dans la 2° (`Grid.Row="1"`), et pour les 2 les alignements vertical et horizontal à « `"Center"` ».
Pour avoir un affichage simple, modifiez vos éléments pour avoir :

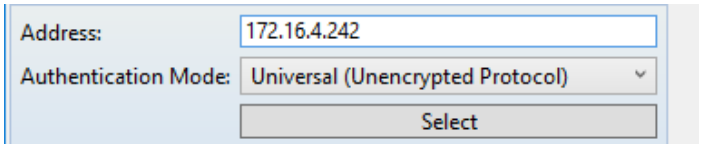
```
<Button Content="Button" Grid.Row="0" Height="109" Margin="0,0,0,0" VerticalAlignment="Top"
                Width="796" HorizontalAlignment="Center" Click="Button_Click"/>
<TextBox x:Name="tb1" Grid.Row="1" HorizontalAlignment="Center" Margin="0,0,0,0" Text="TextBox"
                TextWrapping="Wrap" VerticalAlignment="Top" Height="154" Width="674"/>
```

Double-cliquez sur votre bouton pour intercepter sa méthode « Click ». Dans celle-ci ajoutez le code :

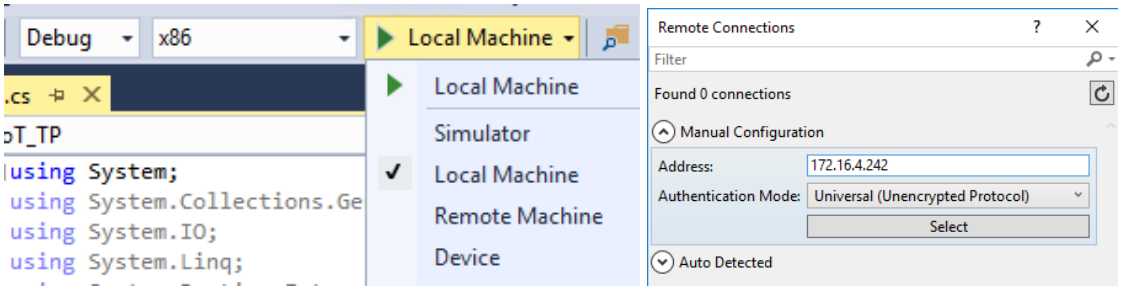
```
var deviceFamily = Windows.System.Profile.AnalyticsInfo.VersionInfo.DeviceFamily;
tb1.Text = deviceFamily + Environment.NewLine;
```

Lancez votre application en locale et vérifiez qu'elle fonctionne.

L'objectif d'une application UWP est de s'exécuter sur votre Os distant. Visual Studio permet une exécution locale mais surtout distante avec Débuggage. Pour cela, il faut configurer la cible en spécifiant l'IP de votre système distant. Déroulez la liste déroulante affichant « Local Machine » et sélectionnez « Remote Machine ». Dans la partie « Adress », saisissez l'IP de votre Rasberry et faites « Select ».



Enfin sélectionnez « ARM » comme type de processeur. 



Lancez votre application sur votre Rasberry et vérifiez qu'elle s'exécute bien.

Vous allez compléter votre programme pour qu'il puisse afficher l'adresse IP de votre PI. Pour cela vous allez ajouter la méthode « GetFirstLocalIp » à la classe de votre « MainPage » comme ci-dessous :

```
public static string GetFirstLocalIp(HostNameType hostNameType = HostNameType.Ipv4)
{
    var icp = NetworkInformation.GetInternetConnectionProfile();

    if (icp?.NetworkAdapter == null) return null;
    var hostname =
        NetworkInformation.GetHostNames()
            .FirstOrDefault(
                hn =>
                    hn.Type == hostNameType &&
                    hn.IPInformation?.NetworkAdapter != null &&
                    hn.IPInformation.NetworkAdapter.NetworkAdapterId == icp.NetworkAdapter.NetworkAdapterId);
    return hostname?.CanonicalName;
}
```

Appelez cette méthode dans la méthode Click de votre bouton et affichez l'adresse IP de votre PI.

Vous devez disposer d'une Rasberry Pi3 avec une carte micro SD Windows 10 IoT core, d'un poste Windows 10 avec Visual Studio (session 3iL normale). Installez Windows IoT sur votre carte SD et démarrez votre Raspberry PI.

3. - Se connecter à un Web Service WCF :

Un objet IoT doit être capable d'interagir avec des applications situées sur Internet notamment des services Web dans le Cloud. Pour le valider, vous allez ajouter un bouton qui se connecte à un Web service de météo.

Ce service, décris sur la page http://meteo.lab3il.fr/Service.aspx , renvoie les données météo de la station de 3iL. Pour lire la température, il suffit d'accéder à l'Url :

`http://www.meteowcfservice.lab3il.fr/meteo3il_2.svc`

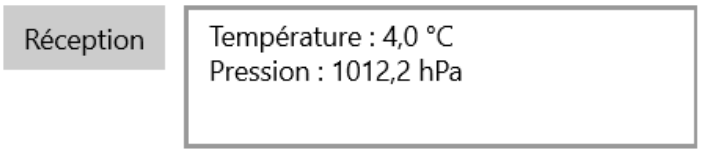
Le service renvoie un objet contenant les données météo courantes.
Ajoutez à votre fenêtre, un bouton « Réception » avec un « Textbox ». Double cliquez sur votre bouton pour intercepter l'évenement « Click ».
Dans l'explorateur de solution votre explorateur de projet, sélectionnez l'item « Référence » et avec le bouton droit de la souris faites « Ajouter une références de service ». Dans la fenêtre qui s'affiche, tapez l'Url du service puis cliquez sur le bouton « Allez à ». Le service doit répondre et vous afficher ses fonctionnalités. Dans « Espaces de noms », nommez votre service « WSMeteo » et faites OK.
Dans la méthode d'événement du bouton, il suffit d'instancier le service par :

```
WSMeteo.Service1Client sr = new WSMeteo.Service1Client();
```

Puis d'appeler la méthode « Get_MeteoData » en mode assynchrone :

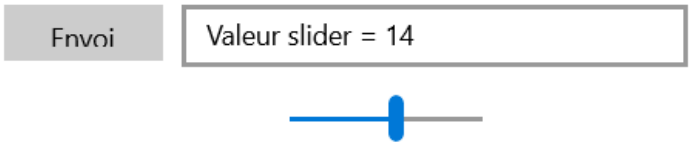
```
var val = await sr.Get_MeteoDataAsync();
```

Il faudra aussi préciser que la méthode d'événement du bouton est assynchrone : `private async void …`
Ensuite il suffit d'afficher dans le textbox (propriété Text) les valeurs de température et de pression qui sont disponibles dans la variable « val » : `val.d_Temp.ToString()`
En C#, une chaine de caractères s'initialise par des guillemets ("Température : "), vous pouvez concaténer des chaines de caractères avec l'opérateur « + », et passer à la ligne avec `Environment.NewLine;`
En testant votre programme sur le Windows IoT de la Raspberry vous devez avoir l'écran suivant :



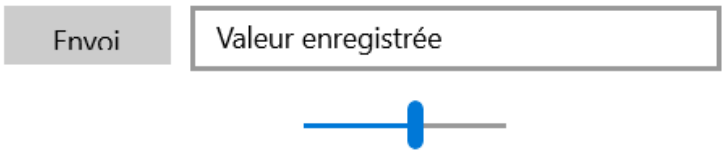
4. - Envoyer des informations à un Web Service :

Dans le monde de l'IoT, un object connecté doit pouvoir envoyer des informations à un Web Service.
Ajoutez une 3° ligne à votre Grid, et dans celle-ci, un bouton, un contrôle « TextBox » et un contrôle « Slider ». Nommez-les chacun avec un nom spécifique (fenêtre Propriétés ou en tapant Name dans les balises). Fixez les valeurs minimale et maximale du contrôle Slider (fenêtre Propriétés). Double cliquez sur le contrôle Slider pour intercepter sa méthode « `ValueChanged` ». Dans cette méthode, affichez la valeur du Slider dans le contrôle « TextBox » en utilisant : `mon_slider.Value.ToString()`



Testez votre programme.

Dans l'explorateur de solution votre explorateur de projet, faites « Ajouter une références de service » et ajoutez le service : http://iot2.lab3il.fr/Service1.svc
Appelez la « `WSIOT` », instanciez ce service comme pour le précédent.
Ce service vous permet d'enregistrer des données sur le site : http://iot2.lab3il.fr/
Pour cela, il vous faut lui passer une chaine de caractères au format : ##identifiant##valeur
Vous devez donc créer une chaine de caractère (`String sVal = "##votre_identifiant##"`) avec l'identifiant que vous voulez, et ajoutez la valeur du contrôle Slider (`+ mon_slider.Value.ToString()`). Ensuite il faut appeler la méthode « `SendDataIoTAsync` » avec cette chaine en paramètre. Cette méthode assynchrone renvoie la valeur 1 si vos données ont été enregistrées sinon un message d'erreur.



Testez votre programme, vos données doivent apparaître sur le site http://iot2.lab3il.fr.

5. - Se connecter à un Web Service REST :

Un objet IoT doit être capable d'interagir tout type de services Web dans le Cloud, notamment les services de type REST.
Un des service décris sur la page http://meteo.lab3il.fr/Service.aspx , et de type REST et renvoie les données météo de la station de 3iL. Pour lire la température, il suffit d'accéder à l'Url :

`http://www.meteorestservice.lab3il.fr/ServiceRest.svc/meteo/1`

qui renvoie simplement la dernière température mesurée.

Créez un nouveau bouton « Température Exterieure » et interceptez sa méthode « Click ». Dans le code de cette méthode ajoutez le code suivant :

```
Windows.Web.Http.HttpClient httpClient = new Windows.Web.Http.HttpClient();
var headers = httpClient.DefaultRequestHeaders;
string header = "ie";
if (!headers.UserAgent.TryParseAdd(header))
```

```
{
    throw new Exception("Invalid header value: " + header);
}
header = "Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0)";
if (!headers.UserAgent.TryParseAdd(header))
{
    throw new Exception("Invalid header value: " + header);
}
Uri requestUri = new Uri(" …… url du service ……");
Windows.Web.Http.HttpResponseMessage httpResponse = new
                                                    Windows.Web.Http.HttpResponseMessage();
string httpBody = "";
try
{
    httpResponse = await httpClient.GetAsync(requestUri);
    httpResponse.EnsureSuccessStatusCode();
    httpBody = await httpResponse.Content.ReadAsStringAsync();
    httpBody = httpBody.Substring(httpBody.IndexOf(">") + 1,
                              httpBody.LastIndexOf("<") - httpBody.IndexOf(">") -1);

    // Affichage de la température

}
catch (Exception ex)
{
    httpBody = "Error: " + ex.Message;
}
```

Tester votre bouton en affichant la temperature dans la TextBox.