|  |  |
| --- | --- |
| **Compte rendu STAGE**  ***Année 2023-2024*** | **SEMAINE 2**  ***09/01/2023 – 13/01/2023*** |

***Sommaire / Summary***

Table des matières

[1. Présentation de l’entreprise. 3](#_Toc124512946)

[2. Mes missions. 4](#_Toc124512947)

[1.1 Simulateur\_CEOG. 4](#_Toc124512948)

[1.1.1 Documents et maquette. 4](#_Toc124512949)

[1.1.2 Modifications graphiques. 8](#_Toc124512950)

[1.1.3 Modifications code 10](#_Toc124512951)

[3. Outils utilisés 33](#_Toc124512952)

[4. Conclusion. 34](#_Toc124512953)

[5. Niko-Niko. 35](#_Toc124512954)

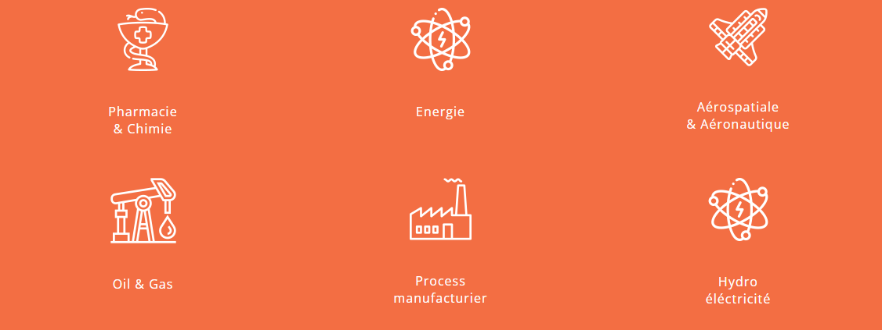
# Présentation de l’entreprise.



**ICONE AUTOMATISATION  
Société spécialisée dans la conception, l’intégration, la mise en service et la maintenance de système automatisés et informatisés pour les outils de production et les moyens d’essais.**

Créé en 1985, Icône Automation est spécialisée dans la conception, l’intégration, la mise en service et la maintenance de systèmes automatisés et informatisés pour les outils de production et les moyens d’essais. Icône Automation a développé depuis de nombreuses années son expertise dans la conception et la réalisation de contrôles commandes en milieu sensible.

Capable de prendre en compte les particularités des secteurs d’activités, Icone sait accompagner dans la réalisation des systèmes d’automatisation et d’informatisation de process industriels.



# Mes missions.

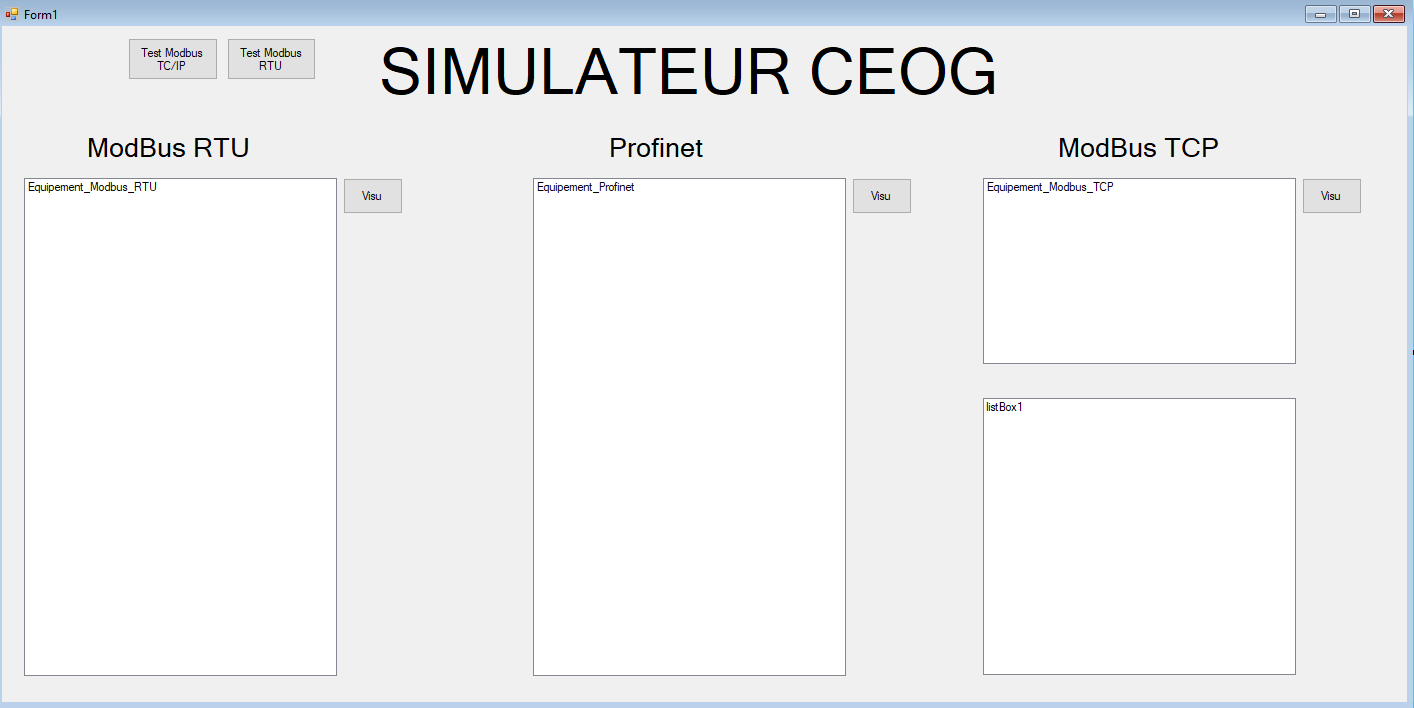
## Simulateur\_CEOG.

### 1.1.1 Documents et maquette.

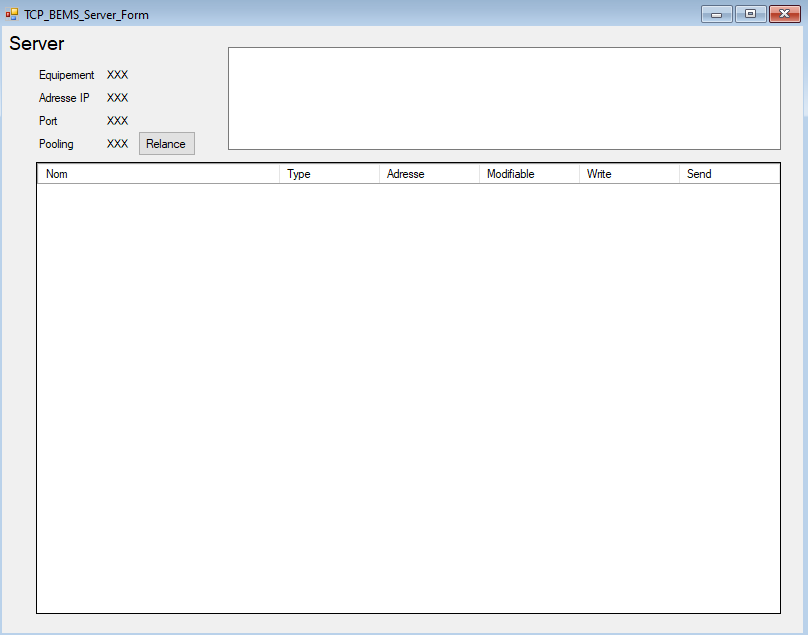
Pour cette semaine, un nouveau projet m’a été affecter.

Le projet se nomme Simlateur\_CEOG et à pour but de faire une application simulation des connexions entre des équipements contenant des protocoles Modbus.

J’ai été fournis pour cela un fichier de documentation du projet ainsi qu’une application C# maquette fait par mon tuteur :



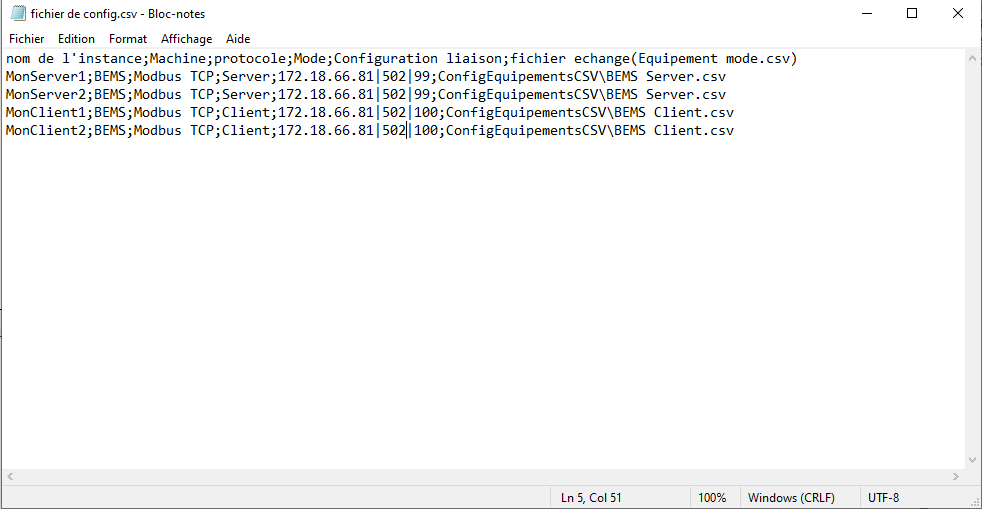
Exemple d’interface d’un Modbus :



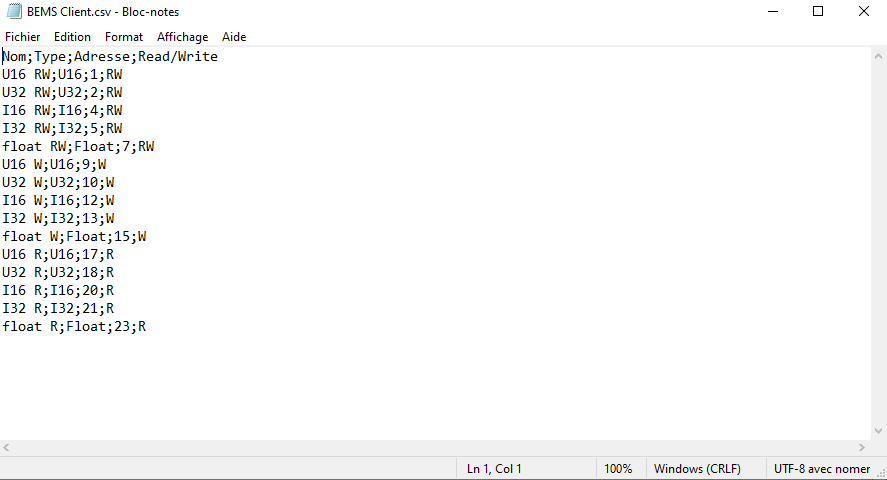
Avant de faire quoi que ce soit, j’ai pris le temps de me renseigner sur les Modbus car je ne connaissais que ne nom.  
J’y ai vite découvert des informations intéressantes qui seront utiles pour le projet tel que les diffèrent protocole (TCP/IP et RTU) et comment ils communiquent (Serveur/Client et Master/Slave).  
Bien comprendre entièrement leurs fonctionnements m’as pris du temps et j’en ai appris même après avoir commencer la modification du projet.

Ensuite j’ai étudié la maquette et le code ainsi que la DLL (EasyModbus) donnée dans le projet.

Le projet consiste a utilisé des fichiers en .csv pour instancier des classes représentant les équipements à simuler ainsi que leurs interfaces.  
Voici un exemple de fichier de configuration (Remplie et choisis par l’utilisateur) :



Et voici un fichier de configuration d’un équipement (définie par le créateur de l’application) :

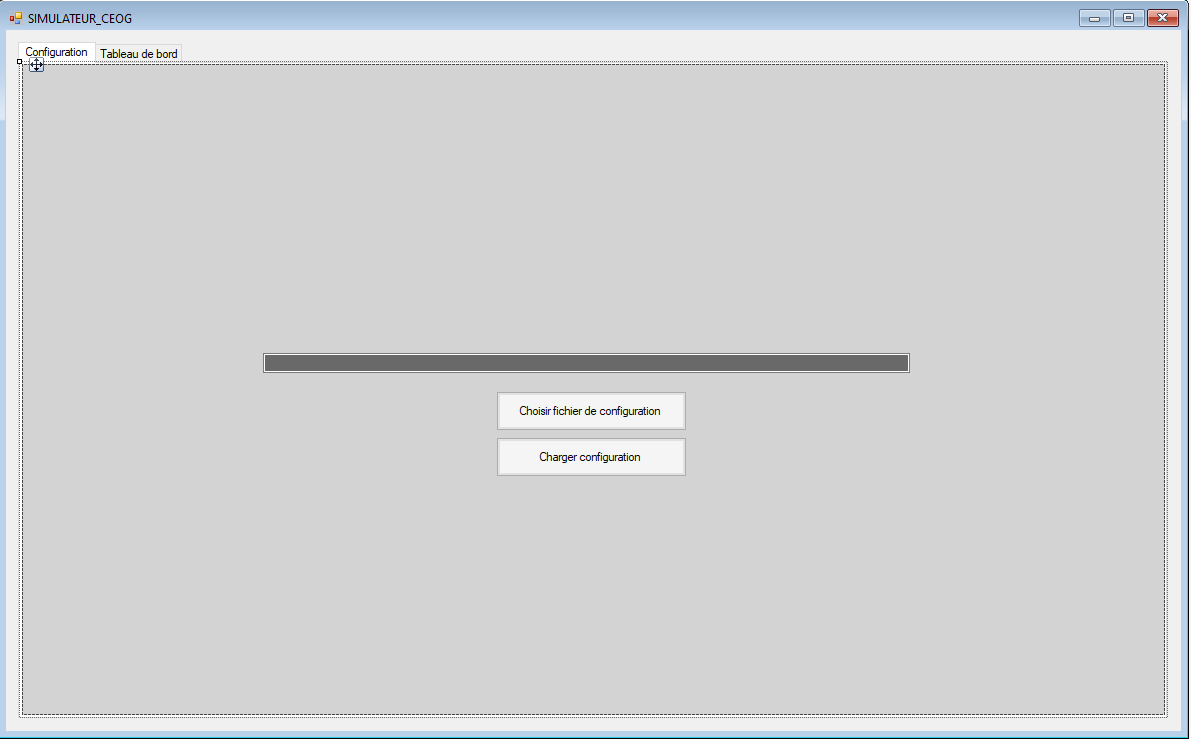


Après avoir regarder le fonctionnement de l’application,  
J’y ai vite remarquer plusieurs problèmes :

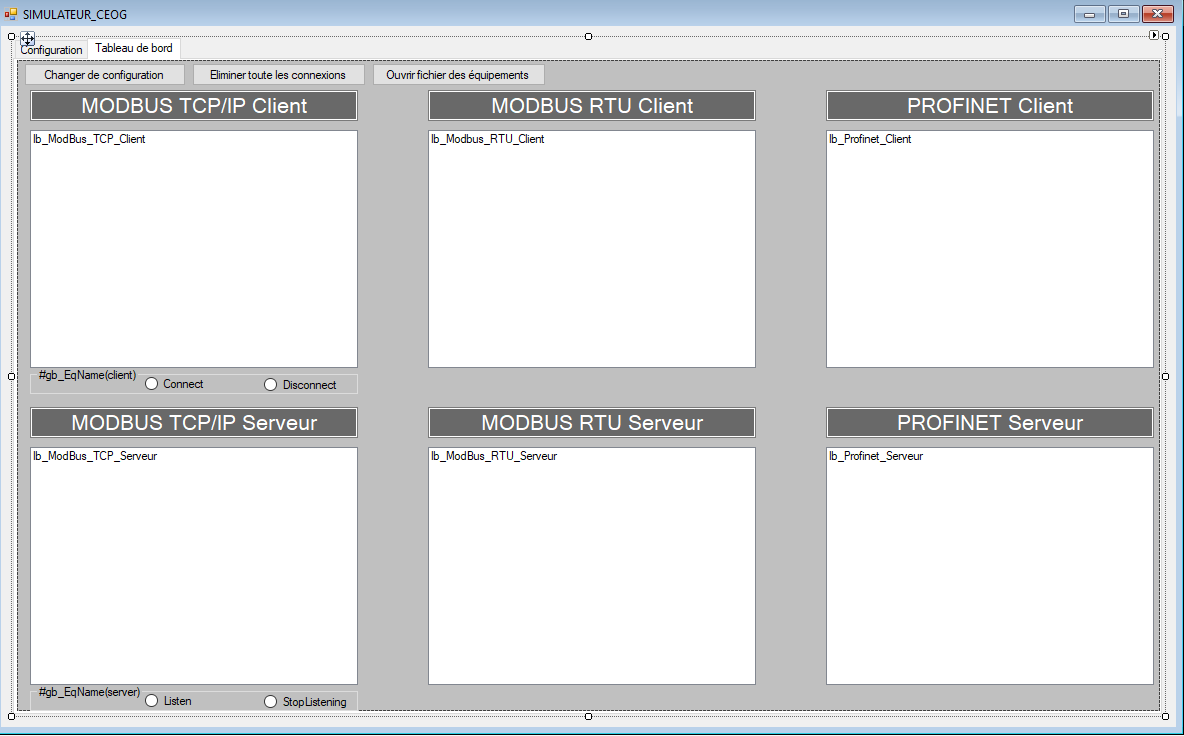
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Nom du problème | Description | Gravité   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  | |
| Interface non adapté | L’interface n’est pas adaptée pour accueillir toutes les possibilités d’équipements |  |
| Aucun choix de connexions | Le choix des connexions entre les equipements n’est pas disponible |  |
| Choix du fichier de config non disponible | Le choix du fichier de configuration n’est pas présent (le fichier de config est codé en dur) |  |
| Manque d’options et de fonctionnalités | Impossible de changer de fichier de config ou d’ouvrir les configurations des equipements |  |
| Grande Répétition de code | Plusieurs centaines de lignes répétées ainsi qu’un code qui demandera beaucoup de copier-coller pour les différents équipements |  |
| Mauvaise utilisation de classes | Classes pour les équipements vide et non utilisés |  |
| Code non fonctionnel | Une partie du code ne fonctionne pas au lancement de l’application (+ 0 commentaires) |  |

### 1.1.2 Modifications graphiques.

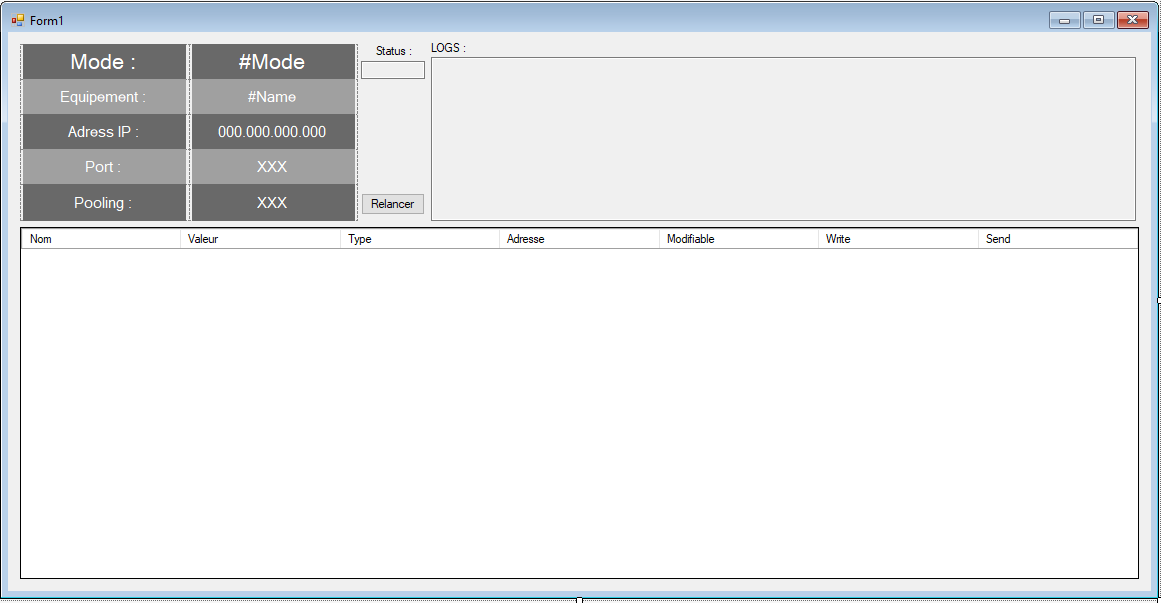
Suite à ces problèmes, j’ai décidé (sous l’accord de mon tuteur) de reprendre le projet de 0 pour partir sur une base convenable.  
Tout d’abord j’y ai apporté des modifications graphiques pour correspondre aux besoins.

Premièrement, une page d’accueil qui force l’utilisateur a choisir son fichier de configuration :  


Ensuite le menu des équipements :



Et l’interface pour un équipement :



En partant de cette interface plus adaptée, j’ai pu partir sur une base solide pour commencer à coder.

### 1.1.3 Modifications code

Tout d’abord il me fallait une logique/un plan

J’ai décidé de faire les connexions de type Modbus TCP/IP et ai donc crée la classe suivante

Il me faut une classe qui contient :

* Les informations de l’équipement :

public class TCP\_Class

{

public string Nom;

public string Machine;

public string Protocol;

public string Configuration\_Liaison;

public string Mode;

public string Fichier\_Echange;

public string AdressIP;

public int Port;

public int Pooling;

public ModbusClient Modbus\_Client;

public ModbusServer Modbus\_Server;

public TCP\_Form Equipement\_Form;

public string ConnexionStatus = "Disconnected";

public TCP\_Class(string Nom, string Machine, string Protocol, string Configuration\_Liaison, string Mode, string Fichier\_Echange)

{

this.Nom = Nom;

this.Machine = Machine;

this.Protocol = Protocol;

this.Configuration\_Liaison = Configuration\_Liaison;

this.Mode = Mode;

this.Fichier\_Echange = AppDomain.CurrentDomain.BaseDirectory + Fichier\_Echange;

this.AdressIP = Configuration\_Liaison.Split('|')[0];

this.Port = Convert.ToInt32(Configuration\_Liaison.Split('|')[1]);

this.Pooling = Convert.ToInt32(Configuration\_Liaison.Split('|')[2]);

}

}

Une fois que nous avons une classe avec toute les informations nécessaires, nous pouvons commencer la lécture du fichier de configuration :

private void selectConfigFile(Object sender, EventArgs e)

{

//permet la selection du fichier de configuration

if (configFileSelector.ShowDialog() == DialogResult.OK) //Si le fichier chemin du fichier de config est bon

{

string pathName = configFileSelector.FileName;

tb\_ConfigFileDir.Text = pathName;

} else

{

MessageBox.Show("Erreur lors de la récupération du fichier de configuration");

}

}

private void ReadConfigFile(Object sender, EventArgs e)

{

string pathName = tb\_ConfigFileDir.Text;

try //Si la lécture du fichier de config se passe correctement

{

tb\_ConfigFileDir.Text = pathName;

StreamReader reader = new StreamReader(File.OpenRead(pathName));

List<string[]> fileLines = new List<string[]>(); //liste contenant les lignes du fichier

while (!reader.EndOfStream) // tant que la lécture n'est pas au bout du fichier

{

string line = reader.ReadLine();

string[] values = line.Split(';');

fileLines.Add(values);

}

fileLines.RemoveAt(0); // Enleve la premiere ligne du fichier (les en-tetes)

foreach (string[] line in fileLines) // pour chaque ligne du fichier

{

#region ALIMENTATION LISTE TCP D'EQUIPEMENTS EN TCP

if (line[2] == "Modbus TCP")

{

//récupération des données et création de nos classes

string Nom = line[0];

string Machine = line[1];

string Protocole = line[2];

string Mode = line[3];

string Configuration\_Liaison = line[4];

string Fichier\_Echange = line[5];

TCP\_Class newTCP\_Equipement = new TCP\_Class(Nom, Machine, Protocole, Configuration\_Liaison, Mode, Fichier\_Echange);

listTCP\_Equipement.Add(newTCP\_Equipement);

}

#endregion

}

//Changement de page vers le tableau de bord

tc\_Main.SelectedIndex = 1;

}

catch //Si erreur pendant la lécture du fichier de config

{

MessageBox.Show("Erreur lécture du fichier de configuration");

}

#region Ajout form dans les classes

foreach (TCP\_Class TCP in listTCP\_Equipement)

{

TCP\_Form Form = new TCP\_Form(TCP);

TCP.Equipement\_Form = Form;

TCP.Equipement\_Form.set\_tbStatusText("Disconnected");

TCP.Equipement\_Form.Show(); //Ceci permet de charger les formulaire puis de les cacher

TCP.Equipement\_Form.Hide();

}

#endregion

#region Ajout dans les ListBox

foreach (TCP\_Class TCPequipement in listTCP\_Equipement)

{

switch (TCPequipement.Mode)

{

case "Client":

lb\_ModBus\_TCP\_Client.Items.Add(TCPequipement.Nom);

break;

case "Server":

lb\_ModBus\_TCP\_Serveur.Items.Add(TCPequipement.Nom);

break;

}

}

#endregion

#region Ajout des modbus Client/Serveur

foreach (TCP\_Class TCPclass in listTCP\_Equipement) //Ajout des modbus de la DLL dans les classes

{

switch (TCPclass.Mode)

{

case "Client":

ModbusClient client = new ModbusClient();

client.IPAddress = TCPclass.AdressIP;

client.Port = TCPclass.Port;

TCPclass.Modbus\_Client = client;

break;

case "Server":

ModbusServer server = new ModbusServer();

IPHostEntry host = Dns.GetHostEntry(Dns.GetHostName());

server.LocalIPAddress = host.AddressList[3];

server.Port = TCPclass.Port;

TCPclass.Modbus\_Server = server;

break;

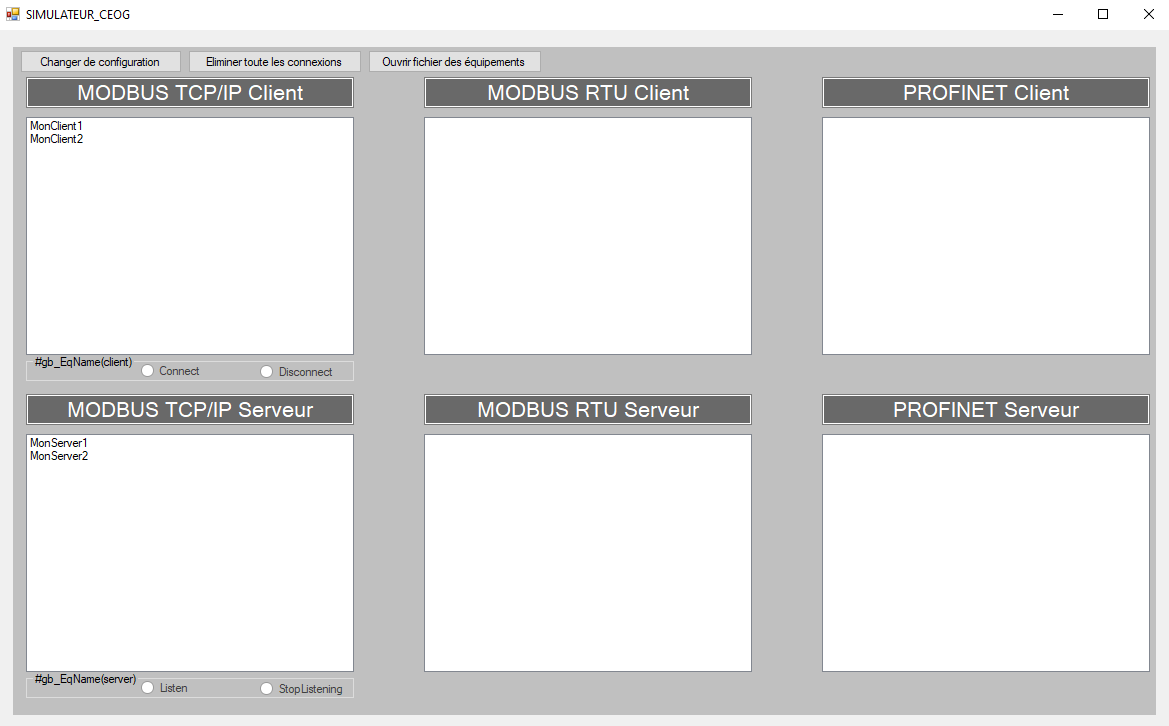
}

}

#endregion

}

Le code précèdent permet l’instanciation des classes avec leurs informations ainsi que l’affichage graphique :



Ensuite il fallait coder la possibilité de connecté les équipements entre eux grâce au radio buttons.  
Voici la logique que j’ai utilisé :

Dés qu’un Item de la ListBox est choisi

Si le status de l’équipements est connecté, décocher les 2 radios buttons puis Enable le bouton « Disconnect »

Si le status est déconnecté, décocher les 2 radios buttons puis Enable le bouton « Connect »

Il faut aussi changer les boutons quand l’appareil se connecte :

Si un radiobuttons se coche,

Disable les 2 radiosButtons

Connecter l’équipement

Changer le statuts de connexion de l’équipement

Si le status de l’équipements est connecté, décocher les 2 radios buttons puis Enable le bouton « Disconnect »

Si le status est déconnecté, décocher les 2 radios buttons puis Enable le bouton « Connect »

Si nous suivons cette logique, nous pouvons connecter nos appareils entre eux de manière libre.  
Voici le code qui permet ceci :

private void Lb\_ModBus\_TCP\_SelectedIndexChanged(object sender, EventArgs e) //Quand l'element choisie de la liste change

{

#region affichage radiobuttons TCP client quand changement de selection d'item

if (sender == lb\_ModBus\_TCP\_Client)

{

try

{

string equipementName = lb\_ModBus\_TCP\_Client.SelectedItem.ToString();

foreach (TCP\_Class TCPequipement in listTCP\_Equipement)

{

if (TCPequipement.Nom == equipementName) //recupere la bonne classe

{

gb\_TCP\_Client\_Connexion.Text = TCPequipement.Nom;

switch (TCPequipement.ConnexionStatus)

{

case "Connected": // si l'élement est connecter

rb\_TCP\_Client\_Connect.Checked = false;

rb\_TCP\_Client\_Disconnect.Checked = false;

rb\_TCP\_Client\_Connect.Enabled = false;

rb\_TCP\_Client\_Disconnect.Enabled = true;

break;

case "Disconnected": // si l'élément est deconnecter

rb\_TCP\_Client\_Connect.Checked = false;

rb\_TCP\_Client\_Disconnect.Checked = false;

rb\_TCP\_Client\_Connect.Enabled = true;

rb\_TCP\_Client\_Disconnect.Enabled = false;

break;

}

}

}

}

catch

{

}

}

#endregion

#region affichage radiobuttons TCP serveur quand changement de selection d'item

if (sender == lb\_ModBus\_TCP\_Serveur)

{

try

{

string equipementName = lb\_ModBus\_TCP\_Serveur.SelectedItem.ToString();

foreach (TCP\_Class TCPequipement in listTCP\_Equipement)

{

if (TCPequipement.Nom == equipementName)

{

gb\_TCP\_Serveur\_Connexion.Text = TCPequipement.Nom;

switch (TCPequipement.ConnexionStatus)

{

case "Listening":

rb\_TCP\_Serveur\_Connect.Checked = false;

rb\_TCP\_Serveur\_Disconnect.Checked = false;

rb\_TCP\_Serveur\_Connect.Enabled = false;

rb\_TCP\_Serveur\_Disconnect.Enabled = true;

break;

case "Disconnected":

rb\_TCP\_Serveur\_Connect.Checked = false;

rb\_TCP\_Serveur\_Disconnect.Checked = false;

rb\_TCP\_Serveur\_Connect.Enabled = true;

rb\_TCP\_Serveur\_Disconnect.Enabled = false;

break;

}

}

}

}

catch

{

}

}

#endregion

}

private void tcpRadioButtons\_CheckedChanged(Object sender, EventArgs e)

{

/\*Cette fonction se trouve dans 2 radiobuttons donc pour eviter qu'elle se déclanche deux fois (elle se déclanche a chaque changement sur 1 radiobutton)

C'est pour cela que les radio button ne sont jamais selectioner par le code, explication :

si nous changons de client sélectioner, il faut que les radios button change en fonction de son status de connexion

seulement, si nous changons directement le radiobutton selectioné, cette fonction est appelé et va tenter une connexion non voulu

vu que nous voulons un changement purement graphique, j'utilise l'option Enabled des radiobutton qui marche bien pour ce cas

\*/

if (((RadioButton)sender).Checked) // pour ne pas que la fonction se lance 2 fois (1 seule bouton est .Checked donc elle se lance une fois)

{

#region Si RadioButton TCP Client changé

if (sender == rb\_TCP\_Client\_Connect) //Regarde quel radiobutton appel l'event

{

string equipementName = lb\_ModBus\_TCP\_Client.SelectedItem.ToString();

foreach (TCP\_Class TCPEquipement in listTCP\_Equipement)

{

if (equipementName == TCPEquipement.Nom)

{

connectTCPEquipement(TCPEquipement); //Connecte l'equipement grace a la fonction écrite plus haut

}

}

}

if (sender == rb\_TCP\_Client\_Disconnect)

{

string equipementName = lb\_ModBus\_TCP\_Client.SelectedItem.ToString();

foreach (TCP\_Class TCPEquipement in listTCP\_Equipement)

{

if (equipementName == TCPEquipement.Nom)

{

disconnectTCPEquipement(TCPEquipement);

}

}

}

#endregion

#region Si RadioButton TCP Serveur changé

if (sender == rb\_TCP\_Serveur\_Connect)

{

string equipementName = lb\_ModBus\_TCP\_Serveur.SelectedItem.ToString();

foreach (TCP\_Class TCPEquipement in listTCP\_Equipement)

{

if (equipementName == TCPEquipement.Nom)

{

connectTCPEquipement(TCPEquipement);

}

}

}

if (sender == rb\_TCP\_Serveur\_Disconnect)

{

string equipementName = lb\_ModBus\_TCP\_Serveur.SelectedItem.ToString();

foreach (TCP\_Class TCPEquipement in listTCP\_Equipement)

{

if (equipementName == TCPEquipement.Nom)

{

disconnectTCPEquipement(TCPEquipement);

}

}

}

#endregion

}

}

private void connectTCPEquipement(TCP\_Class TCPclass)

{

switch (TCPclass.Mode)

{

case "Client": //Connexion client

try

{

if (TCPclass.Modbus\_Client.Connected == false) //si le client n'est pas connécté

{

TCPclass.Equipement\_Form.logMessage("Trying connexion at : IP->" + TCPclass.AdressIP + " / Port->" + TCPclass.Port.ToString());

TCPclass.Modbus\_Client.Connect();

TCPclass.ConnexionStatus = "Connected";

TCPclass.Equipement\_Form.set\_tbStatusText("Connected");

TCPclass.Equipement\_Form.logMessage("Connexion established at : IP->" + TCPclass.AdressIP + " / Port->" + TCPclass.Port.ToString());

//Ceci permet de disable les radiobutton en fonction de la connexion (Explication dans la fonction des radiobutton)

rb\_TCP\_Client\_Connect.Checked = false;

rb\_TCP\_Client\_Disconnect.Checked = false;

rb\_TCP\_Client\_Connect.Enabled = false;

rb\_TCP\_Client\_Disconnect.Enabled = true;

}

}

catch (Exception e)

{

TCPclass.Equipement\_Form.logMessage("Connexion failed at : IP->" + TCPclass.AdressIP + " / Port->" + TCPclass.Port.ToString());

rb\_TCP\_Client\_Connect.Checked = false;

rb\_TCP\_Client\_Disconnect.Checked = false;

rb\_TCP\_Client\_Connect.Enabled = true;

rb\_TCP\_Client\_Disconnect.Enabled = false;

MessageBox.Show("Erreur connexion Client (verifier qu'un serveur écoute et que les adresse IP sont bien attribuées : " + TCPclass.Nom + " | " + e.Message);

}

break;

case "Server": //Ecoute serveur

try

{

TCPclass.Equipement\_Form.logMessage("Trying listen on : IP->" + TCPclass.AdressIP + " / Port->" + TCPclass.Port.ToString());

TCPclass.Modbus\_Server.Listen();

TCPclass.Modbus\_Server.HoldingRegistersChanged += new ModbusServer.HoldingRegistersChangedHandler(TCPclass.Equipement\_Form.registersChanged);

TCPclass.Equipement\_Form.logMessage("Listen successful on : IP." + TCPclass.AdressIP + " Port." + TCPclass.Port.ToString());

TCPclass.ConnexionStatus = "Listening";

TCPclass.Equipement\_Form.set\_tbStatusText("Listening");

rb\_TCP\_Serveur\_Connect.Checked = false;

rb\_TCP\_Serveur\_Disconnect.Checked = false;

rb\_TCP\_Serveur\_Connect.Enabled = false;

rb\_TCP\_Serveur\_Disconnect.Enabled = true;

}

catch (Exception e)

{

TCPclass.Equipement\_Form.logMessage("Listen failed on : IP->" + TCPclass.AdressIP + " / Port->" + TCPclass.Port.ToString());

rb\_TCP\_Serveur\_Connect.Checked = false;

rb\_TCP\_Serveur\_Disconnect.Checked = false;

rb\_TCP\_Serveur\_Connect.Enabled = true;

rb\_TCP\_Serveur\_Disconnect.Enabled = false;

MessageBox.Show("Erreur écoute Serveur : " + TCPclass.Nom + " | " + e.Message);

}

break;

}

}

private void disconnectTCPEquipement(TCP\_Class TCPclass)

{

switch (TCPclass.Mode)

{

case "Client": //Déconnexion client

try

{

if (TCPclass.Modbus\_Client.Connected)

{

TCPclass.Modbus\_Client.Disconnect();

TCPclass.Equipement\_Form.logMessage("Client successfully disconnected from : IP->" + TCPclass.AdressIP + " / Port->" + TCPclass.Port.ToString());

TCPclass.ConnexionStatus = "Disconnected";

TCPclass.Equipement\_Form.set\_tbStatusText("Disconnected");

rb\_TCP\_Client\_Connect.Checked = false;

rb\_TCP\_Client\_Disconnect.Checked = false;

rb\_TCP\_Client\_Connect.Enabled = true;

rb\_TCP\_Client\_Disconnect.Enabled = false;

}

}

catch (Exception e)

{

MessageBox.Show("Erreur déconnexion Client : " + TCPclass.Nom + " | " + e.Message);

}

break;

case "Server": // Arret écoute serveur

try

{

TCPclass.Modbus\_Server.StopListening();

TCPclass.Equipement\_Form.logMessage("Server successfully stopped listening on : IP->" + TCPclass.AdressIP + " / Port->" + TCPclass.Port.ToString());

TCPclass.ConnexionStatus = "Disconnected";

TCPclass.Equipement\_Form.set\_tbStatusText("Disconnected");

rb\_TCP\_Serveur\_Connect.Checked = false;

rb\_TCP\_Serveur\_Disconnect.Checked = false;

rb\_TCP\_Serveur\_Connect.Enabled = true;

rb\_TCP\_Serveur\_Disconnect.Enabled = false;

}

catch (Exception e)

{

//MessageBox.Show("Erreur stop écoute Serveur : " + TCPclass.Nom + " | " + e.Message);

}

break;

}

}

Ensuite il faut pouvoir ouvrir l’interface des équipements :

private void Lb\_ModBus\_TCP\_DoubleClick(object sender, EventArgs e)

{

//affiche le form de la classe séléctioné

#region formulaire TCP client

if (sender == lb\_ModBus\_TCP\_Client)

{

try

{

string selectedEquipementName = lb\_ModBus\_TCP\_Client.SelectedItem.ToString();

foreach (TCP\_Class TCPclass in listTCP\_Equipement)

{

if (selectedEquipementName == TCPclass.Nom) //récupération de la bonne classe

{

TCPclass.Equipement\_Form.Show();

}

}

}

catch // si aucun item selectioné, ne rien faire

{

}

}

#endregion

#region formulaire TCP serveur

if (sender == lb\_ModBus\_TCP\_Serveur)

{

try

{

string selectedEquipementName = lb\_ModBus\_TCP\_Serveur.SelectedItem.ToString();

foreach (TCP\_Class TCPclass in listTCP\_Equipement)

{

if (selectedEquipementName == TCPclass.Nom)

{

TCPclass.Equipement\_Form.Show();

}

}

}

catch

{

}

}

#endregion

}

Après cela est venu la partie la plus compliqué, c’est faire marcher et communiquer les équipements et leurs interfaces.

Voici les fonctions graphique de chaque interface ainsi que quelques fonctions moins importantes :

string Mode;

public string equipementName;

string AdressIP;

int Port;

int Pooling;

string Fichier\_echange;

public DateTime Old\_date = DateTime.Now;

public DateTime New\_date = new DateTime();

public int AdresseMax = 0;

TCP\_Class TCPEquipement;

public TCP\_Form(TCP\_Class TCPEquipement) //Constructeur

{

this.Mode = TCPEquipement.Mode;

this.equipementName = TCPEquipement.Nom;

this.AdressIP = TCPEquipement.AdressIP;

this.Port = TCPEquipement.Port;

this.Pooling = TCPEquipement.Pooling;

this.Fichier\_echange = TCPEquipement.Fichier\_Echange;

this.TCPEquipement = TCPEquipement;

InitializeComponent();

logMessage("Initialisation complete.");

StreamReader streamReader = File.OpenText(this.Fichier\_echange);

string headerLine = streamReader.ReadLine();

int count = 0;

while (!streamReader.EndOfStream)

{

string line = streamReader.ReadLine();

string[] values = line.Split(';');

string Nom = values[0];

string Type = values[1];

string Adresse = values[2];

string ReadWrite = values[3];

dgv\_dataHolder.Rows.Add(Nom, 0, Type, Adresse, true);

//MessageBox.Show(Nom + " / " + Type + " / " + Adresse + " / " + ReadWrite);

switch (ReadWrite)

{

case "R":

dgv\_dataHolder.Rows[count].Cells["Write"].ReadOnly = true;

dgv\_dataHolder.Rows[count].Cells["Write"].Style.BackColor = Color.FromArgb(255, 110, 50, 50);

dgv\_dataHolder.Rows[count].Cells["Write"].Style.ForeColor = Color.White;

dgv\_dataHolder.Rows[count].Cells["Write"].Value = "Lécture seule pour | " + Nom;

break;

case "W":

dgv\_dataHolder.Rows[count].Cells["Valeur"].Value = "Ecriture seule pour | " + Nom;

dgv\_dataHolder.Rows[count].Cells["Valeur"].Style.BackColor = Color.FromArgb(255, 110, 50, 50);

dgv\_dataHolder.Rows[count].Cells["Valeur"].Style.ForeColor = Color.White;

break;

}

count += 1;

}

AdresseMax = Convert.ToInt32(dgv\_dataHolder.Rows[count - 1].Cells["Adresse"].Value);

}

private void BEMS\_TCP\_Form\_Load(object sender, EventArgs e)

{

lb\_Equipement\_Mode.Text = Mode;

lb\_Equipement\_Name.Text = equipementName;

lb\_Equipement\_IP.Text = AdressIP;

lb\_Equipement\_Port.Text = Port.ToString();

lb\_Equipement\_Pooling.Text = Pooling.ToString();

this.Text = this.equipementName;

timer1.Interval = this.Pooling;

foreach(DataGridViewRow row in dgv\_dataHolder.Rows)

{

if (Convert.ToInt32(row.Cells[3].Value) > AdresseMax)

{

AdresseMax = Convert.ToInt32(row.Cells[3].Value);

}

}

}

public void set\_tbStatusText(string text)

{

tb\_Status.Text = text;

}

public void logMessage(string msg) // Permet de faire un message de log

{

string now = DateTime.Now.ToString();

string line= now + " ---> " + msg + "\n";

string historique = tb\_Logs.Text;

tb\_Logs.Text = line + "\r\n" + historique;

}

private void BEMS\_TCP\_Form\_FormClosing(object sender, FormClosingEventArgs e) //empeche le form de se supprimer mais juste de se cacher

{

Hide();

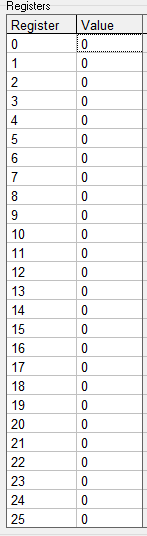
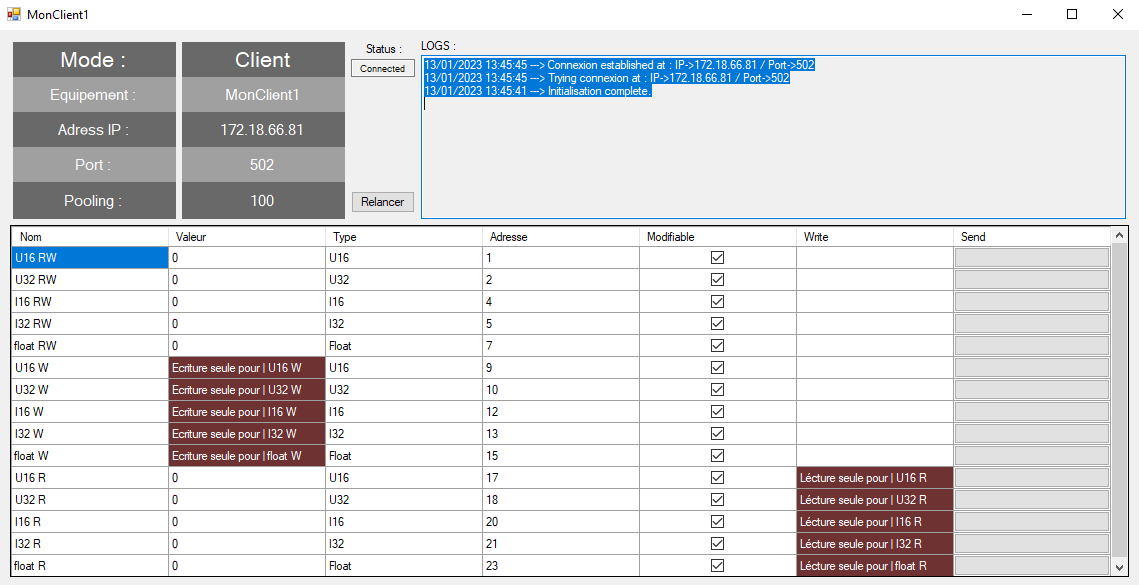
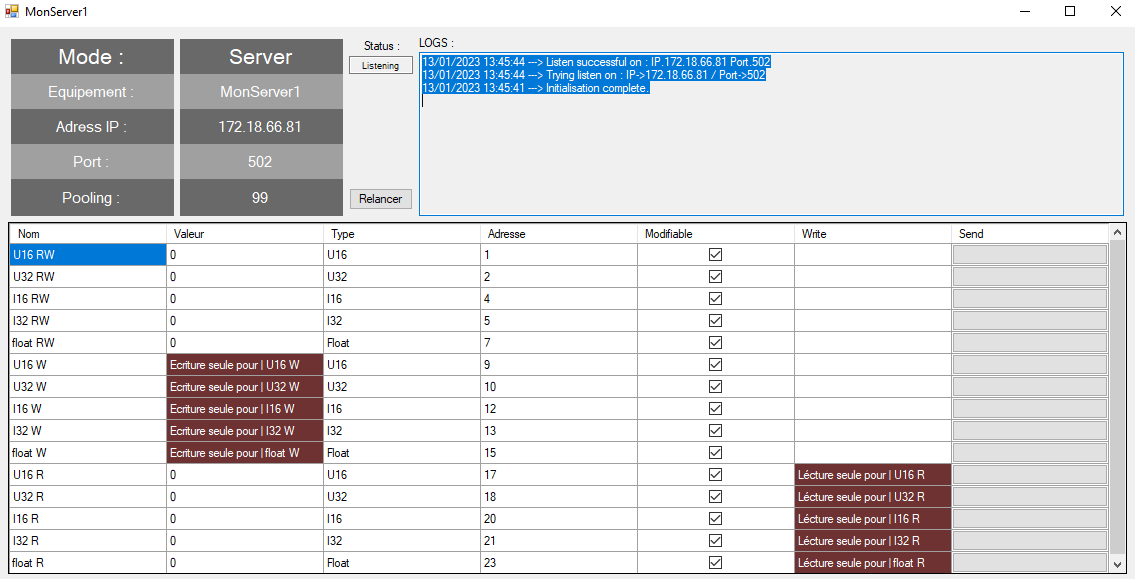
e.Cancel = true;

}

Ensuite est venu la partie la plus compliqué, l’écriture entre Serveur/Client

Pour cela il faut comprendre le fonctionnement.

Voici un schéma représentant le fonctionnement attendu :



Lis chaques (valeur pooling) millisecondes

Lis quand registre modifier

Ecrit dans

Contient

Ecrit dans

Choses à savoir :

* Chaque registre de la table est un Int16 donc 2 Bytes par adresse
* Les registres de vrai Modbus TCP/IP Serveur sont des Int16 qui peuvent accepter les UInt16 (Int16 Min/Max = -32 768 à 32 767 et Uint16 Min/Max = 0 à 65 535)
* Les Uint32/Int32 ou Int prennent 2 registres ou 4 Bytes
* Les floats prennent 2 registres ou 4 Bytes
* Les long ou Int64 m’on pas été demander d’être codé
* Les Uint16/Int16 prennent 1 registre ou 2 Bytes

Avec ces informations, la transmission des valeurs semble simple sauf pour les Uint16/Uint32.

Car un Uint16 prend 1 registre ou 2 Bytes et à pour Min/Max->0/65535  
Alors qu’un registre (Int16) prend 2 Bytes et à pour Min/Max->-32768/32767

Nous devons manipuler les valeurs pour plusieurs cas possible :

Le Uint16 est entre [0 ; 32767], on peut envoyer la valeur car elle rentre dans un Int16

Le Uint16 est entre ]32767 ; 65535], on ne peut envoyer la valeur dans un Int16, j’ai donc manipulé la valeur :

Prenons comme exemple le maximum 65535.

On fait la valeur(65535) – (MaxInt16 + 1)

65535 – (65535+1) = -1

On met donc -1 dans un int16 et on l’envoie au serveur

Le serveur reçoi -1 et fait :

Si la valeur < 0 alors

Valeur = 65535 + (valeur(-1) + 1)

Valeur = 65535 + (-1 + 1)

Valeur = 65535

On retrouve donc notre valeur de base

Maintenant pour expliquer pourquoi je fais le maximum de la donnée + 1  
Si la valeur = 0, quand le serveur reçoi la valeur, celle si ne seras jamais plus petit que 0 donc le calcule ne seras jamais effectuer et la valeur restera à 0.

Après cette longue explication, voici le code d’écriture entre les Modbus (Chaque fonctions se ressemble mais est différente et ne peut être convertie en une seule et même fonction) :

Fonction d’écriture client/serveur (non terminé pour envoie serveur)

//Les 3 fonctions qui suivent qui sont un peu collé sont les fonctions d'envoies, de réceptions et de convertion des valeurs

//1 . le client envoie la valeur (Si serveur envoie la valeur, il écrit juste dans ses registres et passe au 3.)

//2 . Le serveur la reçoi et reconvertie la valeur

//3 . le client récupere les valeurs du serveur et les reconverties

private void sendValueHandler(object sender, DataGridViewCellEventArgs e)

{

if (e.ColumnIndex == 6 && dgv\_dataHolder.CurrentRow.Cells[5].Style.BackColor != Color.FromArgb(255, 110, 50, 50)) //Si c'est le bouton qui est clicker

{

int adresse = Convert.ToInt32(dgv\_dataHolder.CurrentRow.Cells[3].Value); //l'adresse

int value = 0; //la valeur int du datagridview (16/32)

int[] valueTwoRegisters = new int[2]; //conteneur de 2 registre (pour tout /32 ou float)

Single valueFloat = 0; //la valeur float du datagridview

try

{

switch (TCPEquipement.Mode)

{

case "Client": //Si c'est un client

switch (dgv\_dataHolder.CurrentRow.Cells[2].Value.ToString())

{

case "U16":

value = Convert.ToUInt16(dgv\_dataHolder.CurrentRow.Cells[5].Value);

Int16 valInt16 = 0;

if (value > 32767)

{

valInt16 = Convert.ToInt16(value - 65536);

} else

{

valInt16 = Convert.ToInt16(value);

}

TCPEquipement.Modbus\_Client.WriteSingleRegister(adresse, valInt16);

logMessage("Envoie de la valeur : " + value.ToString() + " | à l'adresse : " + adresse.ToString() + " | de type : " + dgv\_dataHolder.CurrentRow.Cells[2].Value.ToString());

break;

case "U32":

UInt32 val = Convert.ToUInt32(dgv\_dataHolder.CurrentRow.Cells[5].Value);

if (val > 2147483647)

{

value = Convert.ToInt32(val - 4294967296L); //Le L spécifie qu'il sagit d'un chiffre de type long (car sinon erreur)

} else

{

value = Convert.ToInt32(val);

}

valueTwoRegisters = EasyModbus.ModbusClient.ConvertIntToRegisters(value);

TCPEquipement.Modbus\_Client.WriteMultipleRegisters(adresse, valueTwoRegisters);

logMessage("Envoie de la valeur : " + val.ToString() + " | à l'adresse : " + adresse.ToString() + " | de type : " + dgv\_dataHolder.CurrentRow.Cells[2].Value.ToString());

break;

case "I16": //Aucun besoin de convertion pour de l'I16

value = Convert.ToInt16(dgv\_dataHolder.CurrentRow.Cells[5].Value); //la valeur du datagridview

TCPEquipement.Modbus\_Client.WriteSingleRegister(adresse, value);

logMessage("Envoie de la valeur : " + value.ToString() + " | à l'adresse : " + adresse.ToString() + " | de type : " + dgv\_dataHolder.CurrentRow.Cells[2].Value.ToString());

break;

case "I32":

value = Convert.ToInt32(dgv\_dataHolder.CurrentRow.Cells[5].Value);

valueTwoRegisters = EasyModbus.ModbusClient.ConvertIntToRegisters(value);

TCPEquipement.Modbus\_Client.WriteMultipleRegisters(adresse, valueTwoRegisters);

logMessage("Envoie de la valeur : " + value.ToString() + " | à l'adresse : " + adresse.ToString() + " | de type : " + dgv\_dataHolder.CurrentRow.Cells[2].Value.ToString());

break;

case "Float":

valueFloat = Convert.ToSingle(dgv\_dataHolder.CurrentRow.Cells[5].Value);

valueTwoRegisters = EasyModbus.ModbusClient.ConvertFloatToRegisters(valueFloat);

TCPEquipement.Modbus\_Client.WriteMultipleRegisters(adresse, valueTwoRegisters);

logMessage("Envoie de la valeur : " + valueFloat.ToString() + " | à l'adresse : " + adresse.ToString() + " | de type : " + dgv\_dataHolder.CurrentRow.Cells[2].Value.ToString());

break;

}

break;

case "Server": //Si c'est un Serveur

switch (dgv\_dataHolder.CurrentRow.Cells[2].Value.ToString()) //Important ! Pour X raisons, l'adresse doit 1 fois plus haute que l'adresse d'envoie (raison du +1/+2)

{

case "U16": // Non fonctionnel

/\*value = Convert.ToUInt16(dgv\_dataHolder.CurrentRow.Cells[5].Value);

valueTwoRegisters = EasyModbus.ModbusClient.ConvertIntToRegisters(value);

TCPEquipement.Modbus\_Server.holdingRegisters.localArray[adresse + 1] = (short)valueTwoRegisters[0];

\*/TCPEquipement.Modbus\_Server.holdingRegisters.localArray[adresse + 2] = (short)valueTwoRegisters[1];

logMessage("Envoie de la valeur : " + value.ToString() + " | à l'adresse : " + adresse.ToString() + " | de type : " + dgv\_dataHolder.CurrentRow.Cells[2].Value.ToString());

break;

case "U32": // Non fonctionnel

UInt32 val2 = Convert.ToUInt32(dgv\_dataHolder.CurrentRow.Cells[5].Value);

/\*valueFourRegisters = EasyModbus.ModbusClient.ConvertLongToRegisters(val2);

TCPEquipement.Modbus\_Server.holdingRegisters.localArray[adresse + 1] = (short)valueFourRegisters[0];

TCPEquipement.Modbus\_Server.holdingRegisters.localArray[adresse + 2] = (short)valueFourRegisters[1];

TCPEquipement.Modbus\_Server.holdingRegisters.localArray[adresse + 3] = (short)valueFourRegisters[2];

TCPEquipement.Modbus\_Server.holdingRegisters.localArray[adresse + 4] = (short)valueFourRegisters[3];

\*/logMessage("Envoie de la valeur : " + val2.ToString() + " | à l'adresse : " + adresse.ToString() + " | de type : " + dgv\_dataHolder.CurrentRow.Cells[2].Value.ToString());

break;

case "I16":

Int16 val = Convert.ToInt16(dgv\_dataHolder.CurrentRow.Cells[5].Value); //la valeur du datagridview

TCPEquipement.Modbus\_Server.holdingRegisters.localArray[adresse + 1] = val;

logMessage("Envoie de la valeur : " + val.ToString() + " | à l'adresse : " + adresse.ToString() + " | de type : " + dgv\_dataHolder.CurrentRow.Cells[2].Value.ToString());

break;

case "I32":

value = Convert.ToInt32(dgv\_dataHolder.CurrentRow.Cells[5].Value);

valueTwoRegisters = EasyModbus.ModbusClient.ConvertIntToRegisters(value);

TCPEquipement.Modbus\_Server.holdingRegisters.localArray[adresse + 1] = (short)valueTwoRegisters[0];

TCPEquipement.Modbus\_Server.holdingRegisters.localArray[adresse + 2] = (short)valueTwoRegisters[1];

logMessage("Envoie de la valeur : " + value.ToString() + " | à l'adresse : " + adresse.ToString() + " | de type : " + dgv\_dataHolder.CurrentRow.Cells[2].Value.ToString());

break;

case "Float":

valueFloat = Convert.ToSingle(dgv\_dataHolder.CurrentRow.Cells[5].Value);

valueTwoRegisters = EasyModbus.ModbusClient.ConvertFloatToRegisters(valueFloat);

TCPEquipement.Modbus\_Server.holdingRegisters.localArray[adresse + 1] = (short)valueTwoRegisters[0];

TCPEquipement.Modbus\_Server.holdingRegisters.localArray[adresse + 2] = (short)valueTwoRegisters[1];

logMessage("Envoie de la valeur : " + valueFloat.ToString() + " | à l'adresse : " + adresse.ToString() + " | de type : " + dgv\_dataHolder.CurrentRow.Cells[2].Value.ToString());

break;

}

break;

}

} catch (Exception err)

{

logMessage("Erreur lors de l'envoie : " + err.Message);

}

}

}

Fonction de lécture du serveur quand registre modifier

public void registersChanged(int register, int numberOfRegisters)

{

try

{

int[] valueTwoRegisters = new int[2];

int value = 0;

Single valueFloat = 0;

valueTwoRegisters[0] = TCPEquipement.Modbus\_Server.holdingRegisters.localArray[register];

valueTwoRegisters[1] = TCPEquipement.Modbus\_Server.holdingRegisters.localArray[register+1];

foreach (DataGridViewRow row in dgv\_dataHolder.Rows)

{

if (row.Cells[3].Value.ToString() == (register - 1).ToString() && row.Cells[1].Style.BackColor != Color.FromArgb(255, 110, 50, 50)) //Car le register est l'adresse + 1 donc il faut enlever 1

{

switch (row.Cells[2].Value.ToString())

{

case "U16":

value = TCPEquipement.Modbus\_Server.holdingRegisters.localArray[register];

if (value < 0)

{

value = 65535 + (value + 1);

}

value = Convert.ToUInt16(value);

row.Cells[1].Value = value;

tb\_Logs.Invoke(new MethodInvoker(delegate

{

logMessage("Ecriture reçu à l'adresse : " + (register - 1).ToString() + " | de valeur : " + value.ToString() + " | et de type : " + row.Cells[2].Value.ToString());

}));

break;

case "U32":

Int32 val = EasyModbus.ModbusClient.ConvertRegistersToInt(valueTwoRegisters);

UInt32 valUint32 = 0;

if (val < 0)

{

valUint32 = Convert.ToUInt32(4294967295L + (val + 1));

} else

{

valUint32 = Convert.ToUInt32(val);

}

row.Cells[1].Value = valUint32;

tb\_Logs.Invoke(new MethodInvoker(delegate

{

logMessage("Ecriture reçu à l'adresse : " + (register - 1).ToString() + " | de valeur : " + valUint32.ToString() + " | et de type : " + row.Cells[2].Value.ToString());

}));

break;

case "I16":

value = TCPEquipement.Modbus\_Server.holdingRegisters.localArray[register];

value = Convert.ToInt16(valueTwoRegisters[0]);

row.Cells[1].Value = value;

tb\_Logs.Invoke(new MethodInvoker(delegate

{

logMessage("Ecriture reçu à l'adresse : " + (register - 1).ToString() + " | de valeur : " + value.ToString() + " | et de type : " + row.Cells[2].Value.ToString());

}));

break;

case "I32":

value = EasyModbus.ModbusClient.ConvertRegistersToInt(valueTwoRegisters);

row.Cells[1].Value = value;

tb\_Logs.Invoke(new MethodInvoker(delegate

{

logMessage("Ecriture reçu à l'adresse : " + (register - 1).ToString() + " | de valeur : " + value.ToString() + " | et de type : " + row.Cells[2].Value.ToString());

}));

break;

case "Float":

valueFloat = EasyModbus.ModbusClient.ConvertRegistersToFloat(valueTwoRegisters);

row.Cells[1].Value = valueFloat;

tb\_Logs.Invoke(new MethodInvoker(delegate

{

logMessage("Ecriture reçu à l'adresse : " + (register - 1).ToString() + " | de valeur : " + valueFloat.ToString() + " | et de type : " + row.Cells[2].Value.ToString());

}));

break;

}

}

}

}

catch (Exception err)

{

logMessage("Erreur de lécture : " + err.Message);

}

}

Fonction lécture client des registres du serveur toute les (#ValeurPooling) \* millisecondes

private void Timer1\_Tick(object sender, EventArgs e)

{

if (TCPEquipement.Mode == "Client")

{

int[] registersTable = TCPEquipement.Modbus\_Client.ReadHoldingRegisters(0, AdresseMax + 30);

foreach (int num in registersTable)

{

Console.Write(num.ToString() + ";");

}

Console.WriteLine("");

Console.ReadLine();

foreach (DataGridViewRow row in dgv\_dataHolder.Rows)

{

if (row.Cells[1].Style.BackColor != Color.FromArgb(255, 110, 50, 50))

{

int adresse = Convert.ToInt32(row.Cells[3].Value);

string type = row.Cells[2].Value.ToString();

int value = 0;

Single valueFloat = 0;

int[] valueTwoRegisters = new int[2];

switch (type)

{

case "U16":

value = Convert.ToInt16(registersTable[adresse]);

if (value < 0)

{

value = 65535 + (value + 1);

}

value = Convert.ToUInt16(value);

row.Cells[1].Value = value;

break;

case "U32":

valueTwoRegisters[0] = registersTable[adresse];

valueTwoRegisters[1] = registersTable[adresse + 1];

value = EasyModbus.ModbusClient.ConvertRegistersToInt(valueTwoRegisters);

UInt32 valUint32 = 0;

if (value < 0)

{

valUint32 = Convert.ToUInt32(4294967295L + (value +1));

}

else

{

valUint32 = Convert.ToUInt32(value);

}

row.Cells[1].Value = valUint32;

break;

case "I16":

value = Convert.ToInt16(registersTable[adresse]);

row.Cells[1].Value = value;

break;

case "I32":

valueTwoRegisters[0] = registersTable[adresse];

valueTwoRegisters[1] = registersTable[adresse + 1];

value = EasyModbus.ModbusClient.ConvertRegistersToInt(valueTwoRegisters);

row.Cells[1].Value = value;

break;

case "Float":

valueTwoRegisters[0] = registersTable[adresse];

valueTwoRegisters[1] = registersTable[adresse + 1];

valueFloat = EasyModbus.ModbusClient.ConvertRegistersToFloat(valueTwoRegisters);

row.Cells[1].Value = valueFloat;

break;

}

}

}

}

} //Pour que le client lis les registers du serveur en fonction du pooling

Avec ceci de fait, nous avons presque terminé la communication entre des équipements Modbus TCP/IP, il reste quelque changement et peaufinage à faire puis il faudra faire pareil pour RTU et Profinet

# Outils utilisés

|  |  |
| --- | --- |
| Nom de l’outil | Fonction de l’outil |
| Visual Studio 2019 | Espace de développement avec multiple langage de programmations |
| VMWare Workstation 16 | VMware Workstation est un outil de virtualisation de poste de travail, il sert à mettre en place un environnement de test pour développer de nouveaux logiciels, ou pour tester l'architecture complexe d’un système d’exploitation avant de l’installer réellement sur une machine physique. |

# Conclusion.

Pour conclure sur cette semaine, j’ai fait je trouve un travail de malade, je ne me suis jamais senti aussi productif avec une découverte de nouvelle technologie, de fonctionnement et de matériel, j’ai repris depuis le début l’application et j’ai bien planifier mes taches avant de partir tête baisser.

Grace a ça j’ai pu avancer vite, vite corriger les bugs et les problèmes même en l’absence de mon tuteur dès le mardi.

Il reste encore plusieurs choses à faire et ceci risque d’être ma tâche principale jusqu’à la fin de mon stage si je ne la finis pas avant.

Sachant que cette application va surement être présenté par la suite à des personnes professionnelles, j’ai hâte de continué et de fournir un travail propre, fonctionnel et qui a de la finalité.

J’attend donc le retour de mon tuteur la semaine prochaine pour son avis sur l’avancement de l’application ainsi que les changements apportés pour pouvoir ensuite trouver un accord sur la continuité du développement de l’application.

# Niko-Niko.

|  |  |  |
| --- | --- | --- |
| JOUR | RESSENTI | TACHE |
| Lundi | Emoji Request - BitingNailsEmoji | Nouveau projet + Passage du permis |
| Mardi | 15,256 Confused Emoji Stock Photos, Pictures & Royalty-Free Images - iStock | Apprentissage du fonctionnement des Modbus |
| Mercredi | 👷 Construction Worker Emoji | Découverte du projet + étude de maquette de projet (Brainstorming sur mise en place de modifications et de fonctionnalités) |
| Jeudi |  | Création interface from scratch + Fonctionnement graphique + résultat épreuve permis (Obtenue) |
| Vendredi | Emoji De Style De Masque De Pirate | Vecteur Premium | Mise en place fonctionnement connexions et communications |