TP n°3 : Utilisation d'un capteur I2C sous Raspberry Pi

CORRECTION

3. - Lecture des données sur le capteur :

Regardez sur Internet la pression du jour, la valeur correspond t-elle et pourquoi ? :
La pression est inférieur car il faut faire une correction d'altitude et la ramener à 0

Ajoutez une correction votre pression pour la ramener au niveau de la mer sachant que la pression dimimue d'un 1 hPa tous les 8 m et que l'altitude de 3iL est d'environ 270 m.
Testez votre application et vérifiez que la pression est correcte.

4. - Exportation des données vers la plateforme :

Affichez les données lues et la valeur renvoyée par la méthode « `GoToThingSpeak` ».

Que représente cette valeur ? : *Le nombre totale de données reçu par votre channel*

CODE :

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Runtime.InteropServices.WindowsRuntime;
using System.Text;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.System.Threading;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;


namespace TPW2
{
    /// <summary>
    public sealed partial class MainPage : Page
    {
        private const string _url = "http://api.thingspeak.com/";
        private const string _APIKey = "____VOTRE_CLE_THINGSPEAK____";
        BMP280 bmp_280;
        bool b_initialize = false;
        public MainPage()
        {
            this.InitializeComponent();
            bmp_280 = new BMP280();
        }

        private async void bt1_Click(object sender, RoutedEventArgs e)
        {
            // Initialize ne doit être appelée qu'une seule fois
            if (!b_initialize) { await bmp_280.Initialize(); }
            b_initialize = true;

            float fTemp = await bmp_280.ReadTemperature();
            tb1.Text = "Température lue : " + fTemp.ToString("0.00") + " °C" +
                                                                Environment.NewLine;
            float fPres = await bmp_280.ReadPreasure();
            fPres = fPres / 100; // Pression de Pa à hPa
            fPres = fPres + 270 / 8; // Correction pression due à l'altitude
            tb1.Text += "Pression lue : " + fPres.ToString("0.0") + " hPa" + Environment.NewLine;
            StringBuilder sb = new StringBuilder();
            sb.Append(_url);
            sb.Append("update?key=");
            sb.Append(_APIKey);
            sb.Append("&field1=" + fTemp.ToString("0.0"));
            sb.Append("&field2=" + fPres.ToString("0.0"));
            String sReturn = GoToThingSpeak(sb.ToString());
            tb1.Text += "Réponse Thingspeak : " + sReturn; // Affichage valeur retournée par TS

        }

        private static string GoToThingSpeak(string QueryString)
        {
            StringBuilder sbResponse = new StringBuilder();
            byte[] buf = new byte[8192];
            HttpWebRequest myRequest = (HttpWebRequest)WebRequest.Create(QueryString);
            HttpWebResponse webResponse = (HttpWebResponse)myRequest.GetResponse();
```

```
            try
            {
                Stream myResponse = webResponse.GetResponseStream();
                int count = 0;
                do
                {
                    count = myResponse.Read(buf, 0, buf.Length);
                    if (count != 0)
                    {
                        sbResponse.Append(Encoding.ASCII.GetString(buf, 0, count));
                    }
                }
                while (count > 0);
                return sbResponse.ToString();
            }
            catch (WebException ex)
            {
                return "0";
            }
        }

        private async void bt2_Click(object sender, RoutedEventArgs e)
        { // Timer
             // Initialize ne doit être appelée qu'une seule fois
            if (!b_initialize) { await bmp_280.Initialize(); }
            b_initialize = true;
            TimeSpan tPeriod = TimeSpan.FromSeconds(30);
            ThreadPoolTimer tPeriodicTimer = ThreadPoolTimer.CreatePeriodicTimer(
                (source) =>
                {
                    SendData();
                }, tPeriod
                );
        }

        private async void SendData()
        {
            float fTemp = await bmp_280.ReadTemperature();
            float fPres = await bmp_280.ReadPreasure();
            StringBuilder sb = new StringBuilder();
            sb.Append(_url);
            sb.Append("update?key=");
            sb.Append(_APIKey);
            sb.Append("&field1=" + fTemp.ToString("0.0"));
            sb.Append("&field2=" + fPres.ToString("0.0"));
            String sReturn = GoToThingSpeak(sb.ToString());
        }
    }
}
```