|  |
| --- |
| ECAM 2012 -2013 |
| Bureau d’étude informatique |
| Application domotique |
| **Belhouari Khaled**  **Timami Eric**  **Leruth Corentin** |
|  |
|  |



Institut supérieur industriel – Haute école Léonard de Vinci 2012 – 2013

Table des matières

[1. Introduction 3](#_Toc356332294)

[2. Interface logiciel 4](#_Toc356332295)

[3. Diagrammes UML 7](#_Toc356332296)

[3.1. Cas d’utilisation 7](#_Toc356332297)

[3.1.1. Interface logiciel domotique 7](#_Toc356332298)

[3.1.2. Système 8](#_Toc356332299)

[3.2. Classes 9](#_Toc356332300)

[4. Base de données 10](#_Toc356332301)

[5. Routines 11](#_Toc356332302)

[5.1. Base de données 11](#_Toc356332303)

[5.1.1. Connexion 11](#_Toc356332304)

[5.1.2. Ajout de consignes 12](#_Toc356332305)

[5.1.3. Afficher logs 13](#_Toc356332306)

[5.2. Régulation 14](#_Toc356332307)

[5.3. Import Excell 20](#_Toc356332308)

# Introduction

Dans le cadre du cours d’informatique de troisième année de Bachelier en orientation génie électrique, l’ECAM propose aux étudiants de réaliser un projet domotique au cours du second semestre.

Pour se faire, nous disposons d’une maquette représentant une maison comportant trois pièces, à savoir : la cuisine, la chambre et pour finir le salon. Les deux premières disposent chacune d’une fenêtre, une chambre ainsi qu’un chauffage. Le salon, lui, possède deux portes : celle de la chambre et celle de la cuisine permettant les communications entres les trois pièces. Pour finir, il comprend également une porte d’entrée de même qu’un chauffage.

Notre programme doit être capable de réguler la température à l’intérieur des différentes pièces selon des consignes définies par l’utilisateur.

Celui-ci doit pouvoir fonctionner selon deux modes :

* Réel
* Virtuel

Le mode réel doit être capable communiquer avec la maquette de la maison afin de pouvoirs allumer ou éteindre le chauffage ainsi que de récupérer l’état des ouvertures (portes et fenêtres). Pour ce faire, nous devrons réaliser une carte électronique nous autorisant les transmissions de données entre la maison et notre ordinateur.

Le mode virtuel, quant à lui, a pour but de réaliser l’interface d’une maison virtuelle tout en étant capable de réguler la température des différentes pièces. Il sera donc nécessaire d’effectuer des calculs de thermodynamiques.

Nous vous détaillerons, tout au long de ce rapport, les différents éléments et phases nous ayant permis la finalisation de notre solution :

* Interface maison virtuelle
* POO (diagrammes use-case et classes)
* Base de données
* Routines principales (accès BDD, régulation, …)
* Carte électronique
* Communication DAQ

# Interface logiciel

Notre interface est divisée en trois onglets principaux :

* L’onglet « Vue Principale » nous donne la représentation de notre maison virtuelle. Nous avons représenté les radiateurs via des boutons et les ouvertures avec des labelles. Ceux-ci deviennent rouges lorsqu’ils sont éteint/fermé ou vert pour allumé/ouvert. Chaque pièce est accompagnée d’un panel contenant : la température actuelle, la température souhaitée (consigne) ainsi que l’état du chauffage. Une « trackbar » apporte l’opportunité d’avancé le temps. Il y a également 2 « RadioButton » permettant de choisir le mode (virtuel/réel).
* L’onglet « Consignes » donne la possibilité d’ajouter des consignes à chacune des pièces. Ceci en spécifiant le jour, l’heure et la température.
* Pour finir, l’onglet « Logs » nous informe des changements d’état subit par chaque ouverture. La date et l’heure de ces changements nous sont indiquées.



Figure 1: Interface "Vue Principale"

1. Indique l’heure actuelle
2. Permet de faire avancer l’heure (uniquement mode virtuel)
3. Indique l’état de la lampe extérieur (blanc/jaune – éteint/allumée)
4. Indique l’état de l’ouverture (rouge/vert – fermé/ouvert)
5. Indique l’état du chauffage (rouge/vert éteint/allumé)
6. Permet de passer d’un mode à un autre
7. Indique le mode actuel

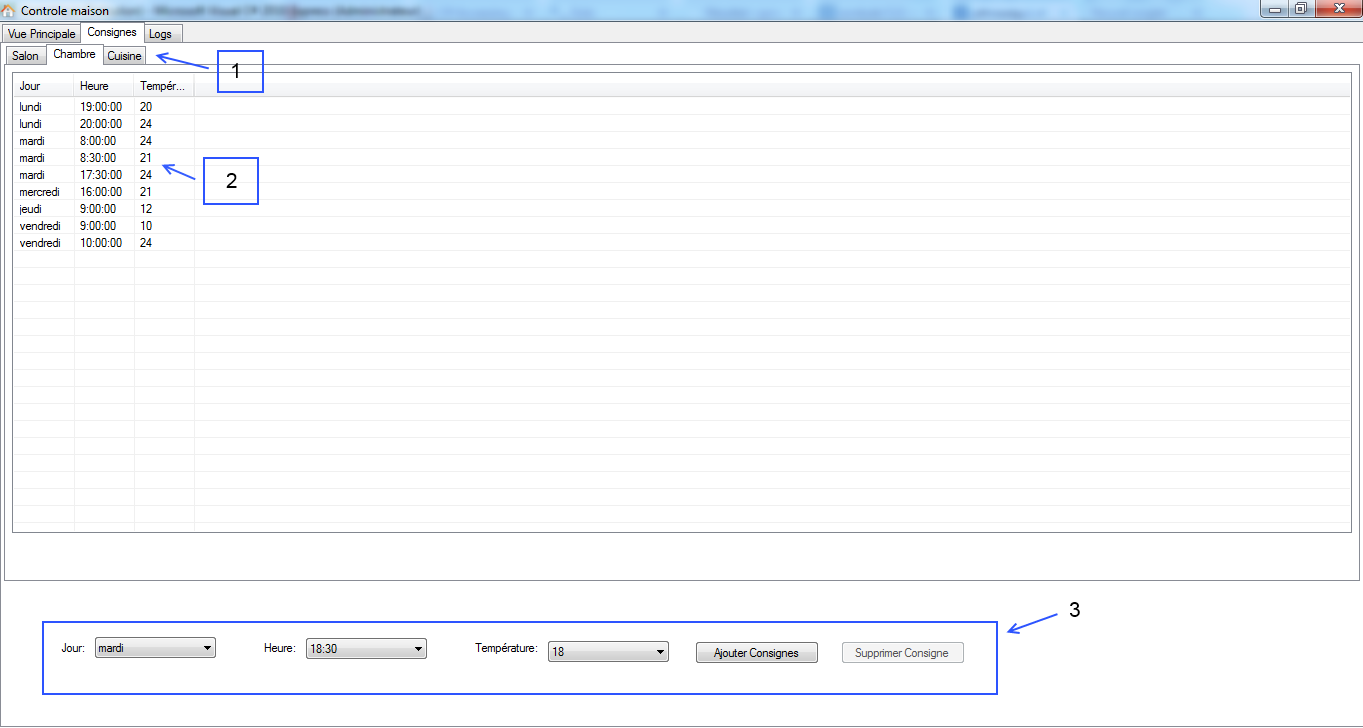


Figure 2: Interface "Consignes"

1. Indique la pièce dans laquelle on veut ajouter une consigne
2. Affiche la liste des consignes de la pièce correspondante
3. Permet d’ajouter une consigne

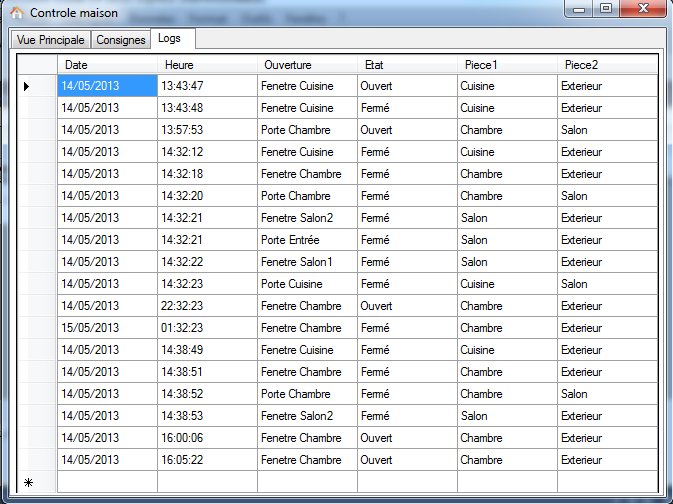


Figure 3: Interface "Logs"

# Diagrammes UML

## Cas d’utilisation

### Interface logiciel domotique



Figure 4: Cas d’utilisation interface logiciel domotique

### Système



Figure 5: Cas d'utilisation système

## Classes



Figure 6: Diagramme classes

# Base de données



Figure 7: Schéma base de données

# Routines

## Base de données

### Connexion

La première étape consiste à se connecter à la base de données. Pour ce faire, il est indispensable d’ajouter les références suivantes :

using System.Data;

using System.Data.SqlClient;

Ensuite, pour s’y connecter :

private SqlConnection con; //pour la connectio

private string strcon; //pour la chaine de connection

private Boolean statusBDD; // Status de connection BDD

strcon = @"Data Source=TATCHI-PC\SQLEXPRESS;Initial Catalog=domotique;Integrated Security=True;Pooling=False;Connection Timeout=2";

con = new SqlConnection(strcon);

try

{

con.Open(); // Connection à la BDD

statusBDD = true;

}

catch (Exception ex)

{

statusBDD = false;

var result = MessageBox.Show("Erreur de connexion à la base de donnée. Voulez-vous quand même continuer?", "Erreur connecion BDD!",

MessageBoxButtons.YesNo,

MessageBoxIcon.Error);

if (result == DialogResult.No)

{

Application.Exit();

}

else

{

setStatusBDD(false);

}

}

### Ajout de consignes

private void buttonAjouterConsigne\_Click(object sender, EventArgs e)

{

Boolean existeDeja = false; // Permet de savoir si consigne existe deja

DateTime date;

String dt;

date = Convert.ToDateTime(comboBoxHeure.SelectedItem.ToString());

dt = String.Format("{0:HHmmss}", date);

TConsigne maConsigne = new TConsigne(Convert.ToInt32(dt), comboBoxJour.SelectedItem.ToString(), Convert.ToInt32(comboBoxTemperature.SelectedItem.ToString()));

foreach (TConsigne cons in maMaison.ListePieces[tabControlPieces.SelectedIndex].ListeConsignes)

{

if ((maConsigne.Jour == cons.Jour) && (maConsigne.Heure == cons.Heure)) // Test si consigne existe deja

{

existeDeja = true;

var result = MessageBox.Show("Une consigne existe deja à cette date la. Voulez-vous l'écraser?", "Erreur!",

MessageBoxButtons.YesNo,

MessageBoxIcon.Error);

if (result == DialogResult.Yes)

{

if (statusBDD == true) // Si connexion ok on delete et ajoute la nouvelle

{

String maRequete;

maRequete = "DELETE FROM domotique.dbo.Consigne WHERE jour = '" + cons.Jour + "' and heure = '" + cons.Heure + "' and temperature = '" + cons.Temperature + "'";

SqlCommand myCommand = new SqlCommand(maRequete, con);

myCommand.ExecuteNonQuery();

String maRequete2;

int piece = tabControlPieces.SelectedIndex + 4;

maRequete2 = "INSERT INTO domotique.dbo.Consigne VALUES ('" + maConsigne.Jour + "' , '" + maConsigne.Heure + "', '" + maConsigne.Temperature + "', '" + piece + "')";

SqlCommand myCommand2 = new SqlCommand(maRequete2, con);

myCommand2.ExecuteNonQuery();

}

maMaison.ListePieces[tabControlPieces.SelectedIndex].SupprimerConsigne2(cons);

maMaison.ListePieces[tabControlPieces.SelectedIndex].AjouterConsigne(maConsigne);

break;

}

}

}

if (existeDeja == false) // Si elle n'existais pas deja

{

if (statusBDD == true) // Si connexion BDD OK on ajoute

{

String maRequete, maRequete2;

SqlDataReader DataReader;

maRequete2 = "SELECT PK\_Piece FROM domotique.dbo.Piece where nom = '" + maMaison.ListePieces[tabControlPieces.SelectedIndex].Nom + "'";

SqlCommand myCommand = new SqlCommand(maRequete2, con);

DataReader = myCommand.ExecuteReader();

DataReader.Read();

int idPiece = DataReader.GetInt32(0);

DataReader.Close();

maRequete = "INSERT INTO domotique.dbo.Consigne VALUES ('" + maConsigne.Jour + "' , '" + maConsigne.Heure + "', '" + maConsigne.Temperature + "', '" + idPiece + "')";

SqlCommand myCommand2 = new SqlCommand(maRequete, con);

myCommand2.ExecuteNonQuery();

maMaison.ListePieces[tabControlPieces.SelectedIndex].AjouterConsigne(maConsigne);

}

else

{

maMaison.ListePieces[tabControlPieces.SelectedIndex].AjouterConsigne(maConsigne);

}

}

maMaison.ListePieces[tabControlPieces.SelectedIndex].TrierListeConsignes(); // On trie la liste de consigne

AfficherConsignes(); // On les affiches

}

### Afficher logs

private void tabPageLogs\_Enter(object sender, EventArgs e)

{

String maRequete;

SqlDataReader DataReader;

int mode = 0;

if (this.maMaison.Virtuel == true) // Test mode de la maison

{

mode = 1;

}

else

{

mode = 2;

}

maRequete = "SELECT PK\_LogPiece, date, etat, Ouverture.nom, Piece.nom, pp.nom from domotique.dbo.LogPiece INNER JOIN domotique.dbo.Ouverture ON FK\_Ouverture = PK\_Ouverture INNER JOIN domotique.dbo.Piece ON FK\_Piece1 = PK\_Piece INNER JOIN domotique.dbo.Piece as pp ON FK\_Piece2 = pp.PK\_Piece where FK\_Mode = '" + mode + "'";

SqlCommand myCommand = new SqlCommand(maRequete, con);

DataReader = myCommand.ExecuteReader();

DateTime date;

String dt, heure, etat;

int ouvert;

dataGridViewLogPiece.Rows.Clear(); // On efface la tableau

while (DataReader.Read()) // On charge le tableau

{

date = DataReader.GetDateTime(1);

ouvert = Convert.ToInt32(DataReader.GetValue(2));

heure = String.Format("{0:HH:mm:ss}", date);

dt = String.Format("{0:d/MM/yyyy}", date);

if (ouvert == 1)

{

etat = "Ouvert";

}

else

{

etat = "Fermé";

}

dataGridViewLogPiece.Rows.Add(dt, heure, DataReader.GetValue(3).ToString(), etat, DataReader.GetValue(4), DataReader.GetValue(5));

}

DataReader.Close();

}

## Régulation

Boolean msgError = false;

public void gerer()

{

if (this.Virtuel == false) // On regarde si on se trouve en mode reel

{

if (this.ULStat.Value.ToString() != "NoErrors") // On test la connexion du DAQ

{

if (msgError == false)

{

msgError = true;

var result = MessageBox.Show("DAQ non connecté. Voulez-vous repasser en mode virtuel?", "Erreur connecion DAQ!",

MessageBoxButtons.YesNo,

MessageBoxIcon.Error);

if (result == DialogResult.Yes)

{

this.Virtuel = true;

msgError = false;

}

else

{

msgError = false;

Application.Restart();

}

}

}

else

{

float ch0;

float ch1;

float ch2;

float ch3;

System.UInt16 DataValue0;

System.UInt16 DataValue1;

System.UInt16 DataValue2;

System.UInt16 DataValue3;

int chan0 = 0;

int chan1 = 1;

int chan2 = 2;

int chan3 = 3;

MccDaq.Range Range;

Range = Range.Bip10Volts;

// Acquisition des température

this.ULStat = this.Board.AIn(chan0, Range, out DataValue0); //acquisition

this.ULStat = this.Board.AIn(chan1, Range, out DataValue1); //acquisition

this.ULStat = this.Board.AIn(chan2, Range, out DataValue2); //acquisition

this.ULStat = this.Board.AIn(chan3, Range, out DataValue3); //acquisition

// Conversion de celles-ci

this.ULStat = this.Board.ToEngUnits(Range, DataValue0, out ch0); //conversion

this.ULStat = this.Board.ToEngUnits(Range, DataValue1, out ch1); //conversion

this.ULStat = this.Board.ToEngUnits(Range, DataValue2, out ch2); //conversion

this.ULStat = this.Board.ToEngUnits(Range, DataValue3, out ch3); //conversion

// Initialisation températures

this.ListePieces[0].Temperature = Convert.ToDouble(ch1) \* 10;

this.ListePieces[1].Temperature = Convert.ToDouble(ch0) \* 10;

this.ListePieces[2].Temperature = Convert.ToDouble(ch3) \* 10;

this.TemperatureExt = Convert.ToDouble(ch2) \* 10;

// Acquisition ouvertures

short D1;

this.ULStat = this.Board.DIn(MccDaq.DigitalPortType.FirstPortB, out D1);

String valeur = null;

for (int i = 0; i <= 7; i++)

{

if ((D1 & (1 << i)) != 0)

valeur = valeur + "1";

else

valeur = valeur + "0";

}

// Initialisation des ouvertures

this.ListePieces[0].ListeParois[0].ListeOuvertures[0].Ouvert = Convert.ToBoolean(Convert.ToInt32(valeur[7].ToString()));

this.ListePieces[2].ListeParois[1].ListeOuvertures[0].Ouvert = Convert.ToBoolean(Convert.ToInt32(valeur[7].ToString()));

this.ListePieces[0].ListeParois[1].ListeOuvertures[0].Ouvert = Convert.ToBoolean(Convert.ToInt32(valeur[6].ToString()));

this.ListePieces[1].ListeParois[1].ListeOuvertures[0].Ouvert = Convert.ToBoolean(Convert.ToInt32(valeur[6].ToString()));

this.ListePieces[0].ListeParois[2].ListeOuvertures[2].Ouvert = Convert.ToBoolean(Convert.ToInt32(valeur[1].ToString()));

this.ListePieces[0].ListeParois[2].ListeOuvertures[1].Ouvert = Convert.ToBoolean(Convert.ToInt32(valeur[4].ToString()));

this.ListePieces[0].ListeParois[2].ListeOuvertures[0].Ouvert = Convert.ToBoolean(Convert.ToInt32(valeur[2].ToString()));

this.ListePieces[1].ListeParois[0].ListeOuvertures[0].Ouvert = Convert.ToBoolean(Convert.ToInt32(valeur[3].ToString()));

this.ListePieces[2].ListeParois[0].ListeOuvertures[0].Ouvert = Convert.ToBoolean(Convert.ToInt32(valeur[5].ToString()));

foreach (TPiece piece in this.ListePieces)

{

if ((piece.OuvertureExtOuverte() || (piece.OuvertureCommuneOuvert() != null && piece.OuvertureExtPieceVoisine() != null))) // Si ouverture ext ouverte ou commune + ext pièce voisine

{

piece.Radiateur = false;

if (piece.Nom == "Salon")

{

int pinChauffageSalon = 1; // 1 en sortie (pin 22)

this.ULStat = this.Board.DBitOut(MccDaq.DigitalPortType.FirstPortA, pinChauffageSalon, MccDaq.DigitalLogicState.Low);

}

else if (piece.Nom == "Chambre")

{

int pinChauffageChambre = 0; // 3 en sortie (pin 24)

this.ULStat = this.Board.DBitOut(MccDaq.DigitalPortType.FirstPortA, pinChauffageChambre, MccDaq.DigitalLogicState.Low);

}

else if (piece.Nom == "Cuisine")

{

int pinChauffageCuisine = 3; // 0 en sortie (pin 21)

this.ULStat = this.Board.DBitOut(MccDaq.DigitalPortType.FirstPortA, pinChauffageCuisine, MccDaq.DigitalLogicState.Low);

}

}

else // Sinon teste de consignes

{

if (piece.Consigne > piece.Temperature) // Si consigne plus grand -> allume chauffage

{

piece.Radiateur = true;

if (piece.Nom == "Salon")

{

int pinChauffageSalon = 1; // 1 en sortie (pin 22)

this.ULStat = this.Board.DBitOut(MccDaq.DigitalPortType.FirstPortA, pinChauffageSalon, MccDaq.DigitalLogicState.High);

}

else if (piece.Nom == "Chambre")

{

int pinChauffageChambre = 0; // 3 en sortie (pin 24)

this.ULStat = this.Board.DBitOut(MccDaq.DigitalPortType.FirstPortA, pinChauffageChambre, MccDaq.DigitalLogicState.High);

}

else if (piece.Nom == "Cuisine")

{

int pinChauffageCuisine = 3; // 0 en sortie (pin 21)

this.ULStat = this.Board.DBitOut(MccDaq.DigitalPortType.FirstPortA, pinChauffageCuisine, MccDaq.DigitalLogicState.High);

}

}

else // Sinon étaint chauffage

{

piece.Radiateur = false;

if (piece.Nom == "Salon")

{

int pinChauffageSalon = 1; // 1 en sortie (pin 22)

this.ULStat = this.Board.DBitOut(MccDaq.DigitalPortType.FirstPortA, pinChauffageSalon, MccDaq.DigitalLogicState.Low);

}

else if (piece.Nom == "Chambre")

{

int pinChauffageChambre = 0; // 3 en sortie (pin 24)

this.ULStat = this.Board.DBitOut(MccDaq.DigitalPortType.FirstPortA, pinChauffageChambre, MccDaq.DigitalLogicState.Low);

}

else if (piece.Nom == "Cuisine")

{

int pinChauffageCuisine = 3; // 0 en sortie (pin 21)

this.ULStat = this.Board.DBitOut(MccDaq.DigitalPortType.FirstPortA, pinChauffageCuisine, MccDaq.DigitalLogicState.Low);

}

}

}

}

}

}

else // Si mode virtuel

{

foreach (TPiece piece in this.ListePieces)

{

if (piece.OuvertureExtOuverte()) // Si ouvertureExt ouverte

{

piece.Radiateur = false; // Coupe chauffage

piece.Temperature = this.temperatureExt; // Température pièce = température extérieur

}

else

{

TParoi paroiCommune = piece.OuvertureCommuneOuvert();

if ((paroiCommune != null)) // Si paroi commune ouverte

{

TPiece pieceVoisine = piece.OuvertureExtPieceVoisine();

if (piece.OuvertureExtPieceVoisine() != null) // Si pièce à une pièce voisine dont ouvertureEXt ouvert

{

if (pieceVoisine == paroiCommune.Piece1 || pieceVoisine == paroiCommune.Piece2) // On regarde si lien entre les 2 pièce

{

piece.Radiateur = false; // Si oui on coupe

}

else if (pieceVoisine.OuvertureCommuneOuvert() != null) // Sinon si piècevoisine (qui a ouvertureExt ouverte) a une ouvertueCommune ouverte

{

piece.Radiateur = false; // On coupe chauffage

}

else

{

piece.Radiateur = true; // Sinon on allume

}

}

else // Si pas de pièce voisine ouvertureExt ouverte

{

if (piece.Consigne > piece.Temperature) //Test consigne

{

piece.Radiateur = true;

}

else

{

piece.Radiateur = false;

}

}

// Moyenne des 2 température

Double temperature;

temperature = ((piece.OuvertureCommuneOuvert().Piece1.Temperature \* piece.OuvertureCommuneOuvert().Piece1.Surface) + (piece.OuvertureCommuneOuvert().Piece2.Temperature \* piece.OuvertureCommuneOuvert().Piece2.Surface)) / (piece.OuvertureCommuneOuvert().Piece1.Surface + piece.OuvertureCommuneOuvert().Piece2.Surface);

piece.OuvertureCommuneOuvert().Piece1.Temperature = temperature;

piece.OuvertureCommuneOuvert().Piece2.Temperature = temperature;

}

else // Sinon (pas d'ouvertures commune ni vers ext)

{

if (piece.Consigne > piece.Temperature) // test consignes

{

piece.Radiateur = true;

}

else

{

piece.Radiateur = false;

}

}

// Calcule des apport

double apport = 0;

double apportT = 0;

foreach (TParoi paroi in piece.ListeParois)

{

if (paroi.Piece1 == piece) // Si on se trouve dans pièce 1

{

if (paroi.Piece2 != null) // Si pièce 2 != de l'extérieur

{

apport = apport + ((paroi.Piece2.Temperature - piece.Temperature) \* paroi.SurfaceReel) / paroi.CoefficientThermo;

}

else // Si c'est l'extérieur

{

apport = apport + ((this.temperatureExt - piece.Temperature) \* paroi.SurfaceReel) / paroi.CoefficientThermo;

}

}

else // Si on se trouve dans pièce 2

{

if (paroi.Piece1 != null) // Si piece 1 n'est pas extérieur

{

apport = apport + ((paroi.Piece1.Temperature - piece.Temperature) \* paroi.SurfaceReel) / paroi.CoefficientThermo;

}

else // Si pièce 1 est extérieur

{

apport = apport + ((this.temperatureExt - piece.Temperature) \* paroi.SurfaceReel) / paroi.CoefficientThermo;

}

}

if (piece.Radiateur) // On test radiateur

{

apport = apport + piece.CoefficientRadiateur;

}

int masseAir = piece.Surface \* hauteur; // Calcule masseAir en fonction surface pièce

int intervalle;

intervalle = TrackBar.Value;

if (intervalle != 1)

{

intervalle = (intervalle - 1) \* 5 \* 60;

}

double temperatureKelvin = piece.Temperature + 273.15;

apportT = (apport / masseAir) / 1004;

apportT = (apportT / 100) \* intervalle;

piece.Temperature = piece.Temperature + apportT; // On ajoute apport a la température

}

}

}

}

}

## Import Excell

private void buttonExcell\_Click(object sender, EventArgs e)

{

// Eléments d'EXCEL

Excel.Application oXL; // Objet Excel application

Excel.Workbook oWB; // Classeur

Excel.Worksheet oSheet; // Feuille

Excel.Range oRng; // Cellules

oXL = new Excel.Application(); // OUVERTURE EXCEL

oXL.Visible = true; // excel visible

//Obtenir un nouveau classeur

oWB = (Excel.Workbook)(oXL.Workbooks.Add(1));

oSheet = (Excel.Worksheet)oWB.ActiveSheet; // et la 1ère feuille

oRng = oSheet.get\_Range("A1", "F1"); // sélection première ligne

oRng.Font.Bold = true; // et passage en GRAS

for (int k = 0; k < dataGridViewLogPiece.ColumnCount; k++)

{

oSheet.Cells[1, k + 1] = dataGridViewLogPiece.Columns[k].HeaderCell.Value.ToString();

}

for (int i = 0; i < dataGridViewLogPiece.Rows.Count - 1; i++)

{

for (int j = 0; j < dataGridViewLogPiece.ColumnCount; j++)

{

//Debug.Print(dataGridViewLogPiece[j, i].Value.ToString());

oSheet.Cells[i+1, j+1] = dataGridViewLogPiece[j, i].Value.ToString();

}

}

oSheet.Rows.AutoFit();

oSheet.Columns.AutoFit();

}