

**2011-2012**

Guillaume Leroy Emmanuel Yagapen

Polytech’Montpellier pour Synox représenté par Jérome Fenwick & Joffrey Verdier

Rapport Technique



Mise en place d’un service SMS

Remerciements

Nous tenons à remercier toutes les personnes qui nous ont permis de mener à bien ce projet pendant ces deux mois :

* M. Jérome Fenwick et M. Joffrey Verdier, de l’entreprise demandeuse Synox, pour leur disponibilité et leur aide au bon déroulement du projet
* Mme Michelle Cart, notre responsable pédagogique, qui nous a suivis régulièrement et conseillé tout au long de ce projet
* L’administration de Polytech’Montpellier pour nous avoir fourni le matériel nécessaire à la bonne réalisation de ce projet

Sommaire

[1 Introduction 5](#_Toc315792588)

[2 Présentation de l’environnement 6](#_Toc315792589)

[2.1 Le groupe SYNOX 6](#_Toc315792590)

[2.2 La plateforme Machine-to-Machine de gestion d’objets communicants 6](#_Toc315792591)

[3 Présentation du projet 7](#_Toc315792592)

[3.1 Le problème initial 7](#_Toc315792593)

[3.2 Les besoins fonctionnels 7](#_Toc315792594)

[3.3 La mission 7](#_Toc315792595)

[3.4 Contraintes 7](#_Toc315792596)

[3.4.1 Contraintes techniques 7](#_Toc315792597)

[3.4.2 Contraintes temporelles 8](#_Toc315792598)

[3.5 Ressources 8](#_Toc315792599)

[4 Déroulement du projet 9](#_Toc315792600)

[4.1 Gestion du projet 9](#_Toc315792601)

[4.2 Démarche 9](#_Toc315792602)

[4.2.1 Méthodes utilisées 9](#_Toc315792603)

[4.2.2 Choix technologiques 9](#_Toc315792604)

[4.3 Communication et synchronisation 10](#_Toc315792605)

[5 Travail réalisé 11](#_Toc315792606)

[5.1 Architecture du système d’envoi et de réception de SMS 11](#_Toc315792607)

[5.1.1 Le service SMS 12](#_Toc315792608)

[5.1.2 La base de données 12](#_Toc315792609)

[5.1.3 Interface utilisateur 12](#_Toc315792610)

[5.2 Conception 12](#_Toc315792611)

[5.2.1 Diagrammes des cas d’utilisation 12](#_Toc315792612)

[5.2.2 Diagramme de classe 13](#_Toc315792613)

[5.2.3 Maquettage 15](#_Toc315792614)

[5.3 Développement 15](#_Toc315792615)

[5.3.1 Les commandes AT 15](#_Toc315792616)

[5.3.2 Le mode PDU et la librairie ATSMS 15](#_Toc315792617)

[5.3.3 Le service SMS 17](#_Toc315792618)

[5.3.4 La base de données 18](#_Toc315792619)

[5.3.5 L’interface graphique 19](#_Toc315792620)

[5.4 Tests 21](#_Toc315792621)

[5.5 Limites 22](#_Toc315792622)

[6 Améliorations possibles 23](#_Toc315792623)

[6.1 Permettre une utilisation multi-utilisateurs 23](#_Toc315792624)

[6.2 Sécuriser l’accès à la base de données 23](#_Toc315792625)

[6.3 Implémenter une gestion des contacts 23](#_Toc315792626)

[7 Difficultés rencontrées 24](#_Toc315792627)

[7.1 Apprentissage des technologies 24](#_Toc315792628)

[7.2 Problèmes de compatibilité entre le Framework .Net et le JavaScript 24](#_Toc315792629)

[7.3 Les accusés de réception 24](#_Toc315792630)

[8 Conclusion 25](#_Toc315792631)

[9 Annexes 26](#_Toc315792632)

[9.1 Les commandes AT 26](#_Toc315792633)

[9.1.1 Choix du format des messages 26](#_Toc315792634)

[9.1.2 Choix de la mémoire 26](#_Toc315792635)

[9.1.3 Affichage des messages 26](#_Toc315792636)

[9.1.4 Envoi d’un message 26](#_Toc315792637)

[9.2 Envoyer et recevoir des données 26](#_Toc315792638)

[9.2.1 Connexion au modem 27](#_Toc315792639)

[9.2.2 Envoi de données au modem 27](#_Toc315792640)

[9.2.3 Réception de données 28](#_Toc315792641)

[9.2.4 Déconnexion du modem 28](#_Toc315792642)

[9.2.5 Exemple d’utilisation du protocole de communication 28](#_Toc315792643)

[9.3 Diagramme de Gantt 30](#_Toc315792644)

Table des images

[Figure 1 - Schéma d'architecture 11](#_Toc315791749)

[Figure 2 - Diagramme des cas d'utilisation : Application 13](#_Toc315791750)

[Figure 3 - Diagramme des cas d'utilisation : Utilisateur 13](#_Toc315791751)

[Figure 4 - Diagramme des classes 14](#_Toc315791752)

[Figure 5 - Informations nécessaires pour l’envoi d’un SMS en mode texte 15](#_Toc315791753)

[Figure 6 – Informations nécessaires pour l’envoi d’un SMS en mode PDU 16](#_Toc315791754)

[Figure 7 - SQL Server Management Studio : Structure de la table Message 19](#_Toc315791755)

[Figure 8 - Requête LINQ : SMS en attente d'accusé 19](#_Toc315791756)

[Figure 9 - Capture d'écran de l'interface d'envoi de SMS 20](#_Toc315791757)

[Figure 10 - Capture d'écran de la boîte d'envoi 20](#_Toc315791758)

[Figure 11 - Capture d'écran de la boîte de réception 21](#_Toc315791759)

[Figure 12 - Extrait de l'algorithme de connexion au modem GSM 27](#_Toc315791760)

[Figure 13 - Extrait de l'algorithme d'envoi de commandes au modem GSM 27](#_Toc315791761)

[Figure 14 - Algorithme de réception des messages dyu modem GSM 28](#_Toc315791762)

[Figure 15 - Protocole de communication entre le service SMS et le modem GSM 29](#_Toc315791763)

[Figure 12 - Diagramme de Gantt (tableau) 30](#_Toc315791764)

[Figure 13 - Diagramme de Gantt (schéma) 30](#_Toc315791765)

# Introduction

Dans le cadre de notre formation de 5ème année en école d’ingénieurs à Polytech’Montpellier, nous devons effectuer un projet industriel en collaboration avec une entreprise. Ce projet a pour but de nous placer dans les conditions du monde professionnel.

Dans cette optique, nous avons choisi de réaliser le projet industriel proposé par M. Fenwick, représentant de l’entreprise SYNOX, société de prestation de services informatiques aux entreprises.

Synox possède une plateforme permettant à des applications d’échanger des informations avec des *objets distants*[[1]](#endnote-1)à travers le réseau *GPRS*[[2]](#endnote-2). Cependant ce réseau présente des problèmes de fiabilité compromettant le bon fonctionnement de la plateforme. Le réseau *GSM*[[3]](#endnote-3)sur lequel s’appuie la technologie *SMS*[[4]](#endnote-4) étant plus fiable, il est alors nécessaire de s’appuyer dessus afin d’assurer une continuité des échanges.

La mission qui nous a été confiée consiste donc à développer un mécanisme **d’envoi et de réception de SMS qui s’intégrera à la plateforme existante**.

Notre choix s’est porté sur ce projet car ce travail porte sur des technologies en fort développement que sont les objets mobiles et le Web. De plus, cette mission comprend des aspects dans le domaine du réseau que nous souhaitions approfondir.

Le projet a débuté le 5 décembre pour prendre fin le 9 février, soit une durée de 10 semaines.

Ce rapport vise à expliquer de manière synthétique le travail accompli durant ce projet. Pour plus de détails, nous vous invitons à vous reporter au rapport technique.

Dans un premier temps nous présenterons l’entreprise et le contexte dans lequel s’inscrit notre mission. Puis nous aborderons les objectifs attendus et le travail réalisé pour les atteindre. Nous décrirons ensuite la démarche suivie et les problèmes que nous avons rencontrés. Enfin, nous ferons un bilan sur le travail que nous avons effectué.

# Présentation de l’environnement

## Le groupe SYNOX

Le groupe SYNOX est une société informatique spécialisée dans les solutions mobiles et collaborative. Son activité porte essentiellement sur le développement spécifique et la mise en place d’infrastructures mobiles.

En 2010 son chiffre d’affaires a atteint plus 2,5 millions d’euros porté par la bonne santé du **cloud computing.**

## La plateforme Machine-to-Machine de gestion d’objets communicants

Synox développe pour ses clients des applications communiquant avec des objets distants grâce au réseau GPRS. Cette communication est gérée par une plateforme Machine-to-Machine (**M2M**) centralisée qui donne accès aux clients à un suivi de leurs objets distants.

Schéma

On peut ainsi prendre l’exemple d’un autobus qui est équipé d’un capteur qui envoie un signal à la plateforme à chaque fois qu’il arrive à un arrêt. La plateforme se charge alors de retransmettre l’information à d’autres services comme par exemple une application mobile.

# Présentation du projet

## Le problème initial

Les objets de la plateforme M2M du groupe Synox utilise actuellement le réseau GPRS pour communiquer. Cependant, il se peut que ce réseau ne soit pas disponible à certains endroits et à certains moments, rendant impossible toute communication entre les objets distants et la plateforme.

On cherche donc **à résoudre un problème de** **fiabilité** lié au système actuel.

## Les besoins fonctionnels

Pour pallier ce problème, Synox a choisi d’utiliser le réseau GSM, beaucoup plus fiable, en permettant aux objets distants de communiquer avec la plateforme par SMS.

Synox a donc besoin de fournir à ces programmes la possibilité d’envoyer des SMS afin d’assurer une continuité du service en cas d’une panne du réseau GPRS.

Il faut donc que les objets distants, dotés en conséquence d’un *modem[[5]](#endnote-5)* GSM, puissent envoyer et recevoir des SMS. De même, la plateforme M2M doit être dotée des mêmes capacités.

La solution sera associée à une base de données enregistrant tous les messages, ainsi qu’à une interface graphique de gestion en ligne pour permettre aux utilisateurs de communiquer par SMS avec les objets distants.

## La mission

Notre mission consiste donc à concevoir et à développer les fonctionnalités suivantes du système d’envoi et de réception de SMS :

* Envoi et réception de SMS à travers un *service Windows[[6]](#endnote-6)*
* Stockage des messages envoyés et reçus dans une base de données
* Associer un statut aux SMS envoyés afin de suivre leur état
* Ecriture et consultation des messages depuis une interface graphique

L’interface graphique a pour objectif de permettre à un utilisateur d’avoir un accès à l’ensemble des fonctionnalités proposées par le système.

## Contraintes

### Contraintes techniques

Pour mener à bien ce projet, nous devons respecter plusieurs contraintes techniques. Tout d’abord, le projet doit être développé à l’aide du *framework .Net*[[7]](#endnote-7) en version 4.0 ou 3.5.

La base de données, afin de s’intégrer au framework .Net, doit être implémentée avec le système de gestion de base de données SQL Server 2008. Ces contraintes sur le framework sont dues au fait que la plateforme M2M déjà existante est développée avec cette technologie. Le Groupe Synox maintient ainsi une cohérence entre tous les éléments de cette plateforme.

Enfin, la liaison des données entre la base de données et le service ou l’interface web doit être effectuée en LINQ to SQL ou en ADO.NET.

### Contraintes temporelles

Nous disposions pour réaliser ce projet d’un délai de 9 semaines à compter du 5 Décembre 2011. Le planning prévisionnel a été établi comme suit dans la lettre de mission :

Le projet débute le lundi 5 Décembre 2011 pour prendre fin le vendredi 3 Février 2012 avec la livraison de la solution.

De plus, une soutenance de projet est prévue entre le 8 et le 10 Février 2012.

## Ressources

Pour mener à bien ce projet, nous avons constitué une équipe de deux élèves-ingénieur avec un planning de 30 jours.

Nous avons pu disposer d'une salle de projet fournie par Polytech’ Montpellier et équipée de d’un ordinateurs et de trois écrans. Nous avons également utilisé nos ordinateurs personnels.

M. Fenwick, notre demandeur, nous a également fourni un modem SMS muni d’une carte SIM.

# Déroulement du projet

## Gestion du projet

Le projet a commencé avec le premier entretien que nous avons eu avec le demandeur. À partir de cette discussion et des réponses qu’il nous a fournies, nous avons élaboré notre lettre de mission qui définit le champ de l’étude, le point d’arrivée et les modalités du projet.

Au cours de cet entretien, nous avons également découpé le projet en plusieurs phases.

La première phase consiste à mettre en œuvre l’envoi de SMS à l’aide du modem GSM. La communication avec le modem se faisant à l’aide d’instructions particulières (les **commandes AT**, aussi appelées **commandes Hayes**) via un port **COM**, il nous faut dans un premier temps étudier les possibilités de ce système et le prendre en main.

La deuxième phase consiste à traiter la réception des SMS et des accusés de réception.

Enfin, la troisième phase, la plus longue, consiste à concevoir et à réaliser le service Windows et l’interface graphique en ligne de gestion des messages.

Nous avons utilisé comme support de planification le logiciel Microsoft Project qui permet de planifier automatiquement les tâches en fonction des contraintes de début et de fin, qui prend en compte les jours fériés et les week-ends. Cet outil permet également de suivre le projet notamment grâce aux outils graphiques (diagramme de Gantt par exemple) qu’il met à disposition des utilisateurs.

Lors de l’évaluation des durées des tâches, nous avons dû prendre en compte le temps de prise en main du modem GSM et des commandes AT, ainsi que d’une productivité plus faible durant la période de Noël.

Vous trouverez en annexe le **diagramme de Gantt** du projet.

## Démarche

### Méthodes utilisées

Nous avons utilisé UML pour modéliser la base de données et écrire les cas d’utilisation. Nous n’avons pas utilisé d’autres méthodes particulières, le projet étant principalement concentré sur la conception et le développement.

### Choix technologiques

Les choix technologiques que nous avons à effectuer sont relativement restreints pas les contraintes du projet.

Afin de mettre en place la liaison entre le service, l’interface graphique et la base données, nous avions le choix entre deux bibliothèques logicielles, **LINQ to SQL** et **ADO.Net**. Nous avons synthétisé les avantages et inconvénients dans le tableau suivant :

|  |  |  |
| --- | --- | --- |
| Bibliothèque Logicielle | LINQ to SQL | ADO.Net |
| Mise en place | Rapide | Plus longue |
| Utilisation | Relativement aisée avec le langage LINQ | Nécessite un temps d’apprentissage |
| Perfomance | Problème avec des bases de données complexes | Très bonne |

Notre base de données n’étant pas complexe, nous avons décidé d’utiliser LINQ to SQL, plus rapide à mettre en place et plus facile à manipuler. De plus, nous avions déjà utilisé cette technologie dans un précédent projet.

## Communication et synchronisation

Afin de pouvoir partager rapidement le résultat de nos travaux, nous avons décidé **de réaliser ce projet de la façon la plus collaborative possible**.

Nous avons ainsi utilisé les outils **Google Documents** pour le partage de la documentation, **Github** pour le partage du code et de documents, et **Remember The Milk** pour le partage des tâches.

De plus la possibilité de disposer d’une salle de projet a été une excellente occasion pour travailler ensemble et communiquer directement pendant ces six semaines.

Concernant notre propre communication et synchronisation au sein du groupe, elle a pu se faire facilement puisque nous étions tous deux présents aux mêmes horaires dans une même salle. De plus, nous nous entendons bien.

Ainsi, nous ne nous sommes pas attribués de rôles fixes mais avons alterné. Cela dans le but de manipuler l’ensemble des technologies et de pouvoir prendre du recul.

# Travail réalisé

## Architecture du système d’envoi et de réception de SMS

**Guillaume**

**Placer les technologies, explications plus techniques**

L’architecture du système d’envoi et de réception de SMS peut être représentée par le schéma suivant :

**Plateforme M2M**

**Système d’envoi et de réception de SMS**

**Objets distants**

**Opérateurs**

**Télécoms**



**Utilisateur**

**Applications**

**Interface graphique**

**Base de données**

**Service SMS**

**Modem GSM**

**(SFR, Orange, Bouygues, Free)**

Figure - Schéma d'architecture

Le système d’envoi et réception de SMS est composé de trois parties : le service SMS, la base de données et interface graphique.

### Le service SMS

Le service a pour but d’envoyer et de réceptionner des messages afin de satisfaire les demandes des clients. Pour cela, il va contrôler le modem SMS à travers des commandes AT afin que ce dernier puisse envoyer et recevoir correctement des messages.

De plus, pour des questions de compatibilité avec la plateforme M2M de Synox, il a été décidé que le service SMS prendra la forme d’un service Windowsqui sera à l’écoute de la base de données afin de traiter les demandes de réception et d’envoi à travers un traitement par lot (batch).

### La base de données

La communication entre les clients et le service SMS s’effectue par l’intermédiaire d’une base de données.

En effet, lorsqu’un utilisateur de la plateforme veut envoyer un SMS, les données du nouveau message sont insérées dans la base de données. Ainsi, le service, qui est configuré pour vérifier régulièrement la base de données, va lire les informations relatives au message afin de transmettre les bonnes commandes d’envoi au modem.

Au niveau de la réception des SMS, le service interroge régulièrement le modem pour y récupérer des nouveaux messages afin de les insérer dans la base de données.

### Interface utilisateur

Cette interface permet aux clients d’envoyer et de recevoir des SMS en alimentant ou en consultant la base de données. Elle sera sous forme d’un site web d’administration qui permettra la visualisation des messages envoyés/reçus et elle inclura une interface pour envoyer le SMS avec les options possibles par le format PDU.

Il peut également être envisagé de créer un web Service pour que les applications Synox puissent s’y connecter simplement.

## Conception

Afin de procéder à la phase de conception, nous avons réalisé des **diagrammes de cas d’utilisation** et un **diagramme de classe.**

### Diagrammes des cas d’utilisation

#### Application

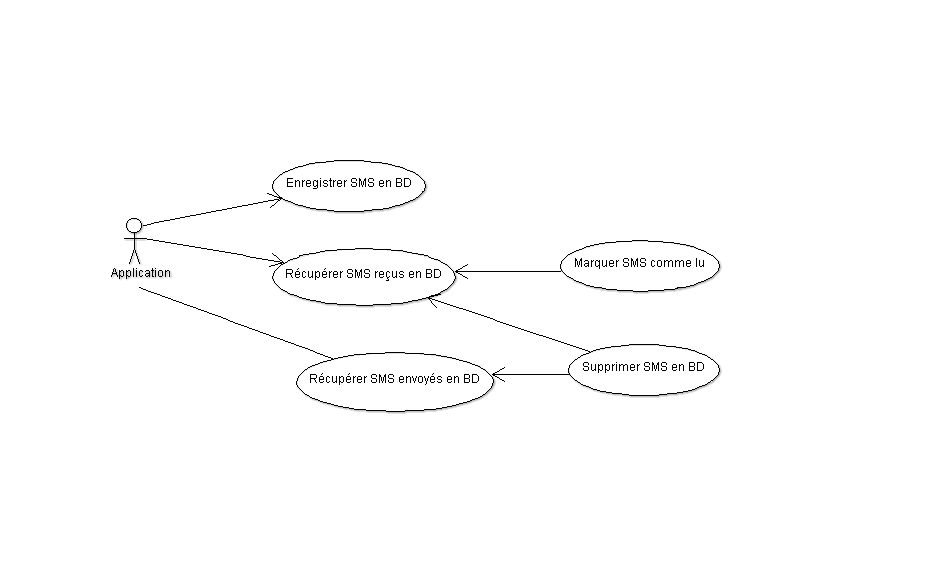


Figure - Diagramme des cas d'utilisation : Application

La communication entre le service chargé de l’envoi et la réception des messages avec les différentes applications se font par l’intermédiaire d’une base de données.

Ainsi, lorsqu’une application veut envoyer un SMS, elle insère toutes les informations concernant le message en base de données. De même, lorsqu’elle a besoin de récupérer les SMS reçus ou envoyés, elle consulte la base de données. L’application peut alors supprimer ou marquer comme lu les messages reçus.

#### F:\Internet\public_space\Polytech\Synox\GitHub\Synox\Docs\Cas utilisation\images\usecase_user.pngUtilisateur

Figure - Diagramme des cas d'utilisation : Utilisateur

Les utilisateurs auront la possibilité d’envoyer et de réceptionner des SMS par l’intermédiaire d’une interface graphique. La consultation des messages se compose de deux parties : les SMS envoyés et ceux reçus par le modem. L’utilisateur pourra les supprimer et marquer comme lu les messages réceptionnés.

### Dictionnaire de données

Après l’étude des besoins fonctionnels et des cas d’utilisation, nous avons pu déterminer de quelles informations nous avions réellement besoin pour le bon fonctionnement de notre application.

Le dictionnaire de données suivant présente l’ensemble de ces informations, regroupées par thème.

|  |  |  |  |
| --- | --- | --- | --- |
| Thème | Nom | Type | Commentaires |
| Message | idMessage | Entier | L'identifiant d'un message |
| messageTexte | Texte | Le contenu du message, s'il s'agit d'un message au format texte |
| messagePDU | Texte | La trame PDU envoyé |
| accuseReceptionDemande | Booléen | VRAI si un accusé est requis, FAUX sinon |
| noDestinataire | Texte | Le numéro du destinataire |
| noEmetteur | Texte | Le numéro de l'émetteur |
| MessageEnvoyé | dureeValidité | Entier | La durée de validité, convertie selon la norme PDU |
| dateDemande | Date | La date de soumission (insertion dans la BD) du message |
| dateEnvoi | Date | La date d'envoi du message |
| dateAccusé | Date | La date de réception de l'accusé |
| refModem | Entier | Le numéro de référence que le modem a associé au message |
| MessageReçu | dateRéception | Date | La date de réception du message |
| dateLecture | Date | La date de lecture du message |
| Encodage | idEncodage | Entier | L'identifiant de l'encodage |
| libelleEncodage | Texte | Le libellé de l'encodage (7bits, 8bits, 16bits) |
| Statut | idStatut | Entier | L'identifiant du statut de message |
| libelleStatut | Texte | Le libellé du statut de message (En attente, Envoyé, Accusé, Erreur) |

### Diagramme de classe

A partir du dictionnaire de données, nous avons élaboré le diagramme de classe. Les thèmes deviennent alors des classes, et chaque information d’un thème devient un attribut de la classe.

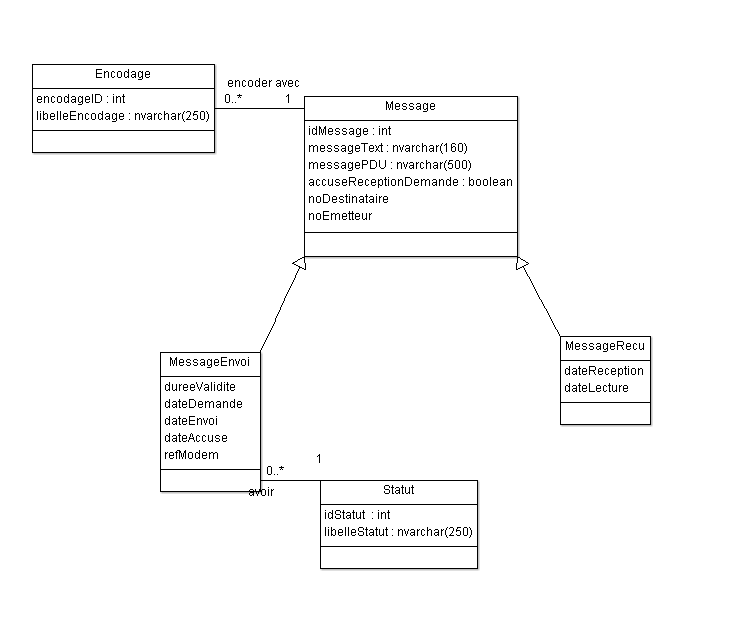


Figure - Diagramme des classes

La classe **Message** regroupe les informations partagées par les classes **MessageEnvoi** et **MessageRecu** qui en héritent. Chaque message est caractérisé par un identifiant unique, un destinataire, un émetteur et un message, qu’il soit au format texte ou PDU. Les numéros de téléphone sont enregistrés sous forme de chaînes de caractères. Il n’était pas possible de les enregistré au format numérique, car ils peuvent contenir le signe « + » au début pour l’indicateur géographique. L’attribut **accuseReceptionDemande** indique si le destinataire ou l’émetteur a demandé un accusé de réception du message au correspondant. Cet attribut devrait être un booléen, mais la base de données SQL Server ne propose pas ce type de données. Nous avons donc choisi d’utiliser un entier, 0 pour faux, 1 pour vrai. Toutes ces informations sont obligatoires (**messagePDU** ou **messageText** peut être nul, si l’autre ne l’est pas).

Un **message envoyé** est caractérisé par une durée de validité. La durée de validité est enregistrée sous forme d’entier, car c’est le format qu’il faut spécifier dans la trame PDU. La conversion est toujours possible en appliquant l’algorithme de conversion. Un destinataire ne reçoit pas un SMS dont la période de validité est expirée, ce contrôle étant effectué par l’opérateur de télécommunication. De plus, le fonctionnement du service SMS étant basé sur du traitement par lot, les messages ne sont pas envoyés immédiatement. L’attribut **dateDemande** renseigne donc le moment où l’utilisateur a validé le SMS à envoyer et **dateEnvoi** correspond alors au moment où le message est réellement envoyé par le service. Lorsqu’un SMS est envoyé par le service, une **référence** lui est assignée par le modem. Cette identifiant est le seul moyen de faire correspondre le message envoyé avec un potentiel accusé de réception.

Un **statut** est obligatoirement associé à chaque message envoyé. Cela permet de connaître le statut d’un SMS à tout moment, s’il est en attente d’envoi, s’il a été envoyé, s’il a été accusé ou s’il y a eu des erreurs.

Un **message reçu** est caractérisé par une date de réception et une date de lecture qui correspond au moment où le message est lu pour la première fois. La présence ou non de la date de lecture permet également de connaitre le statut du message reçu, lu ou non lu.

### Maquettage

**A réaliser et commenter Guillaume**

## Environnement de travail

### Le Framework .Net

**Guillaume**

Parler du principe de fonctionnement ASP.Net / C# / JavaScript

### Installation d’un serveur

#### SQL Server

**Guillaume**

#### VNC

Puisque nous ne disposions que d’un seul modem, il nous fallait trouver une solution pour pouvoir travailler à distance chacun de notre côté sans devoir emporter le modem (à cause des vacances de Décembre, notamment). Nous avons donc décidé d’installer sur le serveur une solution de prise de contrôle à distance, et avons choisi VNC.

Virtual Network Computing (VNC) est un système d'accès à un bureau distant qui permet de prendre le contrôle d'un ordinateur distant. Il permet de transmettre les saisies au clavier ainsi que les clics de souris d'un ordinateur à l'autre. VNC se compose de deux parties, le client et le serveur. Le serveur est l’ordinateur auquel on veut accéder à distance, alors que le client est l’ordinateur avec lequel on accède au serveur.

Nous avons donc installé RealVNC, une implémentation libre de VNC, sur le serveur dans la salle de projet. L’installation en elle-même est assez classique. Il a ensuite fallu ouvrir le port 5900, qu’utilise le protocole de VNC, sur le pare-feu du serveur.

La deuxième consiste à installer VNCViewer, le client VNC, sur nos propres ordinateurs. Après l’installation, la fenêtre suivante apparait pour la connexion au serveur :

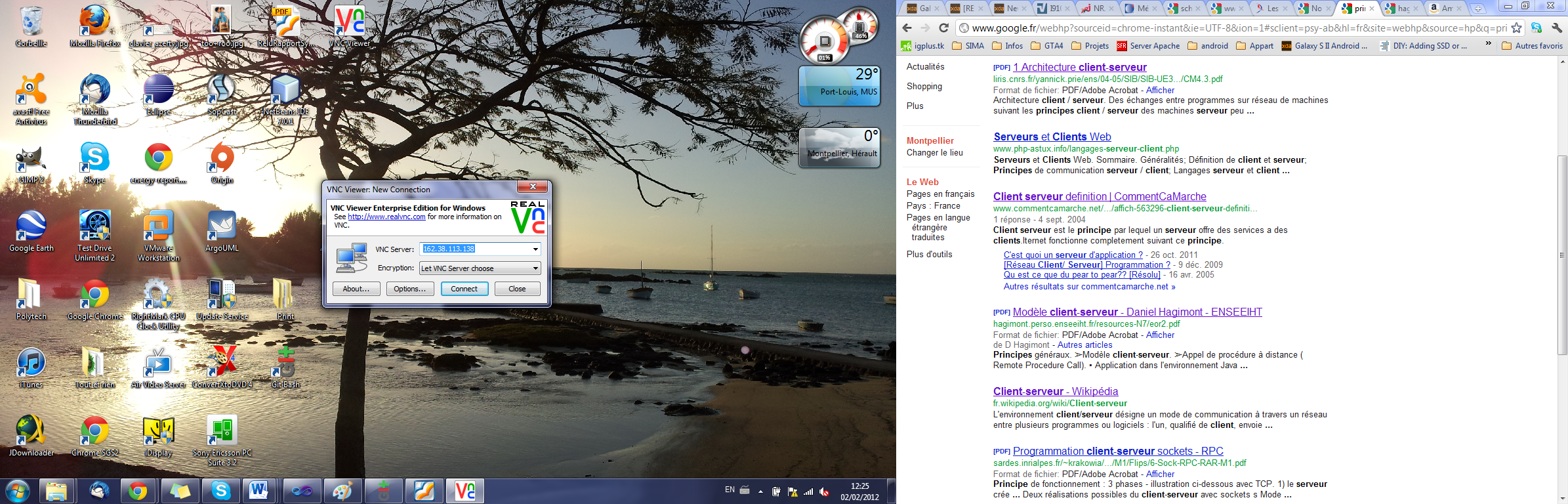


Figure - Fenêtre de connexion de VNC Viewer

Il faut alors renseigner l’adresse IP du serveur et valider. On accède alors au bureau du serveur.

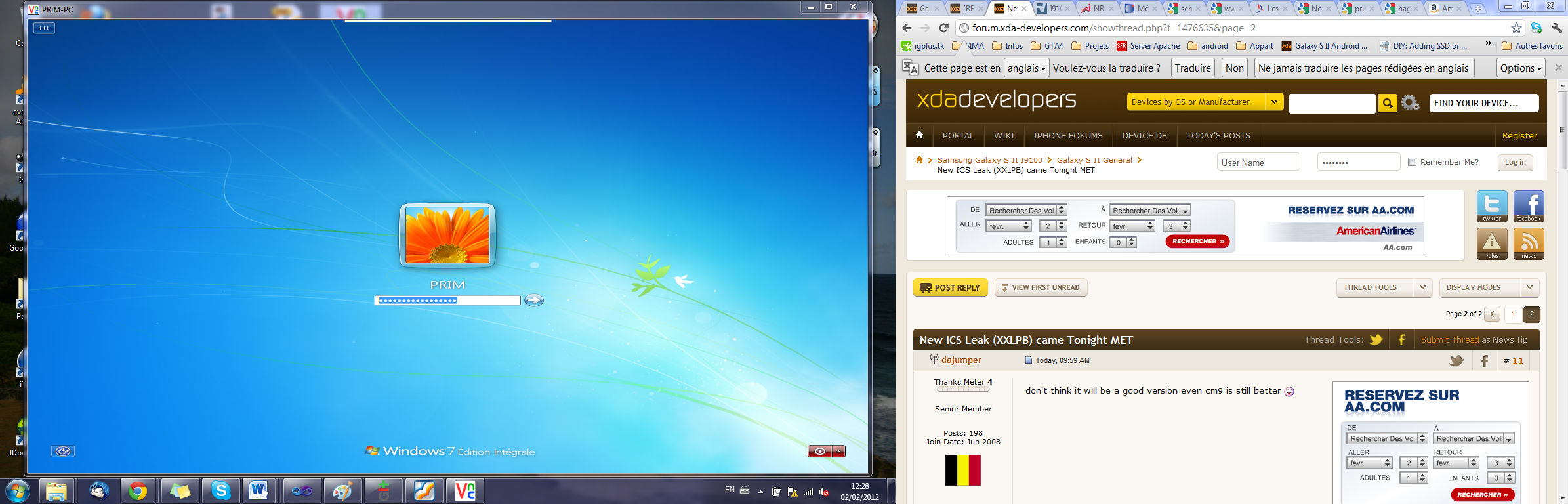


Figure - L'écran de connexion du serveur dans VNC Viewer

### Mise en place du modem GSM

La société Synox nous a fourni un modem GSM, modèle AirLink GL6100, pour toute la durée du projet. Un modem GSM se branche habituellement sur le port COM d’un ordinateur. Cependant, cette connectique est rarement présente sur les ordinateurs portables, et nous n’en disposions malheureusement pas sur nos propres machines. Nous avons donc utilisé un adaptateur USB (modèle ATEN UC232A), lui aussi fournit par Synox, qui nous permet de brancher le modem sur un port USB. Cet adaptateur nécessite un driver, que nous avons finalement trouvé sur le site du constructeur (ATEN).

Une fois tout ceci installé, le modem apparait dans la liste des périphériques, dans les périphériques branchés sur un port COM.

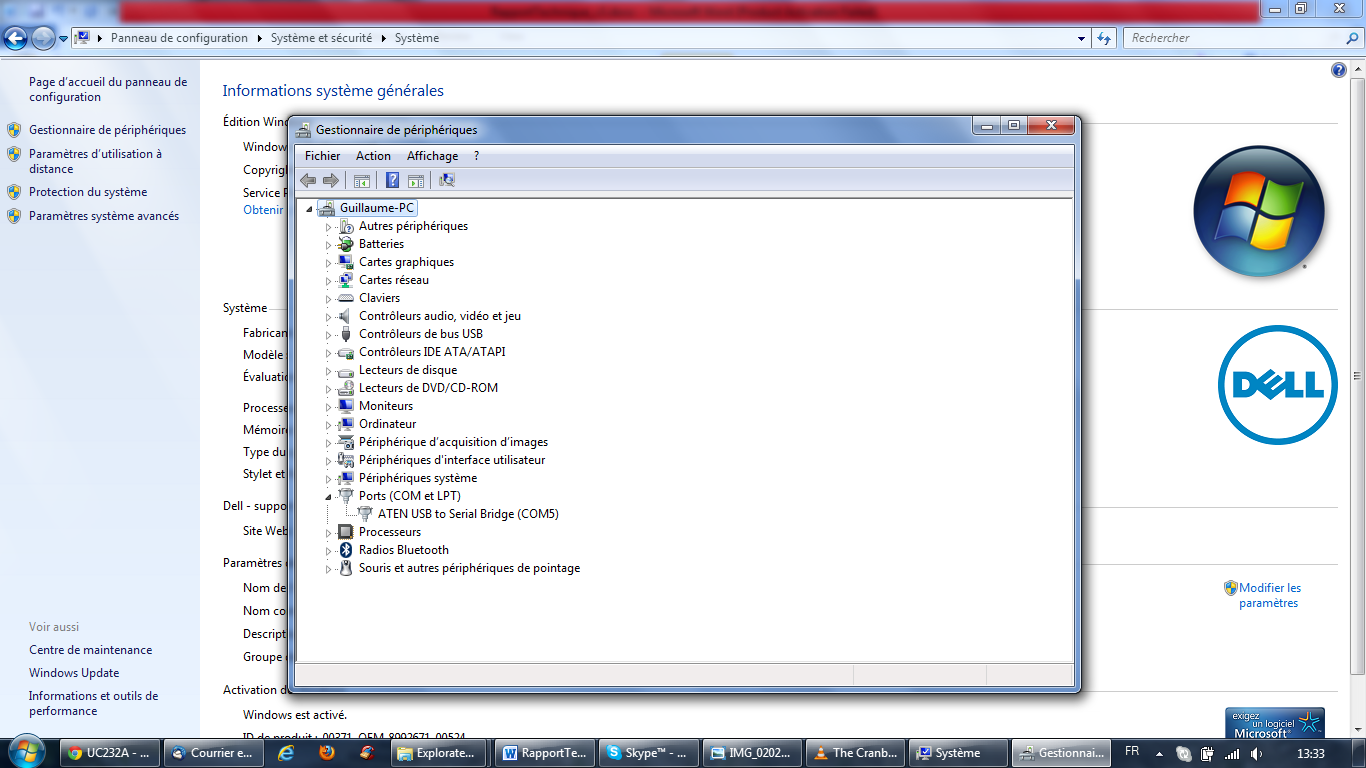


Figure - Le modem GSM dans le gestionnaire des périphériques

Il apparait le nom de l’adaptateur, mais il s’agit bien du modem GSM.

### GitHub

Afin de synchroniser notre travail, il nous fallait une solution de partage et d’échange de fichiers. Nous avons opté pour le logiciel de gestion de versions Git, réputé pour sa performance, et que nous souhaitions utilisé.

4 commandes permettent de synchroniser le travail, chacune précédée par le mot-clé « git ».

#### La commande add

Cette commande prend comme paramètre un chemin (relatif ou absolu). Elle marque les nouveaux fichiers à synchroniser, dans l’arborescence issue du nœud pointé par le chemin passé en paramètre.

La commande est couramment appelée comme suit, la console Git étant configuré pour pointer par défaut sur le répertoire à synchroniser.

**Git add .**

#### La commande pull

Cette commande permet de récupérer en local la dernière version sur le serveur. Si des modifications ont été faites depuis le dernier pull, il faut au préalable le signifier avec la commande commit, puis régler les éventuels conflits.

#### La commande commit

Cette commande crée une nouvelle version de l’espace de travail en prenant en compte les modifications faites depuis la dernière version. Cette commande prend en paramètre un message expliquant les modifications apportées.

**git commit –m « message »**

#### La commande push

Cette commande envoi sur le serveur la dernière version commitée.

### RememberTheMilk

**Guillaume**

Captures d’écran

## Développement

### Les commandes AT

**Guillinar**

#### Choix du format des messages

La commande **AT+CMGF** permet d’indiquer au modem le format des messages. En effet, il existe 2 formats possibles : le format texte, utilisant les caractères ASCII, et le format PDU, en hexadécimal (caractères de 0 à 9 et de A à F).

En mode texte, les informations sont directement compréhensibles par l’utilisateur, alors que le mode PDU nécessite un décodage. Par ailleurs, le mode PDU offre plus de possibilités que le mode texte.

#### Choix de la mémoire

La commande **AT+CPMS** permet de choisir la mémoire avec laquelle le modem communique. Il y a 3 types de mémoires :

* SM : Les messages stockés sur la carte SIM
* SR : La mémoire des accusés de réceptions, sur la carte SIM, limité à 50 accusés
* ME : La mémoire Flash

#### Affichage des messages

La commande **AT+CMGL** permet d’afficher tous les messages stockés sur la mémoire choisit avec la commande AT+CPMS. Ainsi, si la mémoire SM est sélectionnée, la commande AT+CMGL affiche les messages de la carte SIM. Un paramètre permet de spécifier l’état des messages demandés (reçus, lus et/ou non lus, tous).

#### Envoi d’un message

La commande **AT+CMGS** permet d’envoyer un message. La procédure change selon le mode choisit avec la commande AT+CMGF.

En mode texte, il faut d’abord spécifier le numéro du destinataire, puis entrer le message en langage « humain ». En mode PDU, on spécifie d’abord la longueur de la trame PDU, puis la trame PDU elle-même. Le numéro du destinataire, entre autres informations, est déjà compris dans la trame et sera utilisé par le modem.

### Envoyer et recevoir des données

Afin d’envoyer des commandes AT et de recevoir des réponses avec le modem, il faut mettre en place un protocole de communication.

Le modem SMS communique avec le service SMS par l’intermédiaire d’un **port série**. Nous avons ainsi utilisé des fonctions du Framework .Net qui nous permettent d’envoyer et de réceptionner des données simplement sur un port de communication.

Le protocole de communication que nous avons mis en place repose sur quatre fonctions :

* Connexion au modem
* Envoi de données
* Réception de données
* Déconnexion

Chaque commande AT envoyé au modem fait appel à ces quatre fonctions.

#### Connexion au modem

Avant de pouvoir envoyer des commandes AT au modem, il faut d’abord initialiser une connexion avec le modem. Pour cela, nous avons utilisé la classe **SerialPort** du Framework .Netqui nous permet d’instancier un objet représentant la liaison avec le modem.

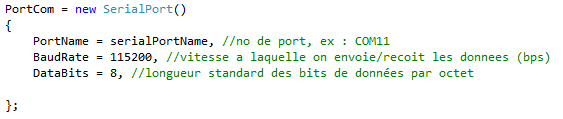


Figure - Extrait de l'algorithme de connexion au modem GSM

Lors de l’établissement de la connexion, nous devons spécifier le **numéro** du port utilisé, la **vitesse** à laquelle les données sont échangées et la **quantité** d’informations contenu dans chaque octet.

Une fois un objet de la classe **SerialPort** instancié, il suffit d’appeler la méthode «**Open()** » sur cette objet pour établir une connexion avec le modem.

#### Envoi de données au modem

L’envoi de données au modem se fait exclusivement à travers des **commandes AT**. Il faut qu’une connexion soit préalablement établie auprès du modem.

Pour transmettre des commandes, il suffit d’écrire des octets sur l'objet **SerialPort** à l’aide des données d’une **mémoire tampon**.

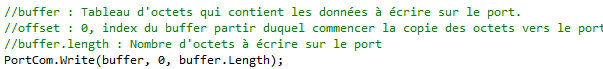


Figure - Extrait de l'algorithme d'envoi de commandes au modem GSM

Cette mémoire tampon permet d’éviter de saturer le modem si la commande de sortie est trop longue. En effet, si le nombre d'octets dans la mémoire tampon de sortie est trop important l'objet **SerialPort** va attendre que le périphérique soit prêt à accepter plus de données.

#### Réception de données

Pour recevoir des données du modem, il faut préalablement avoir envoyé des données au périphérique GSM. En effet, le modem les données envoyées par le modem correspondent uniquement à des réponses de **commandes AT**.

La réception de données s’effectue en lisant continuellement les données présentes sur la mémoire tampon du modem jusqu’à que le message reçu soit complet.

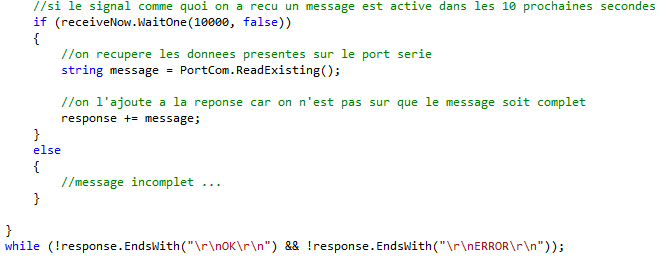


Figure - Algorithme de réception des messages dyu modem GSM

Un message est considéré comme complet lorsqu’il se termine par « **OK** » ou « **ERROR** » .

La fonction **receiveNow.Wait** retourne vrai si le modem reçoit des données avant que le **timeout** spécifié n’expire. Son utilisation a pour but d’éviter une attente active et également d’empêcher un blocage du programme en cas de réponse incomplète.

#### Déconnexion du modem

Lorsque le modem n’est plus utilisé par le service, il faut le déconnecter pour éventuellement permettre à d’autre programme d’utiliser cette ressource. Pour cela, il faut tout simplement appeler la fonction « **Close()** »sur l’objet **SerialPort** instancié lors de la connexion.

#### Exemple d’utilisation du protocole de communication

Le schéma ci-dessous montre un exemple de l’utilisation du protocole de communication lorsque la commande « **AT+CPMS ?**» est envoyée au modem. Cette commande permet de connaître le type de mémoire sur laquelle les messages sont stockés.

Service SMS

Modem GSM

1.Connexion

2. Envoi commande “**AT+CPMS?**”

3.Demande réception données

4.Réponse du modem : “**SM**”

5.Déconnexion

Figure - Protocole de communication entre le service SMS et le modem GSM

« **SM** » signifie que les messages sont stockés sur la carte SIM du modem.

### Le mode PDU

**Guillaume**

### La librairie ATSMS

**Guillaume**

Les fonctions de la librairie

### Le service SMS

**Guillinar**

Explications et extraits de l’algorithme principal

### Intégration de la base de données

**Guillinar**

Explication Linq

### L’interface graphique

**Guillaume**

#### L’envoi de SMS

#### La boîte d’envoi

#### La boîte de réception

## Tests

Tout au long du développement, nous avons effectué **plusieurs niveaux de tests**. En premier lieu, chaque fonction a été testée individuellement, afin de vérifier qu’elle retourne les résultats attendus, et qu’elle traite correctement les cas d’erreurs et les cas limites.

Nous avons également fait d’importants **tests de fiabilité et d’efficacité** sur le service SMS et le modem GSM. En effet, le service SMS est amené à devoir s’exécuter sur de longues durées sans intervention humaine possible en cas de problème. Il faut donc que le service SMS ne s’arrête pas inopinément pour une raison ou une autre, et doit redémarrer en cas de problème. Nous avons donc laissé le script s’exécuter des jours durant, ce qui nous a permis de mettre en évidence un problème récurrent : la communication entre la base de données est régulièrement rompue. Nous avons alors mis en place un **système relançant automatiquement la connexion et reprenant l’opération qui était en cours** quand le problème est survenu.

Nous avons réutilisé ce système de reprise sur erreurs pour tous les problèmes plus rares liés au réseau, comme les réponses incomplètes que renvoient parfois le modem.

Nous avons également voulu tester la capacité du modem à traiter de nombreux messages en même temps, que ce soit en réception ou en envoi, avec ou sans accusés de réception. C’est ainsi que nous avons mis en évidence les limites du modem, développées dans la partie 5.5.

L’interface graphique et les algorithmes liés ont aussi été longuement testés. Par exemple, en vérifiant le comportement de l’application lorsque l’utilisateur rentre des données aberrantes (numéro de téléphone contenant des lettres, durée de validité négative, etc.).

## Limites

Cette solution, telle que nous l’avons conçu selon les demandes du client et les contraintes matérielles, possède cependant certaines limites.

La première concerne l’interface. Celle-ci ne possède en effet aucun système de session et est donc mono-utilisateur. Si cette interface est amenée à être utilisée par plusieurs personnes, il serait alors indispensable de mettre en place un tel système, en n’affichant que les messages destinées et envoyées à ou par une certaine personne.

La seconde limite se situe au niveau du modem, qui n’est pas toujours capable de gérer correctement une importante quantité de messages. Il arrive régulièrement que certains messages soient envoyés en double, ou même jamais transmis au destinataire. Mais les problèmes les plus récurrents concernent les accusés de réception, qui n’arrivent que rarement lorsqu’ils sont nombreux à être attendus.

# Améliorations possibles

## Permettre une utilisation multi-utilisateurs

Le service SMS et la base de données ne permettent pas actuellement une utilisation avec plusieurs utilisateurs.

Pour implémenter une telle fonctionnalité, il faut mettre en place un système d’identifiants unique dans la base de données afin de pouvoir identifier les utilisateurs.

Au niveau du service SMS, il faut que les messages reçus par le modem puissent être identifiés et être redirigés vers les utilisateurs concernés. Pour cela, il faudrait que l’identifiant unique de l’utilisateur figure dans le contenu du SMS reçu.

## Sécuriser l’accès à la base de données

L’interface graphique communique directement avec la base de données pour y insérer les messages à envoyer et pour consulter les SMS. Pour rappel, l’interface créée n’est qu’un exemple afin de montrer les possibilités liées au service SMS. Cependant, il est aussi possible qu’une application puisse interagir directement avec la base de données pour envoyer ou recevoir des SMS.

Or, avec le modèle de fonctionnement actuel où les utilisateurs ont un accès total à la base de données, des problèmes peuvent survenir surtout en mode multi-utilisateurs. En effet, si les modifications de la base de données ne sont pas contrôlées cela peut créer des incohérences ou des pertes d’informations. Par exemple, actuellement, il n’y a rien qui empêche un utilisateur de supprimer tous les messages de la base de données, même ceux ne le concernant pas.

Afin d’éviter de telles problèmes, il faut mettre en place un **web service** qui garantit un accès à distance contrôlé à la base de données à travers des fonctions.

## Implémenter une gestion des contacts

Afin de pouvoir identifier rapidement l’auteur d’un message reçu, il serait intéressant d’implémenter une gestion des contacts par rapport à chaque utilisateur. Ainsi, l’interface utilisateur affichera le nom du contact à la place de son numéro de téléphone. Cette implémentation concerne surtout l’interface graphique et la base de données.

# Difficultés rencontrées

## Apprentissage des technologies

Pendant ce projet, nous avons été amenés à manipuler des technologies que nous ne connaissions pas.

Tout d’abord, il a fallu que nous comprenions l’intérêt et la syntaxe des différentes commandes AT afin de cerner celles que nous devions utiliser afin de communiquer avec le modem GSM.

De plus, même si nous avions déjà réalisé des projets avec le Framework .Net, nous n’avions jamais mis en place un protocole de communication avec cette technologie. De même, l’aspect Web de ce Framework nous était inconnu.

L’utilisation de ces différentes technologies a induit que nous devions assimiler rapidement leur fonctionnement afin d’être productif le plus tôt possible.

## Problèmes de compatibilité entre le Framework .Net et le JavaScript

Lors de la réalisation de l’interface graphique, nous avons été confrontés à plusieurs problèmes au niveau de la compatibilité entre la partie Web du Framework .Net et le *JavaScript[[8]](#endnote-8)*.

Ce problème de compatibilité vient surtout du fait que le Framework .Net possède ses propres composants Web, comme par exemple les champs de saisie ou les boutons. Leur utilisation facilite la gestion des évènements et la récupération des données depuis les champs de saisie.

Cependant, les composants du Framework .Net n’ont pas la même structure que les composants HTML. Or, le JavaScript est conçu pour fonctionner avec la structure des objets HTML.

Ainsi, dans certains cas le code JavaScript que nous avions écrit ne marchait pas. Par exemple, lorsque nous avons voulu améliorer le rendu visuel du formulaire d’envoi avec la librairie externe JQuery écrite en JavaScript, le gestionnaire d’erreurs nous indiquait que des éléments de la page étaient introuvables.

## Les accusés de réception

Pendant l’élaboration du service SMS, nous avons rencontré des difficultés avec la réception des accusés. En effet, il arrivait que modem GSM ne reçoive aucun accusés pendant quelques heures avant de reprendre un fonctionnement normal.

Après une longue série de tests nous nous sommes aperçus que le problème a deux origines distinctes. La première concerne une mauvaise configuration du modem GSM qui n’est alors pas configuré pour recevoir des accusés. La deuxième origine provient du fait que lorsque le modem reçoit un trop grand nombre d’accusés de réception, il arrive qu’il y ait une saturation et que certains accusés soient perdus.

# Conclusion

Ce projet de deux mois a été **riche en expériences** sur plusieurs niveaux. Tout d’abord, cette mission était pour nous un certain défi sur le plan technique. Nous n’avions en effet que peu d’expérience dans le domaine du réseau, et nous avons également eu l’occasion d’approfondir nos connaissances du Framework .Net.

De même, nous avons pu tester notre aptitude à être responsables de projet, c’est-à-dire estimer les charges des différentes phases, gérer nos plannings et prendre des décisions. C’est réellement la meilleure façon de comprendre un tel métier et d’apprendre de nos erreurs.

Par la suite, ce projet nous a également permis de développer notre aptitude à travailler en équipe. Nous avons donc pu **mettre en application** nos capacités d’écoute, d’ouverture d’esprit, de sens relationnel ainsi que d’organisation et d’analyse.

Nous pouvons dire que nous avons **atteint les objectifs définis** dans la lettre de mission. En outre, nous avons identifié quelques améliorations qu’il serait intéressant de mettre en œuvre afin de perfectionner le système.

Les technologies mobiles sont très demandées aujourd’hui avec la forte présence d’objets mobiles communicants comme les smartphones ou les machines embarquées. Il nous semble que ce domaine prend une place de plus en plus importante dans l’informatique, et qu’il sera inévitable pour nous, en tant que futurs ingénieurs en informatique, de s’y intéresser afin de suivre l’évolution technologique.

Finalement, que ce soit sur nos réussites ou nos difficultés, ce projet industriel reste une étape enrichissante dans cette dernière année de cycle d’ingénieur.

# Annexes

## Diagramme de Gantt

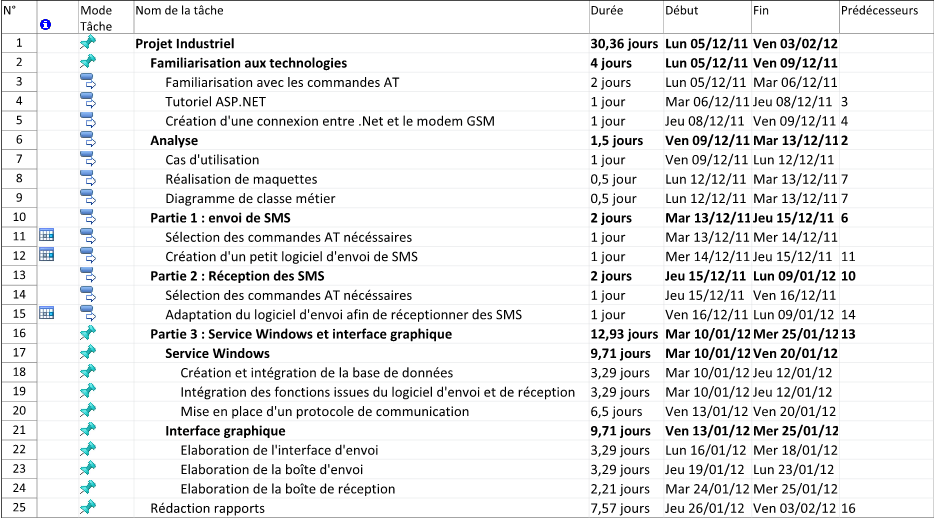


Figure - Diagramme de Gantt (tableau)

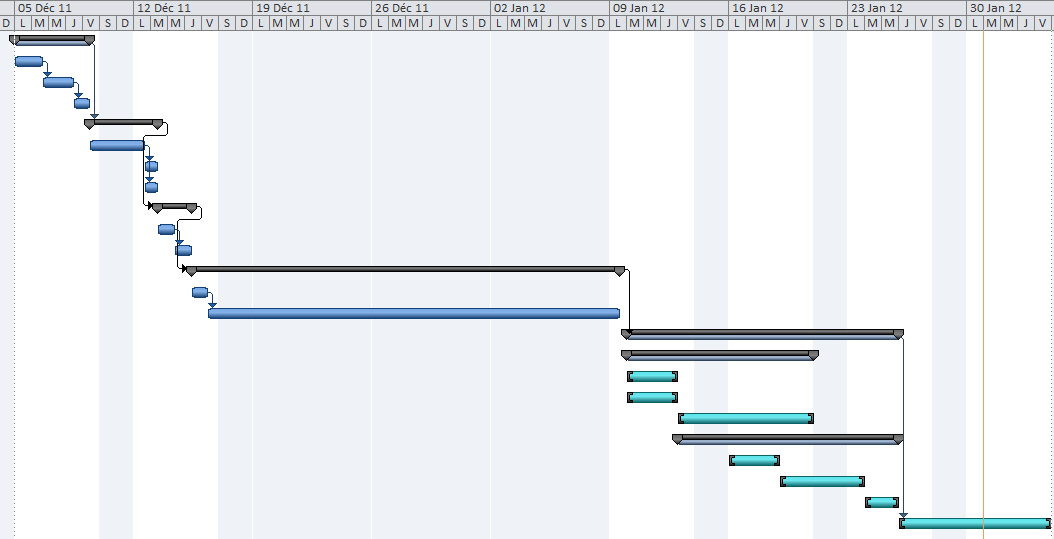


Figure - Diagramme de Gantt (schéma)

Glossaire

1. ***Objets distants :***

   Objets avec un système embarqué communiquant avec la plateforme. Ils envoient des informations qui seront interprétées par des applications Synox ou des utilisateurs. [↑](#endnote-ref-1)
2. [↑](#endnote-ref-2)
3. ***GSM – Global System for Mobile Communications :***

   Réseau de deuxième génération pour la téléphonie mobile. [↑](#endnote-ref-3)
4. ***SMS – Short Message Service :***

   Service permettant de transmettre de courts messages sur le réseau GSM. [↑](#endnote-ref-4)
5. ***Modem – Modulateur-démodulateur :***

   Périphérique servant à communiquer avec des utilisateurs distants. [↑](#endnote-ref-5)
6. ***Service Windows :***

   Logiciel s’exécutant au démarrage de l’ordinateur, en arrière-plan, ne nécessitant pas d’intervention de l’utilisateur. [↑](#endnote-ref-6)
7. ***Framework.Net :***

   Environnement de travail Microsoft, fournissant des fonctions permettant de concevoir des applications lourdes ou légères. Cet environnement possède un compilateur intégré afin de pouvoir tester et débugger ces applications. [↑](#endnote-ref-7)
8. ***JavaScript :***

   Langage de script qui permet des interactions avec les différents objets d’une page Web [↑](#endnote-ref-8)