# CODE SOURCE

## Sommaire

# 1. Thread.cs

```
using System;
using Microsoft.SPOT;
using System.Threading;



namespace Acquisition
{
    public class Program
    {
        public static void Main()
        {
            //Point d'entrée du programme
            C_Traitement aquisition = new C_Traitement();
            Thread newThread = new Thread(() => aquisition.Traitement());
//Initialiser l'objet thread en passant la méthode à éxécuter  Traitement()
            newThread.Start(); //Exécution du thread
        }

    }
}
```

## 2. C_Traitement

```
using System;
using Microsoft.SPOT;
using System.Threading;
using System.Text;
using System.IO;
using System.IO.Ports;
using Microsoft.SPOT.Hardware;



namespace Acquisition
{
    class C_Traitement
    {

        //Emetteur
        //Création d'un port série pour la communication avec le module
XBEE.
        SerialPort xbee_E = new SerialPort("COM1", 9600, Parity.None, 8,
StopBits.One);


        public void Traitement()
        {


            C_Panda_I2C I2C = new C_Panda_I2C();



            double temp;
            double masse;
            //DateTime DT;
            double tensionAlim;


            string temp_s;
            string masse_s;
            string tensionAlim_s;

            string data;
            byte[] wdata = new byte[1];

            bool i = true;
           // int j = 0;
```

```
            xbee_E.Open();
            if (xbee_E.IsOpen == true)
            {
                Debug.Print("Module sans fil connecté !");
            }




            while (i == true)
            {

                Debug.Print("En cours d'acquisition ...");


                temp = I2C.AcquisitionTemperature();
                // poids = I2C.correction();
                //masse = 0.0;
                masse = I2C.AquisitionPoids();
                //DT = I2C.read_DateTime();
                tensionAlim = I2C.AquisitionTensionAlim();

              //  DT = DateTime.Now;
               // Debug.Print("Nous sommes le " + DT.ToString());


                masse_s = masse.ToString();
                if (masse < 10) { masse_s = "0" + masse_s; }
                if (masse_s.Length == 2) { masse_s = masse_s + ".0"; }
                masse_s = masse_s.Substring(0, 4);
                Debug.Print("Le poids est de " + masse_s + "Kg.");


                temp_s = temp.ToString();
                if (temp < 10) { temp_s = "0" + temp_s; }
                if (temp_s.Length == 2) { temp_s = temp_s + ".0"; }
                temp_s = temp_s.Substring(0, 4);
                Debug.Print("La temperature est de " + temp_s + " C.");


                tensionAlim_s = tensionAlim.ToString();
                //if (tensionAlim < 10) { tensionAlim_s = "0" +
tensionAlim_s; }
                //if (tensionAlim_s.Length == 2) { tensionAlim_s =
tensionAlim_s + ".0"; }
                tensionAlim_s = tensionAlim_s.Substring(0, 3);
                Debug.Print("La tension d'alim est de " + tensionAlim_s +
"V.");
```

```
                data = masse_s + " " + temp_s + " " + tensionAlim_s;//des
données sous forme de chaine
                wdata = Encoding.UTF8.GetBytes(data);//encodage des données
en byte
                xbee_E.Write(wdata, 0, wdata.Length);



                Thread.Sleep(3000); // Toutes les 3sec une acquisition se
fait.

            }
        }
    }
}
```

# 3. C_Panda_I2C

```
using System;
using Microsoft.SPOT.Hardware;



namespace Acquisition
{

    class C_Panda_I2C
    {

        //Création d'objet I2c
        I2CDevice.Configuration Temperature = new
I2CDevice.Configuration(0x4D, 400); //DS1621
        I2CDevice.Configuration Poids = new I2CDevice.Configuration(0x49,
100); //PCF8591P
        I2CDevice MyI2C;
        I2CDevice.I2CTransaction[] xActions; //Création d'opérations
        //Représente une instance de l'interface I2c pour un dispositif I2c

        public C_Panda_I2C() //Constructeur
        {
            MyI2C = new I2CDevice(Poids);
        }

        public void Dispose() //Libère les ressources utilisées par l'objet
de I2CDevice
        {
            MyI2C.Dispose();
            xActions = null;
        }

        public double AcquisitionTemperature()
        {
            byte[] temp = new byte[2]; //Besoin de 2 octets
            double realtemp = 0.0; //15 chiffres de précision(virgule
flottante)

            MyI2C.Config = Temperature;
            xActions = new I2CDevice.I2CTransaction[3]; //Création de 3
transactions
```

```
            byte[] commande1 = new byte[1] { 238 }; //Commande EE(Début de
conversion)
            byte[] commande2 = new byte[1] { 170 }; //Commande AA(Lecture
de la température)

            //Transactions
            xActions[0] = I2CDevice.CreateWriteTransaction(commande1);
//Représente une transaction I2c qui écrit dans le dispositif adressé
            xActions[1] = I2CDevice.CreateWriteTransaction(commande2);
            xActions[2] = I2CDevice.CreateReadTransaction(temp);
//Représente une transaction I2c qui lit le dispositif adressé

            MyI2C.Execute(xActions, 1000); //Accès au bus I2c avec un délai
de 1sec

            realtemp = temp[0];
            if (temp[1] != 0) //(opérateur d'inégalité)
                realtemp = realtemp + 0.5;

            return realtemp;

        }

        public double AquisitionPoids()
        {

            byte[] poids = new byte[1];
            double masse = 0.0;

            MyI2C.Config = Poids;
            xActions = new I2CDevice.I2CTransaction[2]; //Création
transactions

            byte[] commande = new byte[1] { 1 }; //Mettre 1 pour lire la
tension d'alim sur AIN1

            xActions[0] = I2CDevice.CreateWriteTransaction(commande);
            xActions[1] = I2CDevice.CreateReadTransaction(poids);

            MyI2C.Execute(xActions, 1000);

            masse = (( 4.7 * poids[0]) / 255) ; //Calcul d'acquisition du
poids
            return masse;

        }

        public double AquisitionTensionAlim() // Lecture de la
tension(Convertisseur Analogique/Numérique)
        {

            double tensionalims = 0;
            MyI2C.Config = Poids;
            xActions = new I2CDevice.I2CTransaction[2];
```

```
            byte[] commande = new byte[1] { 0x00 }; //Mettre 0 pour lire la
tension d'alim sur AIN0
            xActions[0] = I2CDevice.CreateWriteTransaction(commande);

            byte[] alims = new byte[1];
            xActions[1] = I2CDevice.CreateReadTransaction(alims);

            MyI2C.Execute(xActions, 1000);

            tensionalims = ( 4.7 * alims[0]) / 255; //Calcul d'acquistion
de la tension
            return tensionalims;

        }

        //public DateTime read_DateTime()
        //{
        //    MyI2C.Config = DateTime;
        //    xActions = new I2CDevice.I2CTransaction[2];
        //    xActions[0] = I2CDevice.CreateWriteTransaction(new byte[] {
0x00 });
        //    byte[] ReturnedDateTime = new byte[7];
        //    xActions[1] =
I2CDevice.CreateReadTransaction(ReturnedDateTime);
        //    if (MyI2C.Execute(xActions, 1000) == 0) new Exception("Failed
to send I2C data");

        //    int sec = bcdToDec(ReturnedDateTime[0]) & 0x7f;
        //    int min = bcdToDec(ReturnedDateTime[1]);
        //    int hour = bcdToDec(ReturnedDateTime[2]) & 0x3f;
        //    int dayofweek = bcdToDec(ReturnedDateTime[3]);
        //    int dayofmonth = bcdToDec(ReturnedDateTime[4]);
        //    int month = bcdToDec(ReturnedDateTime[5]);
        //    int year = bcdToDec(ReturnedDateTime[6]) + 2000;

        //    DateTime dt = new DateTime(year, month, dayofmonth, hour,
min, sec);
        //    return dt;
        //}

        //public void set_DateTime(DateTime datetime)
        //{
        //    MyI2C.Config = DateTime;
        //    xActions = new I2CDevice.I2CWriteTransaction[1];
        //    byte[] sb = new byte[8] {   0x00,
        //                               decToBcd(datetime.Second),
        //                               decToBcd(datetime.Minute),
        //                               decToBcd(datetime.Hour),
        //
decToBcd((int)datetime.DayOfWeek),
        //                               decToBcd(datetime.Day),
        //                               decToBcd(datetime.Month),
        //                               decToBcd(datetime.Year - 2000)
```

```
        //                              };

        //    xActions[0] = I2CDevice.CreateWriteTransaction(sb);
        //    if (MyI2C.Execute(xActions, 1000) == 0) new Exception("Failed
to send I2C data");
        //}

        public byte decToBcd(int val)
        {
            return (byte)((val / 10 * 16) + (val % 10));
        }

        public byte bcdToDec(byte val)
        {
            return (byte)((val / 16 * 10) + (val % 16));
        }



    }
}
```