



Ingénierie & systèmes automatisés

COMPTE RENDU STAGE*Année 2023-2024***SEMAINE 3**

16/01/2023 – 20/01/2023

FONCTION	NOM
Étudiant stagiaire	Allan ESCOLANO
Représentante entreprise	Grégory MEUNIER
Tuteur en entreprise	Jean-François DANCKAERT
Tuteur professionnel	Jean-François DANCKAERT
Représentante Lycée	Patricia BUËR
Professeur chargé du suivi	David ROUMANET

Sommaire / Summary

Table des matières

1.	Présentation de l'entreprise.....	3
2.	Mes missions.....	4
1.1	Simulateur_CEOG.	4
1.1.1	Documents et maquette.....	Erreur ! Signet non défini.
1.1.2	Modifications graphiques.	Erreur ! Signet non défini.
1.1.3	Modifications code	Erreur ! Signet non défini.
3.	Outils utilisés.....	19
4.	Conclusion.....	20
5.	Niko-Niko.	21

1. Présentation de l'entreprise.



ICÔNE 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.



2. Mes missions.

1.1 Simulateur CEOG.

1.1.1. Connexions Modbus RTU.

En continuité de la semaine dernière, après avoir fait fonctionner la connexion et la communication grâce au Protocol Modbus TCP, il m'a été demandé de faire la même chose pour les communications Modbus RTU.

Pour cela, l'utilisation de la bibliothèque EasyModbus permet de créer ces connexions et communications de la même manière que celle pour Modbus TCP ce qui a rendu la tâche beaucoup plus simple.

Le seul problème, c'est qu'à défaut de matériel, il est impossible de tester la connexion ainsi que la communication car les connexions RTU se font par câbles (Master/Slave).

Malgré ce problème, tout devrait fonctionner car le fonctionnement reste le même que pour la communication TCP

1.1.2. Création de la classe.

La classe reste essentiellement la même sauf au niveau de la configuration de la liaison dans laquelle il faut renseigner les informations supplémentaires suivantes :

- UnitIdentifier
- SerialPort
- Baudrate
- Bits de parité
- Bits de stop

Ces informations sont renseignées dans le fichier de configuration sous ce format :

SerialPort|Baudrate|BitsDeParité|BitsDeStop|UnitIdentifier|Pooling|Port

Une fois ceci en tête, la classe reste la même que celle pour TCP avec quelques changements :

```
public class RTU_Class
{
    public string Nom;
    public string Machine;
    public string Protocol;
    public string Configuration_Liaison;
    public string Mode;
    public string Fichier_Echange;

    public byte ID;
    public string SerialPort;
    public int Pooling;
    public int Baudrate;
    public System.IO.Ports.Parity Parity;
    public System.IO.Ports.StopBits Stopbits;
    public int Port;

    public ModbusClient Modbus_Client;
    public ModbusServer Modbus_Server;

    public RTU_Form Equipement_Form;

    public string ConnexionStatus = "Disconnected";

    public RTU_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.ID = Convert.ToByte(Configuration_Liaison.Split('|')[4]);
this.SerialPort = Configuration_Liaison.Split('|')[0];
this.Pooling = Convert.ToInt32(Configuration_Liaison.Split('|')[5]);
this.Baudrate = Convert.ToInt32(Configuration_Liaison.Split('|')[1]);
this.Port = Convert.ToInt32(Configuration_Liaison.Split('|')[6]);
string parity = Configuration_Liaison.Split('|')[2];
switch (parity)
{
    case "0":
        this.Parity = System.IO.Ports.Parity.None;
        break;
    case "1":
        this.Parity = System.IO.Ports.Parity.Odd;
        break;

    case "2":
        this.Parity = System.IO.Ports.Parity.Even;
        break;

    case "3":
        this.Parity = System.IO.Ports.Parity.Mark;
        break;

    case "4":
        this.Parity = System.IO.Ports.Parity.Space;
        break;

}

string stopbits = Configuration_Liaison.Split('|')[3];
switch (stopbits)
{
    case "0":
        this.Stopbits = System.IO.Ports.StopBits.None;
        break;
    case "1":
        this.Stopbits = System.IO.Ports.StopBits.One;
        break;

    case "2":
        this.Stopbits = System.IO.Ports.StopBits.Two;
        break;

    case "3":
        this.Stopbits = System.IO.Ports.StopBits.OnePointFive;
        break;

}
}
}

```

Ensuite, la connexion, la communication et le fonctionnement de l'application reste la même de TCP à RTU donc sur ce plan la aucun changement n'est nécessaire.

1.1.3. Protocol et bibliothèque Profinet.

Et pour finir, le dernier type de communication demander est Profinet.

PROFINET est un standard de communication ouvert pour l'automatisation industrielle. Il a été créé par PI (PROFIBUS & PROFINET International) - l'organisation des utilisateurs PROFIBUS qui compte plus de 1200 membres - et développé par Siemens, Phoenix Contact, Molex et d'autres constructeurs. De par son ouverture et l'utilisation d'un média de communication standard (Ethernet), PROFINET permet l'utilisation de toutes marques de matériel. La première version de ce standard a été publiée en août 2001. La version courante est la version V2.2. PROFINET est normalisé CEI 61158 et CEI 61784. (Source : <https://fr.wikipedia.org/wiki/Profinet>)

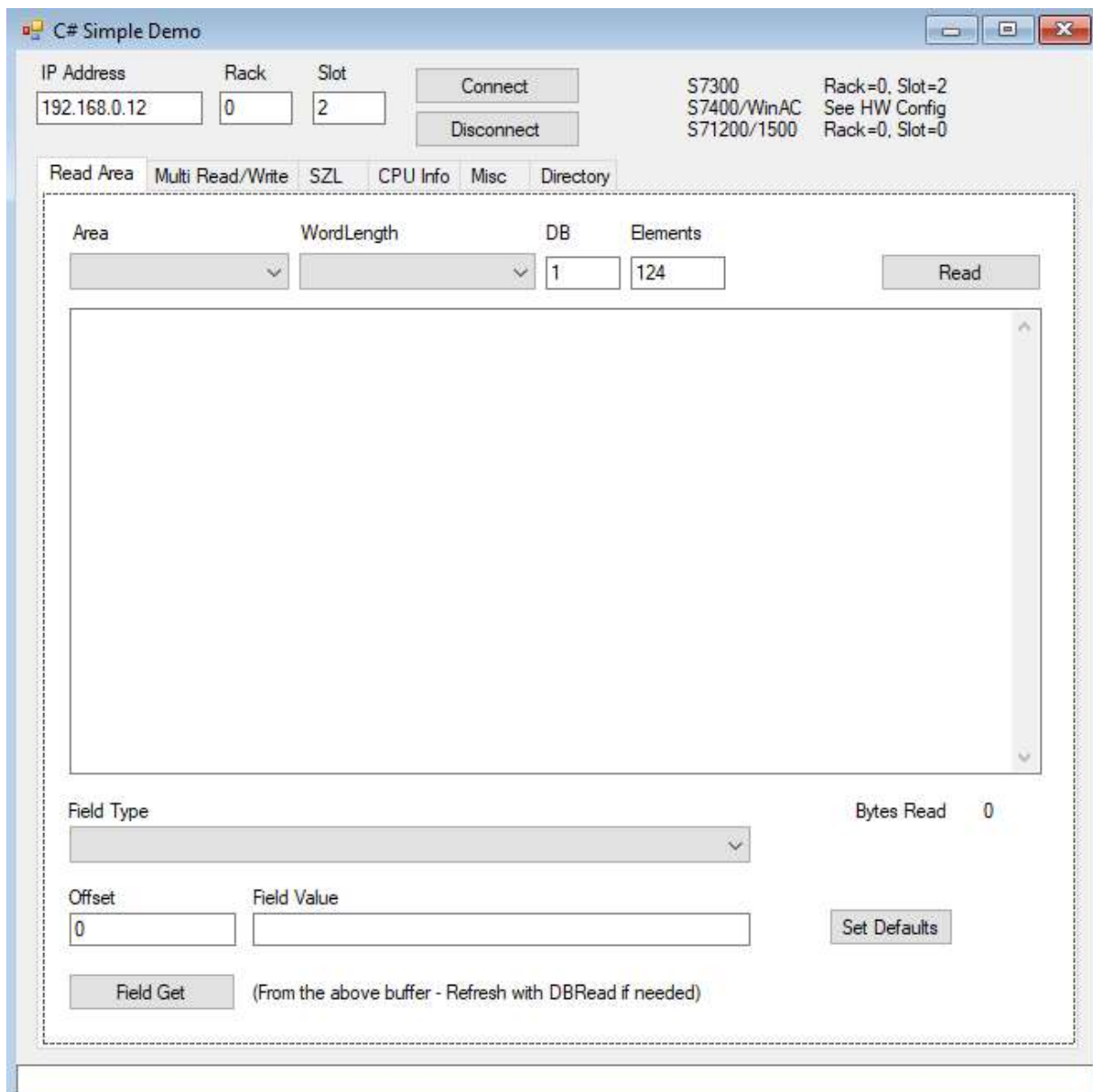
Pour ceci, l'implémentation d'une nouvelle bibliothèque est nécessaire. Après avoir fait des recherches, je suis tombé sur Sharp7, une adaptation de Snap7, une bibliothèque de communication par protocole S7Protocol (Protocole de communication développé par Siemens pour les automates), Pour C# (<https://snap7.sourceforge.net/sharp7.html>)

Pour C#, l'adaptation a été faite sous forme d'une classe Sharp7.cs contenant toutes les fonctions plutôt qu'une bibliothèque.

Avec ceci est fournis un fichier de documentation sur le fonctionnement ainsi que des exemples d'application fonctionnant avec la classe

1.1.4. Implémentation Sharp7.

J'ai donc intégré la classe (6000+ lignes) et l'interface qui se trouve dans l'exemple dans mon application :



Seulement, a cause d'un manque de communication des attentes de l'utilisateur concernant les communications Profinet, il en a donc été dit plus judicieux de ne pas continuer le développement de cette partie avant plus d'informations.

1.1.5. Documentation projet.

J'ai part la suite rédigé un fichier de documentation pour les utilisateurs de l'applications.

Il m'a été demande d'expliquer le but de l'application, comment la faire marcher, décrire les possibles erreurs et solutions ainsi que les prérequis a son bon fonctionnement.

Voici le sommaire du document :

Table des matières

1.	Contexte et cas d'utilisation.	2
1.1.	Protocoles de communications :	2
2.	Prérequis d'utilisation.	3
2.1.	Format fichier d'instance des équipements.	3
2.2.	Format fichier de configuration.....	5
3.	Fonctionnement de l'application.....	6
3.1.	Menu principal.	6
3.2.	Interface équipements.	7
3.3.	Interface équipement non connecté.....	8
3.4.	Interface équipement connecté.....	9

Le document explique :

- Comment remplir les fichiers des configurations des équipements CSV
- Comment remplir le fichier de configuration CSV
- Le fonctionnement des boutons et leurs fonctions
- Les erreurs possibles et leurs solutions
- Fonctionnement des communications et des protocoles

1.1.2. Normes de nommages et créations.

Pour créer les procédures stocker, il fallait suivre la norme de nommage suivante :

DbO.ICONNE_sp_action_table

Exemples :

Action	Table	Résultat
GET	Test	DbO.ICONNE_sp_get_test
UPDATE	Variable	DbO.ICONNE_sp_variable
DELETE	MrKoro	DbO.ICONNE_sp_MrKoro

Ensuite, pour la norme de création des procédures stocker, il faut impérativement renseigner en haut de la procédure des informations de cette manière :

```
-- =====
-- Author :      Prénom NOM
-- Description :  description de la procédure
-- Donnée d'entrée :
--      @Parametre1,
--      @Parametre2,
--      @...
-- Code erreur :
--      1 => Raison si la procédure retourne 1
--      2 => Raison si la procédure retourne 2
--      100 => Raison si la procédure retourne 100
--      ... => ...
-- =====
```

Cette norme permet à n'importe qui de reprendre la procédure stocker et la comprendre dans le plus court délai.

1.1.3. Table historisable.

Il m'a été par la suite demander de créer une table "Historisable" sous cette forme-là :

Variable_ID(clé étrangère)	Limite_haute	Limite_base	Bande_Morte
Voir table variable	Float (NOT NULL)	Float (NOT NULL)	Float (NOT NULL)

Puis sur cette table, j'ai dû faire un CREATE, GET et UPDATE (DELETE se fait en cascade sur variables)

Voici le CREATE :

```
USE [BASE_ICONECT]
GO
/***** Object:  StoredProcedure [dbo].[ICONE_sp_create_historisable]    Script Date:
18/01/2023 16:21:18 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      Allan ESCOLANO
-- Description:  Crée la ligne Historisable
-- Donnée d'entrée :
--      @ID int,
--      @Limite_haute float,
--      @Limite_base float,
--      @Bande_morte float,
-- Code erreur :
--      1 => Limite_haute ne peut pas etre NULL
--      2 => Limite_base ne peut pas etre NULL
--      3 => Bande_morte ne peut pas etre NULL
--      100 => Aucune variable ne contient l'ID renseigné
--      101 => La variable.historisable est a NULL
-- =====
ALTER PROCEDURE [dbo].[ICONE_sp_create_historisable]
    @ID int,
    @Limite_haute float,
    @Limite_base float,
    @Bande_morte float
AS
BEGIN
    -- on verifie si la variable existe
    if (SELECT COUNT(*) FROM dbo.Variables WHERE ID = @ID) = 0
    begin
        return 100;
```

```
end

-- on verifie que la variable.historisable est TRUE
declare @variable_historisable bit;
select @variable_historisable = historisable FROM dbo.Variables WHERE ID = @ID
if (@variable_historisable = 0 or @variable_historisable is null)
begin
    return 101;
end

-- on verifie que Limite_haute n'est pas NULL
if (@Limite_haute is null)
begin
    return 1;
end

-- on verifie que Limite_base n'est pas NULL
if (@Limite_base is null)
begin
    return 2;
end

-- on verifie que Bande_morte n'est pas NULL
if (@Bande_morte is null)
begin
    return 3;
end

begin try
    INSERT INTO dbo.Historisable (Variable_ID, Limite_haute, Limite_base,
Bande_morte)
    VALUES (@id, @Limite_haute, @Limite_base, @Bande_morte)
end try
begin catch
    return -1;
end catch

END
```

Ensuite le GET :

```
USE [BASE_ICONECT]
GO
/***** Object: StoredProcedure [dbo].[ICONE_sp_get_historisable]    Script Date:
18/01/2023 16:22:33 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:          Allan ESCOLANO
-- Description:      Recupere la ligne historisable
-- Donnée d'entrée :
--      @ID int
-- Code erreur :
--      100 => Aucune variable ne contient l'ID renseigné
--      101 => La variable.historisable est a NULL
--      102 => La variable ne contient pas de ligne historisable
-- =====
ALTER PROCEDURE [dbo].[ICONE_sp_get_historisable]
    @ID int
AS
BEGIN

    -- on verifie si la variable existe
    if (SELECT COUNT(*) FROM dbo.Variables WHERE ID = @ID) = 0
    begin
        return 100;
    end

    -- on verifie que la variable.historisable est TRUE
    declare @variable_historisable bit;
    select @variable_historisable = historisable FROM dbo.Variables WHERE ID = @ID
    if (@variable_historisable = 0 or @variable_historisable is null)
    begin
        return 101;
    end

    -- on verifie si la il y a une ligne historisable pour la variable
    if (SELECT COUNT(*) FROM dbo.Historisable WHERE Variable_ID = @ID) = 0
    begin
        return 102;
    end

    begin try
        SELECT Variable_ID, Limite_haute, Limite_base, Bande_morte FROM
dbo.Historisable
        WHERE Variable_ID = @ID
    end try
    begin catch
        return -1;
    end catch

END
```


Et pour finir pour la table historisable, le update :

```
USE [BASE_ICONECT]
GO
/***** Object: StoredProcedure [dbo].[ICONE_sp_update_historisable]    Script Date:
18/01/2023 16:23:23 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:          Allan ESCOLANO
-- Description:      Mettre a jour la ligne Historisable
-- Donnée d'entrée :
--      @ID int,
--      @Limite_haute float,
--      @Limite_base float,
--      @Bande_morte float,
-- Code erreur :
--      1 => Limite_haute ne peut pas etre NULL
--      2 => Limite_base ne peut pas etre NULL
--      3 => Bande_morte ne peut pas etre NULL
--      100 => Aucune variable ne contient l'ID renseigné
--      101 => La variable.historisable est a NULL
--      102 => La variable ne contient pas de ligne historisable
-- =====

ALTER PROCEDURE [dbo].[ICONE_sp_update_historisable]
    @ID int,
    @Limite_haute float,
    @Limite_base float,
    @Bande_morte float
AS
BEGIN

    -- on verifie si la variable existe
    if (SELECT COUNT(*) FROM dbo.Variables WHERE ID = @ID) = 0
    begin
        return 100;
    end

    -- on verifie que la variable.historisable est TRUE
    declare @variable_historisable bit;
    select @variable_historisable = historisable FROM dbo.Variables WHERE ID = @ID
    if (@variable_historisable = 0 or @variable_historisable is null)
    begin
        return 101;
    end

    -- on verifie si la variable existe
    if (SELECT COUNT(*) FROM dbo.Historisable WHERE Variable_ID = @ID) = 0
    begin
        return 102;
    end

    -- on verifie que Limite_haute n'est pas NULL
    if (@Limite_haute is null)
    begin
```

```
        return 1;
    end

    -- on verifie que Limite_base n'est pas NULL
    if (@Limite_base is null)
    begin
        return 2;
    end

    -- on verifie que Bande_morte n'est pas NULL
    if (@Bande_morte is null)
    begin
        return 3;
    end

    begin try
        UPDATE dbo.Historisable
        SET Limite_haute = @Limite_haute, Limite_base = @Limite_base, Bande_morte
= @Bande_morte
        WHERE Variable_ID = @ID
    end try
    begin catch
        return -1;
    end catch

END
```

3. 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.
Microsoft SQL Server	Microsoft SQL Server est un système de gestion de base de données en langage SQL incorporant entre autres un SGBDR développé et commercialisé par la société Microsoft.

4. Conclusion.



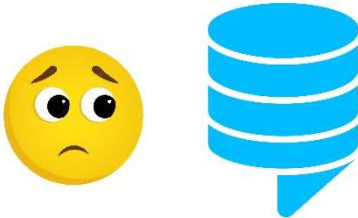


Cette semaine j'ai pu 'conclure' le projet Simulateur_CEOG tout en apprenant une nouvelle bibliothèque ainsi que la fonction de plusieurs types de Protocol et équipements, j'ai pu mettre mes compétences et ma réflexion à l'épreuve et produire un développement efficace qui m'a rendu fier.

J'ai aussi pu approfondir mes compétences en SQL tout en utilisant MICROSOFT SQL Server, sachant que SQL et les bases de données ne sont pas mes points forts, c'est rafraichissant de sortir un peu de C# et approfondir une de mes lacunes.

J'ai aussi l'impression de mieux m'y prendre niveau organisation et préparation avant le développement pur et dur, à force de m'être planté plusieurs fois par manque de plan de travail, j'y ai appris à utiliser les ressources disponibles pour faire un plan et avoir des réflexions plus claires sur les tâches et le projet en général.

Ceci est particulièrement utile en entreprise car la plupart des projets sont partagés entre plusieurs personnes et sont d'une taille bien plus grande que les projets faits en cours.

5. Niko-Niko.

JOUR	RESSENTI	TACHE
Lundi		Mise en place des communications RTU
Mardi		Découverte protocole Profinet + Sharp7 et implémentation. Documentation utilisateur de l'application
Mercredi		Etude diagramme base de données I-CONNECT + début modifications de la DB
Jeudi		Création de procédures stockées
Vendredi		Création de procédures stockées (Tuteur malade)