PARTIE DE L’ETUDIANT B - LUDWIG ALEXANDRE

Table des matières

[Présentation du matériel utilisé 32](#_Toc420446693)

[Présentation du XBEE 32](#_Toc420446694)

[Présentation du calculateur PANDA 33](#_Toc420446695)

[Tests unitaires 33](#_Toc420446696)

[Premier test unitaire : Communication entre les deux modules Xbee 33](#_Toc420446697)

[Deuxième test unitaire : Intégration du module émetteur sur le calculateur PANDA 37](#_Toc420446698)

[Troisième test unitaire : Créer une interface graphique pour le PC de l’apiculteur 38](#_Toc420446699)

[Application finale 41](#_Toc420446700)

[Conclusion 47](#_Toc420446701)

# Présentation du matériel utilisé

## Présentation du XBEE

Un module XBee est un circuit de communication sans-fil utilisant les **protocoles 802.15.4 et/ou Zigbee**, permettant de réaliser différents montages, d’une liaison série RS232 classique à un réseau maillé (mesh) auto-configuré.

Un composant XBee série 1 coûte désormais **moins de 20 euros**.

Selon les modèles et les pays, ils utilisent la bande des 2,4 gigahertz (comme le Wifi ou le Bluetooth) ou les 900 MHz. Selon leur puissance, ils émettent à une distance comprise entre 30 mètres et 1,5 kilomètre.

Ils sont développés à l’origine par la société Maxstream, devenue Digi. Vous pouvez toujours trouver en vente les premières séries du module marquée Maxstream.

Le débit peut atteindre 250 kbps, mais si on les utilise pour réaliser une liaison série sans fil, les débits standards sont compris entre 9600 bps à 38400 bps.

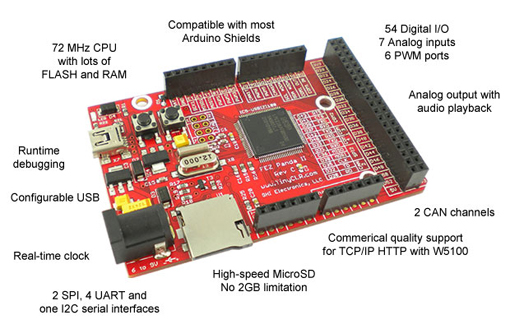
Ce module possède des cartes de tests et un logiciel de configuration (XCTU) afin de faciliter son utilisation.

Le XBee possède trois modes : **TRANSPARENT**, **COMMAND** et **API**.

* Le mode TRANSPARENT est le mode par défaut à la mise en marche du module, celui qui reçoit et envoie les données.
* Le mode COMMAND permet de configurer le module, ses entrées, ses sorties, son adresse, l'adresse de destination de ses messages, etc.
* Le mode API est un peu plus compliqué, il faut fabriquer ces propres trames et les envoyer au module. Ce mode est plus puissant et rapide mais nécessite plus de programmation et un firmware de type API.

Ces modes sont exploitables via X-CTU ou bien HyperTerminal.

## Présentation du calculateur PANDA



La platine FEZ "PANDA II" est probablement le système de développement embarqué (capable d'être programmé sous environnement Microsoft™ .NET Micro Framework™) parmi le plus petit et le plus économique du marché.

Se présentant sous la forme d'une petite platine, le FEZ "PANDA II" bénéficie d'une librairie de fonctions étendues lui permettant de gérer très facilement des entrées/sorties tout ou rien, des entrées de conversion analogique/numérique, une sortie analogique, des ports de communication mais également la gestion de cartes mémoires SD™.

Un connecteur mini USB permet de relier le module à un PC afin de pouvoir télécharger et debugger les applications

Des extensions pour la carte PANDA sont disponibles, comme par exemple un afficheur tactile.

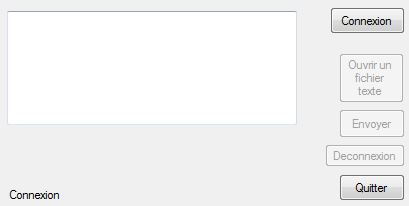
Nous utilisons cette carte car elle est compatible avec les cartes ARDUINO, permet d’accueillir un capteur de poids, de température et notre module Xbee sans fil !

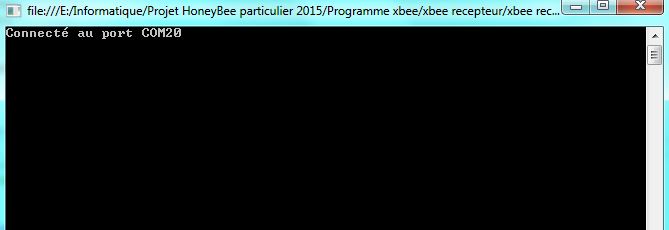
# Tests unitaires

## Premier test unitaire : Communication entre les deux modules Xbee

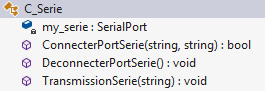
Afin de pouvoir faire communiquer les deux modules Xbee, j’ai créé deux applications : Une pour le module émetteur, et une pour le module récepteur.

L’application Xbee émetteur est une application Windows forms codée en c# :

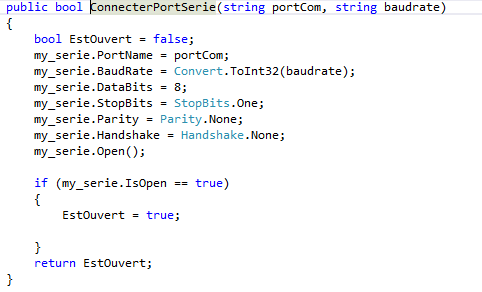


Tandis que l’application Xbee récepteur est une application console codée en c# :

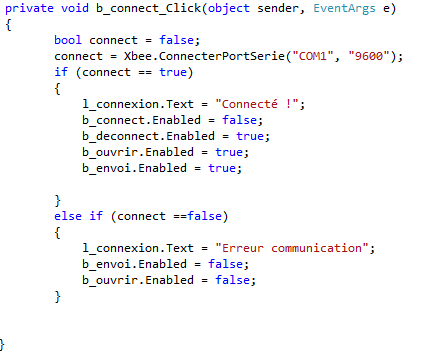
L’application émettrice possède une classe C\_Serie qui contient les méthodes ConnecterPortSerie, TransmissionSerie, DeconnecterPortSerie, et une instance de type SerialPort nommée « my\_serie » :



La méthode ConnecterPortSerie permet d’ouvrir le port série auquel le Xbee est connecté (COM1) :



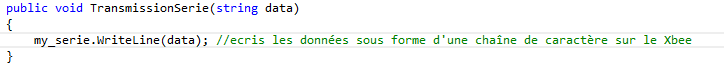
Cette méthode est appelée lors du clique du bouton « Connexion » de l’interface graphique :



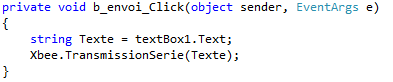
Nous devrions avoir ce résultat similaire :



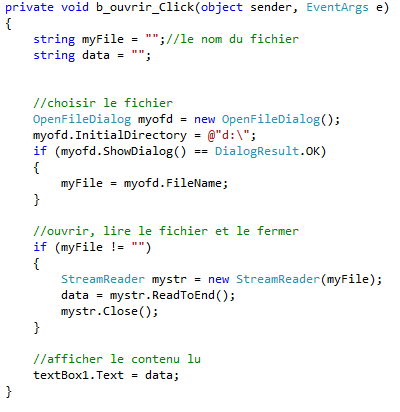
Désormais, nous pouvons entrer un texte à envoyer au récepteur dans la zone de texte, puis l’envoyer grâce au bouton « Envoyer » qui appelle la méthode « TransmissionSerie » :



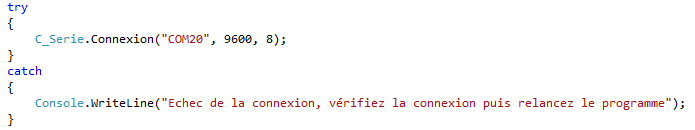
Code du bouton « envoyer » :



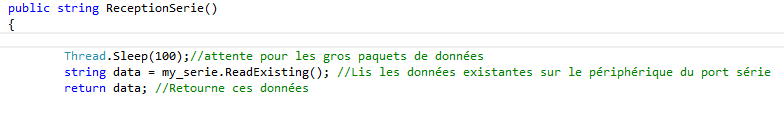
Nous pouvons également envoyer le contenu d’un fichier texte (.txt) grâce au bouton « Ouvrir un fichier texte », dont le code est ci-dessous :



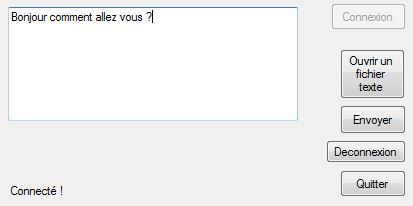
Du côté de l’application réceptrice, le contenu du xbee est analysé en permanence grâce à une boucle while(true) et grace à la méthode ReceptionSerie, avec un intervalle de 1 seconde :

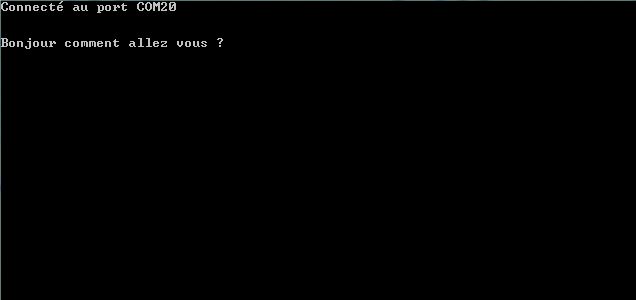


Méthode ReceptionSerie :



**Remarque : Le Xbee récepteur, malgré le fait qu’il soit connecté en USB, est traité de la même manière qu’un module série, grâce à un pilote qui permet de générer un port série correspondant au port USB utilisé par ce module (Digi USB Driver) . Pour tous les programmes, et pour le poste de l’apiculteur, le port affilié au Xbee récepteur sera toujours le port COM20. Le numéro de port est paramétrable dans le gestionnaire de périphériques du PC.**

Les modules communiquent parfaitement ensemble et quasiment instantanément :

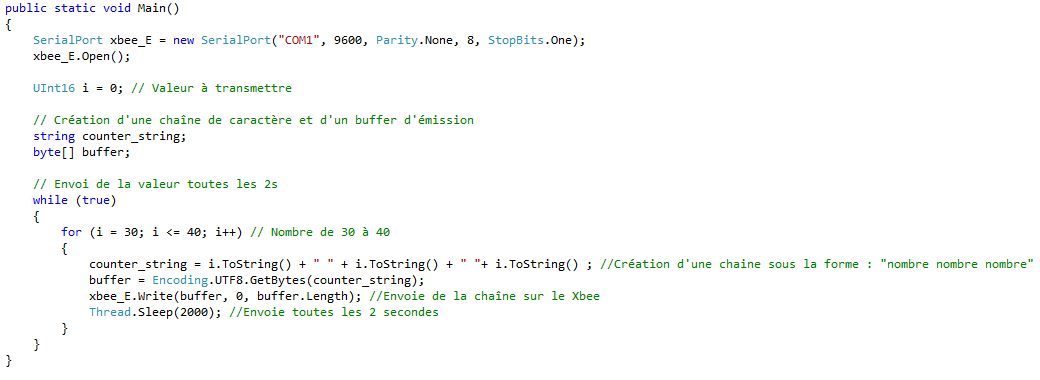


## Deuxième test unitaire : Intégration du module émetteur sur le calculateur PANDA

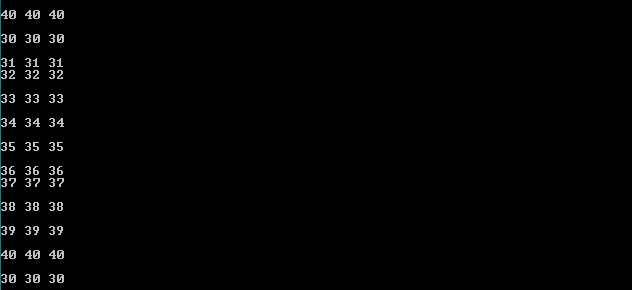
Pour le test précédent, le module Xbee a été directement connecté sur le port série du PC, il sera cependant connecté au calculateur panda lors du déploiement du système.

Pour s’assurer de la bonne communication des deux modules avec le calculateur PANDA pour l’émetteur, j’ai créé un programme sur le calculateur PANDA qui envoie 3 nombres identiques séparés par des espaces, qui pourrait correspondre par exemple au poids de la ruche, à la température de la ruche, et au niveau de batterie. Ces nombres seront compris entre 30 et 40, et augmentent au fur et à mesure des envois.

Cette application sera une application de type MicroFramework.



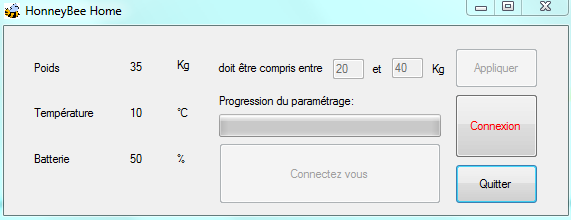
Apres exécution du programme, coté application réceptrice, on obtient ceci :



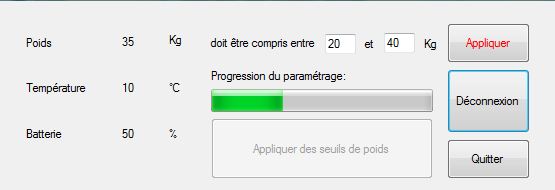
## Troisième test unitaire : Créer une interface graphique pour le PC de l’apiculteur

L’apiculteur, n’étant pas informaticien, doit avoir sur son PC une application simple à utiliser, possédant toutes les fonctionnalités que l’apiculteur recherche. J’ai donc mis au point une application graphique pour le poste du client, permettant la lecture des données en temps réel, une gestion de seuils de poids, et une gestion d’alerte en cas de batterie faible. L’application a été conçue dans le but de ne jamais pouvoir faire de mauvaises manipulations de la part de l’apiculteur, et de pouvoir le guider afin qu’il puisse obtenir les données qu’il désire recevoir.

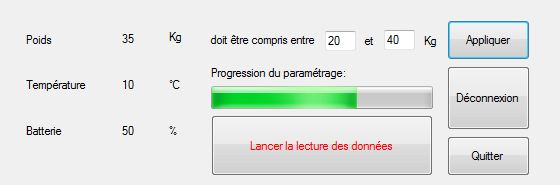
Lors du lancement de l’application, le client voit ceci :

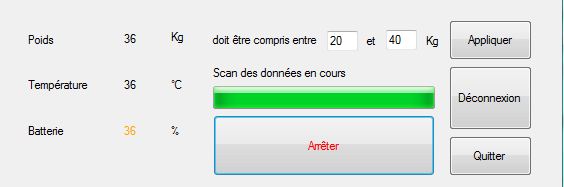


Lorsque le client se connecte, il peut désormais entrer des seuils de poids maximum et minimum pour sa ruche :

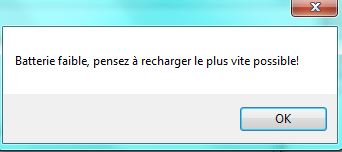


Apres avoir choisi les limites de poids, le client peut enfin lancer la lecture des données, et peut changer les seuils de poids à tout moment :



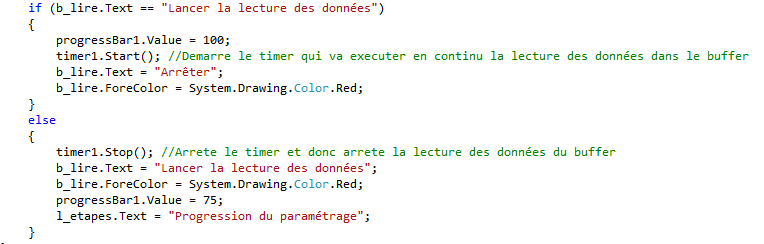


Cependant, lorsque le seuil de poids maximum est dépassé, le poids minimum est dépassé, ou la batterie est à 20% ou moins, l’utilisateur reçoit ces alertes :

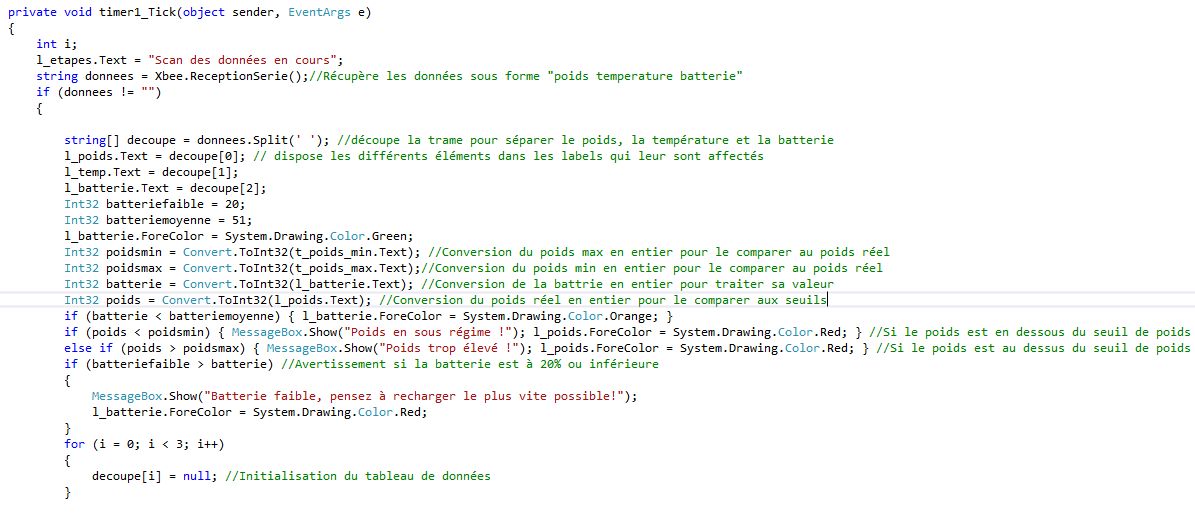


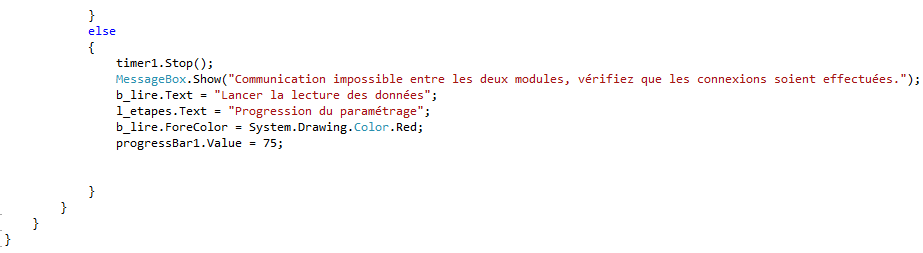
Le code de ce programme contient une classe avec les méthodes identiques des programmes précédents. Cependant, la lecture et l’analyse des données reçues se font grâce à un « timer » qui se lance lors du clic du bouton « lancer la lecture des données », et s’arrête en cliquant à nouveau sur ce même bouton. Ce timer lit et analyse les données toutes les 500 millisecondes, mais cet intervalle peut être modifié.

Code du bouton :

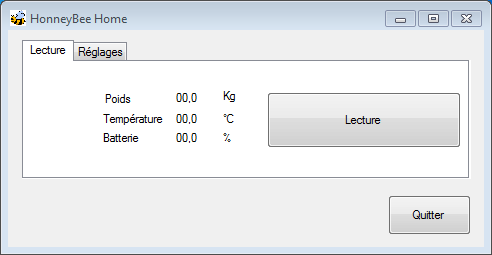


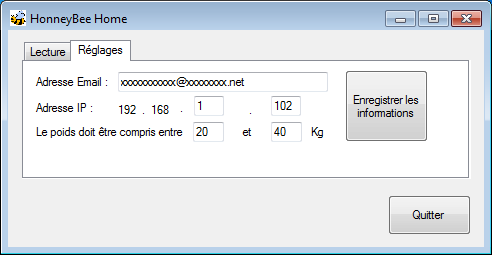
Code du timer :





# Application finale



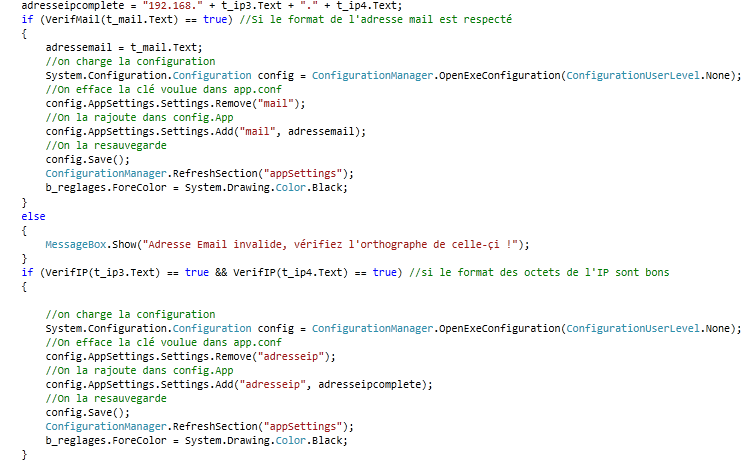


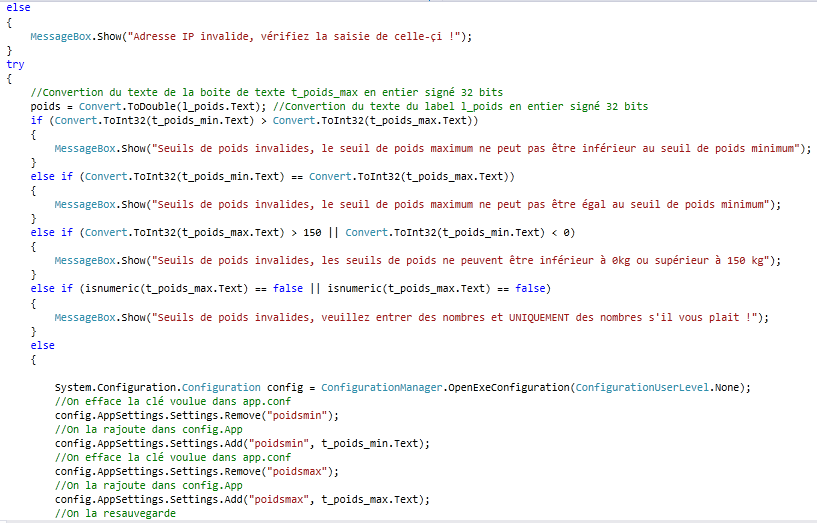
L’application finale contient toutes les fonctionnalités imposées dans le cahier des charges. Elle permet à l’apiculteur de recevoir les mails d’alerte depuis son adresse mail, recevoir les données depuis son smartphone deux fois par jour (12 h et 21h).

Contrairement à l’application du troisième test unitaire, cette application peut enregistrer les modifications des réglages (mail, IP, seuils de poids) afin que l’apiculteur puisse rentrer les informations qu’une seule fois. Les dernières modifications seront appliquées et affichés lors des prochains démarrages de l’application.

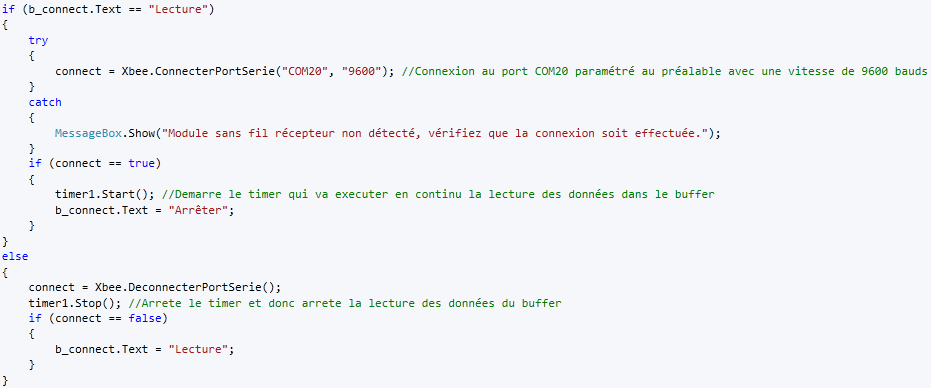
En appuyant sur le bouton « Enregistrer les informations », les informations précédentes seront effacées du ficher app.config du programme et remplacées par les nouvelles. Bien évidemment, si les informations sont incorrects comme des seuils de poids invalides ou une faute de frappe dans l’adresse ip ou mail, les modifications ne seront pas enregistrées.

Code du bouton « Enregistrer les informations » :





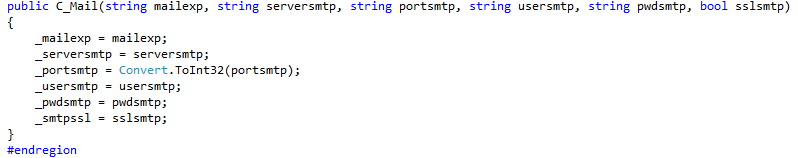


De plus, le nombre de boutons a été réduit. Le bouton « Lecture » permet au programme de s’assurer de la bonne connexion entre le module et le PC, et permet de lire les données si la connexion est effectuée. Ce bouton exécute les tâches du bouton « lecture des données » et « Connexion » de l’application du test unitaire :

Les mails sont envoyés à l’adresse mail rentrée par l’utilisateur et les données à l’adresse IP du téléphone de l’utilisateur. Un mail est envoyé en cas de seuil de poids dépassé ou de batterie faible par session, afin que l’utilisateur ne se fasse pas spammer de mails. Les données sont envoyées à 12h et 21h (heure du PC).

Une classe C\_Mail a été créée afin de gérer l’envoi des mails. Cette classe contient un constructeur et une méthode « Envoi ».

Constructeur :



mailexp correspond à l’adresse mail de l’expéditeur.

serversmtp correspond au serveur SMTP utilisé pour envoyer notre mail (dans notre cas, nous utiliserons celui de gmail).

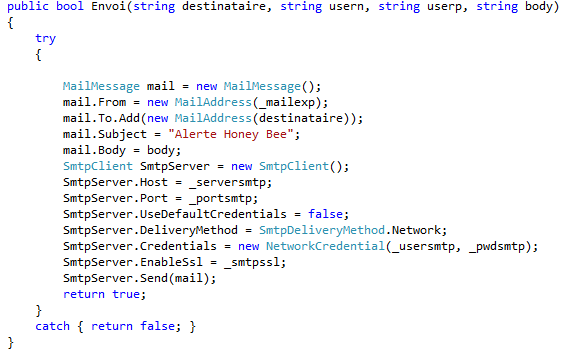
portsmtp correspond au port du serveur SMTP utilisé (port 587).

usersmtp correspond au nom d’utilisateur SMTP (identique à l’adresse mail de l’expediteur).

pwdsmtp correspond au mot de passe de l’adresse mail de l’expéditeur.

Sslsmtp correspond à l’état (activé ou désactivé) du SSL, un protocole de sécurisation des échanges sur Internet. Il sera désactivé dans notre cas.

Méthode « Envoi » :



Destinataire correspond à l’adresse mail du destinataire.

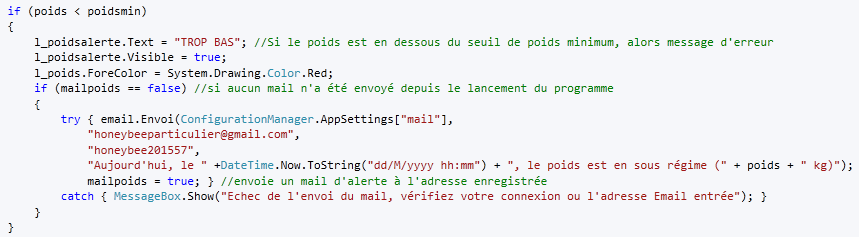
Usern correspond à l’adresse mail de l’expéditeur.

Userp correspond au mot de passe de l’adresse mail de l’expéditeur.

Body correspond au contenu texte du mail.

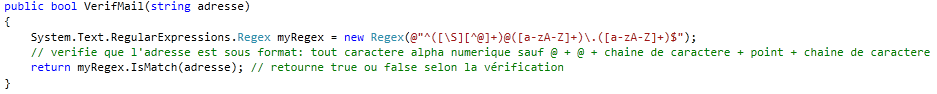
Le Credential permet de définir les informations d’authentification à utiliser.

Exemple d’envoi de Mail en cas de poids inférieur au seuil de poids minimum :



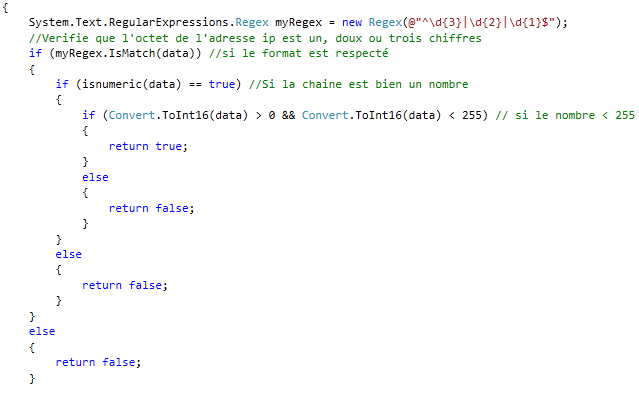
Pour cette application, nous utilisons désormais des Regex. Regex est une classe qui permet de vérifier qu’une chaine de caractères correspond à un modèle prédéfini, et est indispensable pour s’assurer que les réglages rentrés par l’utilisateur, et les données reçues par le module récepteur soient corrects et correspondent à ce que le programme exige.

Méthode de vérification de l’adresse mail :

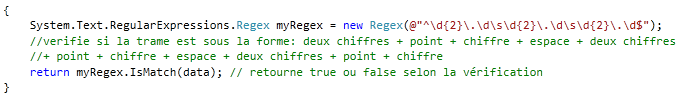


La méthode retourne « True » si le format et respecté, et « False » si le format n’est pas respecté.

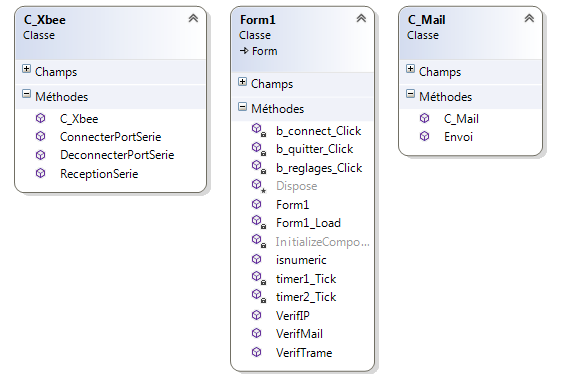
Méthode de vérification d’un octet d’adresse IP :



Vérification du bon format des données reçues :



Nous obtenons donc au final ce diagramme de classe :



# Conclusion

Ce projet a été pour moi très bénéfique. Il m’a permis de travailler en équipe et de pouvoir gérer mon temps, afin que le projet puisse être déployé dans les temps impartis. Ce projet m’a permis d’enrichir mes connaissances dans le domaine de la programmation en c#, ou j’ai dû à de nombreuses reprises chercher les solutions par moi-même et m’aider à devenir de plus en plus autonome.