République Tunisienne Ministère de l'Enseignement Supérieur et de la RechercheScientifique

Université de Tunis El Manar

Institut Supérieur d'Informatique

Rapport de Projet de Fin d'Études

Présenté en vue de l'obtention du

Diplôme National d'Ingénieur en Sciences Appliquées et Technologiques

Spécialité : Génie Informatique des Systèmes Industriels

Par

Mohamed Amine BARRAK

Développement de notification et de communication du système d'alarme

Réalisé au sein de Comelit Recherche
Et Développement

Encadrant à l'entreprise : Monsieur Walid BARREH

Monsieur Haithem Sekri

Encadrant à l'ISI : Madame Hanene BEN FRADJ

Année universitaire : 2015-2016

Dédicaces

A ma chère mère, source de tendresse et d'amour. Tu n'as pas cessé de m'encourager et de prier pour moi. Que ce travail soit témoignage de ma reconnaissance.

A mon cher père, qui m'a toujours soutenu dans les moments difficiles. Ce travail est le fruit des sacrifices que tu as consentis pour mon éducation et ma formation.

A mes deux sœurs et mon frère, les mots ne suffisent guère pour exprimer l'attachement, l'amour et l'affection que je porte pour vous.

Que dieu vous accorde santé et prospérité.

Remerciements

Au terme de ce travail, je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage et qui m'ont aidé lors de la rédaction de ce rapport.

Tout d'abord, j'adresse mes remerciements à Monsieur Lassaad JEMAI, Directeur de la société Comelit, qui m'a offert l'opportunité de faire mon stage de PFE. Je remercie également mes encadrants Monsieur Walid BARREH, Chef de projet à Comelit, et Monsieur Haithem Sekri, Ingénieur à Comelit, pour leurs conseils précieux et leurs remarques constructives. Un grand merci aussi à toute l'équipe de Comelit, de m'avoir bien accueilli et de m'avoir incité à mener à bien ce travail.

Je tiens à exprimer ma gratitude et ma reconnaissance à mon encadrante à l'ISI Madame Hanene Ben Fradj, Maître Assistante à l'ISI, pour ses directives et ses conseils judicieux.

Mes remerciements les plus distingués sont adressés aux membres du jury qui m'ont fait l'honneur de bien vouloir accepter d'évaluer ce travail.

Enfin, je profite de cette opportunité pour exprimer mes remerciements et mon profond respect à tous les enseignants de l'Institut Supérieur d'Informatique, ainsi qu'à tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail.

Table de matières

Liste des figures

Liste des tableaux

Liste des abréviations

ADC: Analog Digital Converter.

ARC: Alarm Receiving Center.

APN: Access Point Name.

API: Application Programming Interface.

DTMF: Dual Tone Multi Frequency.

GCM: Google Cloud Messaging.

HTTP: Hyper Text Transfer Protocol.

HTTPS: Hyper Text Transfer Protocol Secure.

MMS: Multimedia Messaging Service.

PDP: Packet Data Protocol.

RAM: Random Access Memory.

SMS: Short Message Service.

SMTP: Simple Mail Transfer Protocol.

SIM: Subscriber Identity Module.

TCP: Transmission Control Protocol.

UART: Universal Asynchronous Receiver Transmitter.

UDP: User Datagram protocol.

Introduction générale

Le cambriolage touche tous les types d'habitat, à n'importe quel moment de la journée. Plusieurs systèmes de sécurité ont été développés ces dernières années pour palier à la recrudescence des cambriolages et des tentatives d'effraction. En effet, le marché de sécurité est en pleine essor et propose divers types de produits : portes blindées, vitres anti-effraction, système d'alarme, télésurveillance, etc [1].

La transmission temps réel de signaux d'alarmes devient primordiale dans tout système de sécurité et offre une grande flexibilité pour le contrôle et la surveillance à distance. Actuellement, le GSM (Global System for Mobile Communication), système de la deuxième génération de la communication sans fil cellulaire, constitue le communicateur principal permettant la connexion permanente des utilisateurs à leurs systèmes de sécurité. Le GSM offre des services permettant à un utilisateur, disposant d'un système d'alarme, de communiquer, de paramétrer et de piloter son système à distance par l'envoi d'une télécommande par SMS (Short Message Service).

Avec l'avènement de l'Internet et le développement de nouvelles technologies mobiles, plusieurs services ont été créés pour apporter de l'innovation aux systèmes de sécurité. En effet, un utilisateur disposant d'une connexion Internet et d'un smart phone pourrait profiter des services disponibles sur le serveur GCM (Google Cloud Messaging) comme le service push notification permettant de l'alerter d'un événement (incendie, intrusion, etc) durant son absence quel que soit l'endroit où il est.

Ainsi le développement d'un système de sécurité efficace comportant plusieurs entités est une tâche d'une grande envergure. C'est dans ce cadre que s'inscrit notre projet de fin d'études qui s'est déroulé à l'entreprise CRED (Comelit Research and Development Tunisia) spécialiste dans le développement et la fabrication de systèmes en vidéophonie, domotique, vidéosurveillance, et de systèmes d'alarmes et de contrôle d'accès. Nous nous sommes intéressés en particulier au développement et la mise en œuvre d'un système de communication et de notification pour un système d'alarme. Notre travail a porté sur la conception et le développement d'un driver 3G permettant la gestion et la configuration d'un module de transmission de données 3G pour un système d'alarme ainsi que le développement de services contrôle SMS et push notification assurant le contrôle à distance de ce système.

Les étapes de réalisation de notre projet sont présentées d'une façon détaillée dans ce rapport qui sera organisé comme suit : Nous aborderons notre rapport par un premier chapitre

qui sera consacré à la présentation de l'organisme d'accueil, du contexte de notre projet, de l'environnement de développement et du cahier des charges. Dans le deuxième chapitre, nous présenterons les étapes de conception et de développement d'un driver pour un module 3G et du service contrôle SMS. Le dernier chapitre nous détaillerons la conception et le développement de l'application push notification pour les téléphones du type Android. Nous clôturerons notre rapport par une conclusion générale et des perspectives.

Chapitre 1

Présentation du cadre du stage

1.1 Introduction

Le but de ce chapitre est de présenter le cadre général de notre stage. La première partie de ce chapitre est consacrée à la présentation de la société d'accueil et de ses domaines d'activité. Nous présentons également l'environnement de travail pour réaliser le produit alarme. A la fin de ce chapitre, nous citons le cahier des charges ainsi que la démarche de conduite du projet afin de mener le travail demandé.

1.2 Présentation de l'entreprise d'accueil

La société Comelit R&D est une société italienne renommée mondialement avec plus de 50 ans d'expérience. La technologie et les systèmes de sécurité sont des composants essentiels de l'expérience de Comelit. Au fil du temps, Comelit a acquis un savoir-faire de haut niveau dans les systèmes de vidéophonie, de vidéo surveillance, d'alarme anti-intrusion, de domotique et de contrôle des accès [2].

La branche de la société Comelit en Tunisie est gérée par Mr JMAII LAASSAD depuis son lancement en avril 2015 et dispose d'un centre de développement sis au Pôle El Ghazala des Technologies de Communications – Ariana Nord – Tunis. Comelit a opéré sur des perspectives ambitieuses, une étude de marché cohérente, et une équipe de travail expérimentée et douée des dernières innovations en « High Tech », à fin de surmonter les challenges d'une période économique délicate.

Avec un bon personnel hautement qualifié, Comelit a envisagé une demande en hausse et un large potentiel de clients à satisfaire, son logo est indiqué dans la Figure 1.1.

Figure 1.1: Logo de Comelit

1.2.1 Histoire de Comelit dans le monde

Les produits Comelit sont présents dans plus de 70 pays à travers le monde, distribués par un réseau technico-commercial de 13 filiales et garantis par une recherche depuis toujours à l'avant-garde, par l'optimisation continue des processus fonctionnels et applicatifs et par un design entièrement made in Italie. La Figure 1.2 indique la distribution des sociétés Comelit dans le monde.

Figure 1.2 : Distribution des sociétés Comelit dans le monde

1.2.2 Les certifications

L'évolution de la production Comelit est une réalité faite de passion, de ténacité et d'intelligence, mais surtout de qualité.

Des choix de qualité qui ont permis à l'entreprise de jouer un rôle de leader mondial sur ce secteur d'activité, tout en s'ouvrant à d'autres marchés par les nombreuses certifications qui accompagnent le projet d'entreprise et l'offre de production.

Depuis 1997, la certification ISO 9001, confirme des choix de gestion qui se sont consolidés au fil des ans et, depuis 2004, Comelit a étendu son système de gestion aux aspects environnementaux. La certification ISO 14001 rend évident son engagement d'agir dans une logique de développement durable.

1.2.3 Les produits de Comelit

Les systèmes de visiophonie Comelit se caractérisent par leur fiabilité et longévité. Satisfaisant les exigences de qualité les plus pointues, les leaders du secteur choisissent Comelit pour ses nombreux domaines d'application : maisons individuelles, complexes résidentiels, édifices publics, industrie.

1.2.3.1 Vidéophonie

L'interphone vidéo est un système de contrôle d'accès dans une propriété ou un immeuble au même titre qu'un interphone audio ou une sonnette.

Le boîtier placé dans la rue peut être équipé d'un digicode pour renforcer la sécurité d'une propriété.

Le produit SimpleVidéo est composé de systèmes à 2 fils et systèmes IP numériques, c'est La meilleure solution technique pour satisfaire les exigences de toutes les applications. La figure 1.3 présente des exemples de produits vidéophonie.

Figure 1.3 : Produit Vidéophonie

1.2.3.2 Contrôle d'accès

Le contrôle d'accès consiste à vérifier si une personne demandant l'accés à un bâtiment a les droits nécessaires pour le faire.

La technologie représente des badges et lecteurs de proximité en technologie Myfare, pour la gestion des accès et des scénarios.

La figure 1.4 présente des exemples de produits contrôle d'accès.

Figure 1.4 : Produit Contrôle d'accès

1.2.3.3 Vidéo Surveillance

Le système de vidéosurveillance est un système de caméras et de transmission d'images, disposé dans un espace public ou privé pour le surveiller à distance, c'est un type de télésurveillance. Les images obtenues avec ce système, peuvent être traitées automatiquement et/ou visionnées puis archivées ou détruites.

Le produit SimpleTVCC est une gamme complète de produits pour la surveillance vidéo. Analogique, IP, HD-SDI, en mesure de garantir un excellent niveau de sécurité.

La Figure 1.5 présente des exemples de produits vidéo surveillance.

Figure 1.5 : Produit Vidéo Surveillance

1.2.3.4 Domotique

Le produit SimpleHome est un système de domotique pour le contrôle de l'éclairage, des volets, de l'arrosage et des consommations.

Il permet au consommateur d'avoir le pouvoir du contrôle total de son propre espace au service du confort et de la sécurité disposant également de la simulation de présence.

La Figure 1.6 présente des exemples de produits domotique.

Figure 1.6 : Produit Domotique

1.2.3.5 Alarme anti-intrusion

Les alarmes anti-intrusion ont pour fonction première de dissuader d'éventuels cambrioleurs d'effectuer leur méfait.

Le produit SimpleSafe est un système complet en mesure de répondre à toutes les exigences en matière de sécurité, à partir de systèmes câblés ou via radio: de la détection à la signalisation, à l'intégration avec d'autres systèmes, toujours au service de l'utilisateur.

La Figure 1.7 présente des exemples de produits anti-intrusion.

Figure 1.7 : Produit Anti-intrusion

1.3 Présentation du contexte du projet et de l'environnement de développement

Dans cette section, nous présentons le contexte de notre projet ainsi que l'environnement de développement utilisé.

1.3.1 Contexte du projet

Notre projet porte sur la conception et le développement d'un support de communication et de services pour un produit alarme. Nous présentons dans ce qui suit le contexte général de notre projet.

1.3.1.1 Description d'un produit alarme

La sécurité est devenue une préoccupation majeure, d'autant plus que la plupart des pays développés ont enregistré une hausse significative des cambriolages de maison dans les dernières décennies. Renforcer la sécurité d'une maison est la première étape dans la prévention des cambriolages.

Pour cette raison, les systèmes d'alarme sont de plus en plus propagés et demandés dans les marchés mondiaux et spécialement dans les pays modernes, sa réalisation permet de mener à des activités intéressantes et instructives sur les plans technique et scientifique dans le cadre de la voie technologique [3].

1.3.1.2 Chiffre d'affaire mondial des produits alarmes

Selon une étude réalisée par « businesscoot » sur les marchés d'alarme. La fabrication d'alarmes et du matériel de vidéosurveillance est un secteur en forte croissance depuis 10 ans. Le marché mondial s'élevait à 14.1 milliards de dollars en 2013 [4].

1.3.1.3 Les normes des produits alarmes

Pour entrer dans le marché des produits d'alarmes, les constructeurs doivent obéir à des normes spécifiques à la fabrication. Il existe deux grandes normes pour l'alarme [5] :

- Norme NFA2P
- Norme Européennes EN 50131

Les utilisateurs des produits alarme préfèrent les systèmes d'alarme certifiés pour être sûrs et certains que leurs systèmes d'alarme seront résistants aux tentatives de neutralisation.

1.3.1.4 Détail technique sur les produits alarmes

Le produit alarme développé par Comelit présente des innovations et une flexibilité par l'ajout de fonctionnalités selon leurs études réalisées pour couvrir les besoins de sa clientèle cible.

Les systèmes doivent protéger la maison contre le cambriolage, le système comprend essentiellement la centrale d'alarme, les détecteurs, une communication avec les smartphones, la télécommande, la sirène et un écran tactile.

Une fois installé, le système d'alarme doit obéir à plusieurs fonctionnalités [6] :

- Détecter : Son premier but est de détecter la présence d'intrus. Cette détection passe par les différents détecteurs installés. Ces détecteurs sont munis d'une autoprotection que les protège contre la détérioration ou l'arrachement.

- Dissuader : On considère que le son émis par la sirène d'alarme permet de dissuader plus de 95% des cambrioleurs. Il est très puissant pouvant atteindre 110dB. De plus, la plupart des sirènes dispose d'un signal lumineux, attirant notamment l'attention du voisinage.

- Analyser : La centrale alarme, cœur du système est chargée de déclencher les outils de dissuasion et de communication comme la sirène ou encore les transmetteurs téléphoniques comme par exemple le module 3G qui fera l'objet de notre étude.

- Alerter : C'est le point final du processus d'action d'un système d'alarme. Les voisins, les proches, une centrale de télésurveillance, un poste de police ou n'importe qui peut être averti par un système de défense et un mécanisme de notification. Grâce à cette alerte, une intervention rapide et efficace pourra être enclenchée.

Après la mise en marche du système d'alarme, il doit détecter les changements du milieu, traiter l'information, dissuader les intrus et enfin alerter les individus responsables et compétents à l'intervention.

1.3.2 Spécification de l'alarme Comelit

Nous présentons dans cette section, l'environnement de développement de l'alarme et le choix des environnements Matériel et Logiciel. Nous exposons une architecture du produit alarme et les périphériques utilisés.

Après une étude de la conception du produit alarme, l'équipe Comelit a réalisé une spécification de son produit alarme qui envisage de respecter la norme européenne EN50131.

Pour la bonne mise en œuvre du produit alarme l'équipe Comelit a décomposé le produit en des éléments liés entre eux. L'architecture de l'environnement de développement se décompose comme suit :

- Couche Matériel : constituée de cartes électroniques, d'un microcontrôleur, des périphériques et de modules d'extensions.
- Couche Logiciel : elle se compose d'un système Linux, des services et d'une application graphique pour gérer la couche Matériel.

La figure 1.8 représente la hiérarchie de notre environnement de travail et l'organisation des différentes couches.

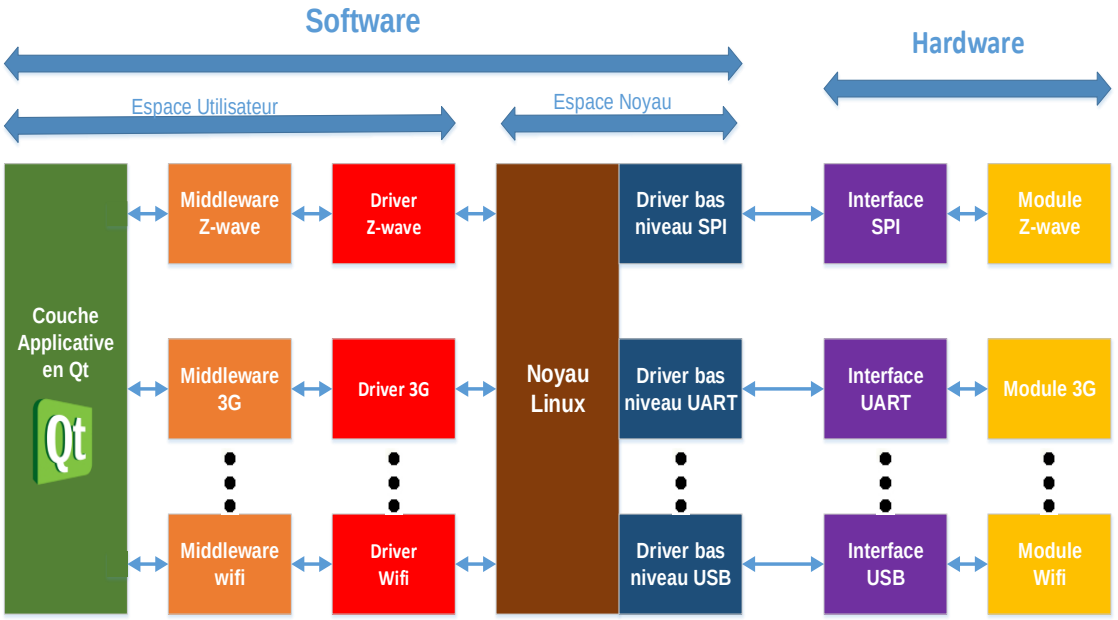


Figure 1.8 : Architecture du Système Alarme développé par CRED

Dans ce qui suit, nous présentons les couches Matériel et Logiciel du produit alarme de la société Comelit.

1.3.2.1 Présentation du Matériel

Dans cette section, nous détaillons la partie matérielle de notre produit alarme. En effet, il est constitué essentiellement de la plateforme Nuvoton. Cette dernière est constituée par un microcontrôleur N3292x à base d'un processeur ARM9.

1.3.2.1.1 Présentation du Microcontrôleur N3292x

Le microcontrôleur N3292x de Nuvoton est basé sur le processeur 32 bits ARM926EJ-S. Il possède 128 broches MCP (Multi-Chip Package) dont 80 broches de GPIO (General Purpose

Input Output). Il intègre plusieurs périphériques tels que le décodeur vidéo (H.264), MAC Ethernet, JPEG codec, interface de capteur CMOS, 32 canaux SPU (Sound Processing Unit), ADC (Analaog Digital Converter), DAC (Digital Analog Converter), encodeur TV. La figure 1.9 représente le schéma synoptique du microcontrôleur N3292x.

Le N3292x est désigné spécialement pour l'accélération du streaming audio/video à une haute performance et avec une fréquence maximale de 240MHz. Il est bien conçu en terme de rentabilité (coût/efficacité) pour la solution streaming audio/vidéo.

Notre microcontrôleur basé sur ARM peut être porté sous l'environnement Linux. D'autre part, l'environnement de programmation open source donne au produit à développer sur la plateforme plus de flexibilité [7].

Nous citons quelques domaines d'application supportés par le microcontrôleur N3292x de Nuvoton :

- Les cameras IP
- Smartphone et accessoire pour tablettes
- Vidéo moniteur pour bébé
- Les Interfaces Homme Machine (IHM)
- domotique

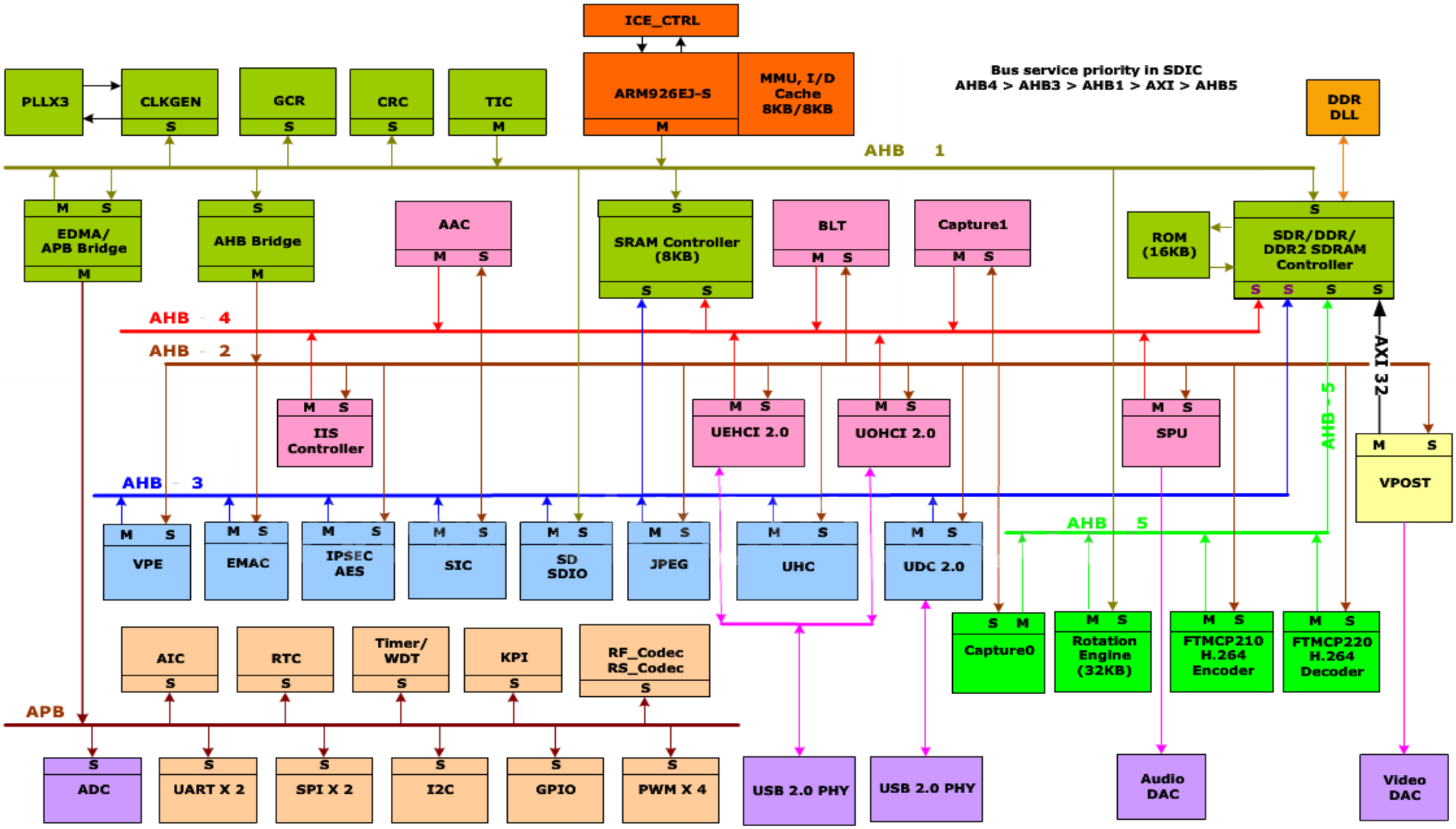


Figure 1.9: Schéma synoptique du microcontrôleur N3292x [8]

1.3.2.1.2 Présentation de la plateforme de test du produit alarme

L'équipe Matériel de Comelit a réalisé une plateforme de test spécifique pour le développement du produit.

Cette plateforme est composée essentiellement de :

- Microcontrôleur N3292.
- Module 3G Quectel M95
- Connecteur Z-Wave.
- Ecran tactile
- Bouton analogique
- Mémoire NAND
- Camera
- Wifi
- Ethernet
- Microphone
- Haut-parleur
- Connecteur USB

Tous ces éléments sont indiqués sur la figure 1.10.

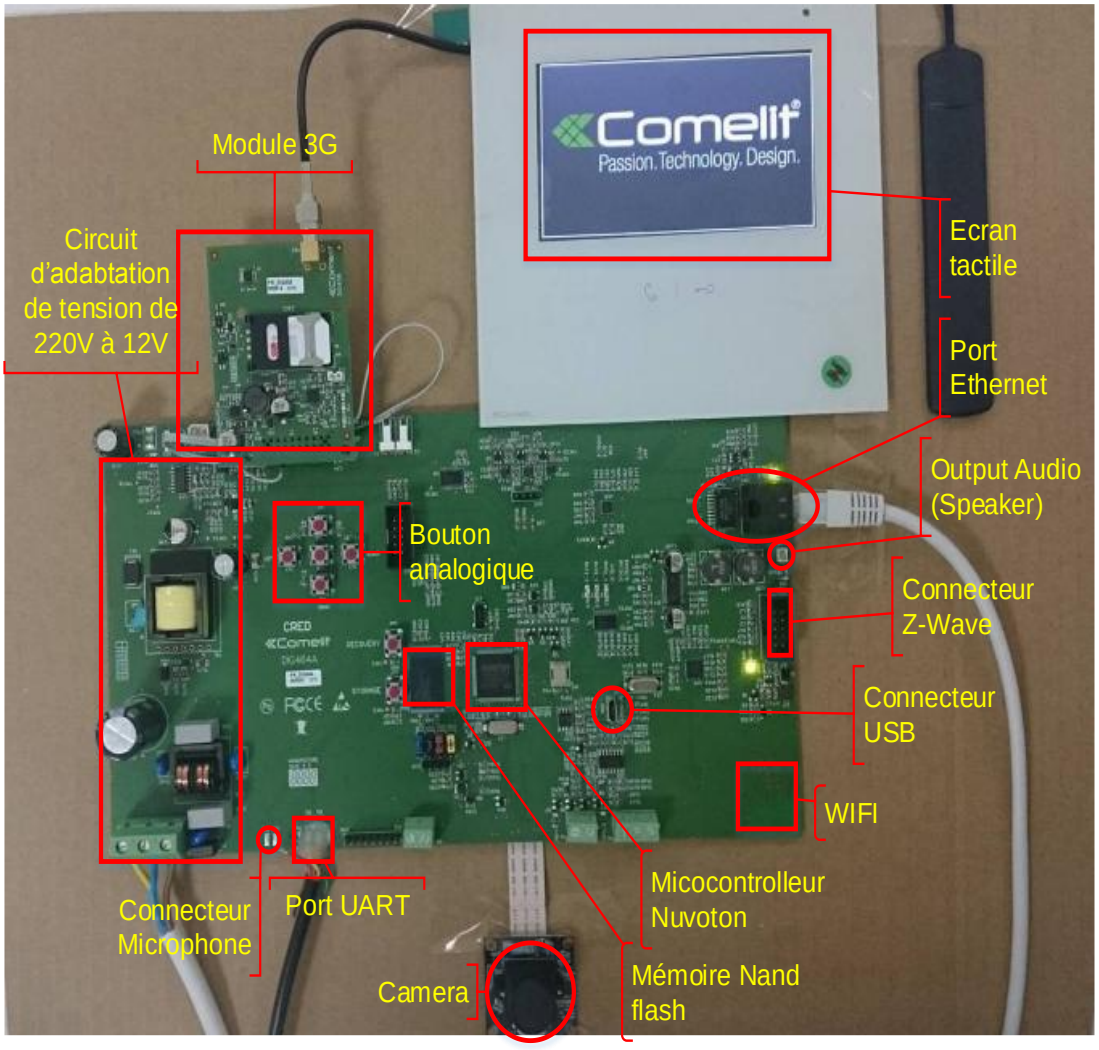


Figure 1.10 : Prototype du produit de test d'alarme

1.3.2.2 Présentation du Logiciel

Pour développer le produit alarme, l'équipe Comelit a choisi d'utiliser Linux comme système d'exploitation pour être l'installé sur le microcontrôleur de Nuvoton basé sur ARM. Ils ont choisi l'environnement de développement Buildroot et ils ont ajouté des bibliothèques spécifiques aux besoins des applications cibles.

L'image 1.11 représente les fonctionnalités principales pour le développement d'un système embarqué. Il se compose de deux environnements, le PC de développement et du code source et le système embarqué cible.

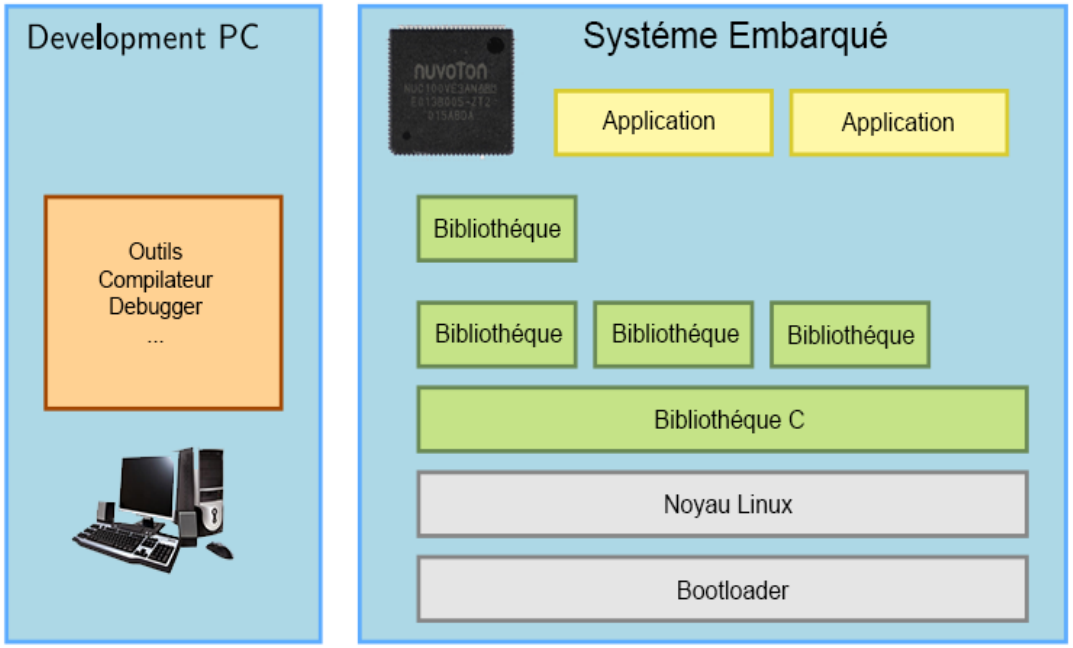


Figure 1.11 : Architecture de développement système embarqué Linux

Dans cette section, nous présentons la partie Logiciel, dans laquelle nous définissons l'environnement Buildroot, le cross-compilateur Toolchain, le noyau Linux, les bibliothèques logiciel, le code source du projet alarme et le bootloder [9].

1.3.2.2.1 Cross-compilateur (Toolchain)

Le cross-compilateur est une collection d'outils utilisés pour développer une application embarquée spécifique à un système cible. C'est un compilateur qui tourne sur la machine de développement. Il est souvent utilisé dans le cadre de la construction d'un fichier binaire exécutable qui sera installé ou exécuté sur un autre système d'architecture différente à celle utilisé pour le développement.

1.3.2.2.2 Bootloader

C'est un fichier binaire qui est responsable de l'initialisation des différents périphériques de la plateforme matérielle. Il permet de charger et d'exécuter le noyau Linux dans la mémoire. C'est le premier programme qui s'exécute lors du démarrage de la carte.

1.3.2.2.3 Noyau Linux

Le noyau Linux est un noyau de système d'exploitation de type UNIX. C'est un logiciel libre développé essentiellement en langage C. il permet l'ordonnancement des processus, la gestion de la mémoire et du réseau, des driver bas niveau de différents périphériques. Il permet aussi de fournir des services pour l'espace utilisateur.

1.3.2.2.4 Les bibliothèques logiciel

C'est une couche intermédiaire entre l'espace utilisateur et l'espace noyau. Une bibliothèque est une collection de fonctions, qui peuvent être déjà compilées et prêtes à être utilisées par des programmes.

1.3.2.2.5 L'environnement Buildroot

Buildroot est un outil qui simplifie et automatise le processus de construction d'un système Linux complet pour un système embarqué, en utilisant la compilation croisée. Il contient des fichiers correctifs (patchs) qui seront appliqués sur les paquets cibles. Buildroot peut générer un système de fichiers racine (rootf), une image du noyau et une image du bootloader. Buildroot est surtout utile pour les personnes travaillant sur les systèmes embarqués, en utilisant différentes architectures de processeurs (x86, ARM, MIPS, PowerPC, etc.) [10].

Il permet d'automatiser le processus de construction des systèmes embarqués et facilite le processus de compilation croisée.

Dans la figure 1.12 nous présentons l'interface de configuration du buildroot.

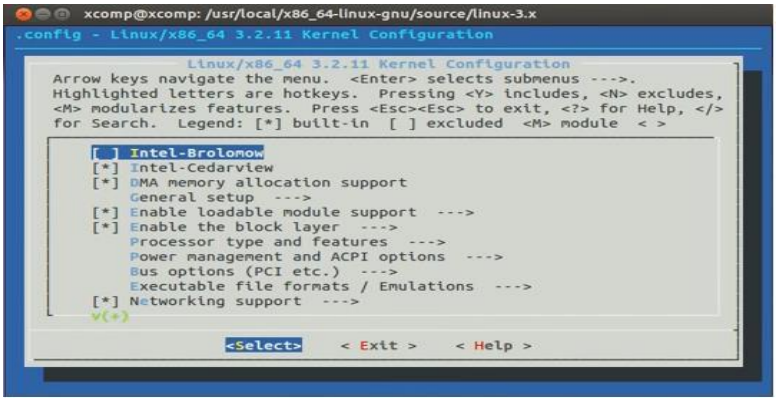


Figure 1.12 : Interface Menuconfig

Dans l'environnement de développement, l'équipe Comelit a ajouté quelques spécificités pour la carte de développement Nuvoton, ainsi que les bibliothèques nécessaires pour le développement du Logiciel de l'alarme.

Voici les caractéristiques de la version buildroot utilisée durant le développement :

- Cible: arm-nuvoton-linux-uclibcgnueabi
- Thread model: posix
- gcc version 4.8.4 (Buildroot 2015.02-00121-ga40c05b)

1.3.2.2.6 Composition du projet alarme

L'équipe Comelit a choisi de décomposer l'environnement de développement de l'espace utilisateur en trois couches qui sont liées entre elles pour la bonne hiérarchie et structuration du code développé :

- Couche Driver (Low layer)
- Couche management (Middleware)
- Couche applicatives (High layer)

La couche Driver est un ensemble de fichiers qui ont une intéraction directe avec le matériel, nous citons par exemple :

- Driver 3G : il réalise une communication avec le module 3G Quectel M95 via des fonctionnalités Read/Write du driver UART.
- Driver UART : il réalise une communication série avec d'autres périphériques.
- Driver ADC : il réalise la conversion analogique numérique qui permet l'interfaçage de plusieurs périphériques comme le Keypad, l'écran tactile, capteurs RF.

La couche Middleware est la couche intelligente qui gère les fonctionnalités des drivers. Elle représente une couche intermédiaire entre la couche applicative et les drivers. Elle se base sur plusieurs services tels que :

- Le service Telephony, qui est responsable de la réception des notifications reçues du module 3G.
- Le service Push qui est responsable de l'envoi des messages Push et la gestion des utilisateurs concernés par la réception des messages Push.

La couche Applicative est basée sur une application graphique en utilisant la bibliothèque QT, comme une interface pour l'utilisateur afin de manipuler son système d'alarme.

1.4 Présentation des besoins et du cahier des charges

Comelit a pour objectif de réaliser la conception et le développement d'une solution alarme, pour cela elle envisage de développer les trois couches de l'alarme, la couche Driver, la couche Middleware et la couche applicative au niveau de l'espace utilisateur.

Parmi les fonctionnalités qui doivent être supportées par le produit alarme est la communication et le transfert de données avec l'utilisateur, la centrale de sécurité et la police.

Le produit alarme doit être capable de notifier l'utilisateur de chaque nouveau changement d'état dans les différentes zones de la maison. Il doit être aussi contrôlér à distance par l'utilisateur pour gérer l'état de son alarme.

Il s'agit donc de mettre en place un mécanisme de contrôle et de notification avec les utilisateurs, ainsi que le support de communication.

Afin de réaliser cet objectif, nous nous sommes intéressés à l'environnement logiciel qui se base sur le Noyau Linux et spécialement dans l'espace utilisateur afin d'effectuer les tâches suivantes :

- Développement du Driver 3G qui communique avec le module 3G Quectel M95.
- Notifier la couche Middleware des événements externe reçu.
- Implémenter le Driver 3G sur le microcontrôleur de Nuvoton.
- Réaliser le service contrôle SMS qui est un service pour l'utilisateur afin qu'il puisse contrôler son alarme.

Nous avons aussi la charge de réaliser une application qui a pour objectif de notifier l'utilisateur lors d'une intrusion, c'est le mécanisme de « Push Notification ». Au cours du développement de cette application, nous devons réaliser :

- Une application Android qui permet de
- La configuration de l'API du serveur Google Cloud Messaging qui permet
- Le développement d'un service qui sera implémenté sur la plateforme Nuvoton et qui permet d'envoyer un message Push aux utilisateurs.

Le driver 3G, les services SMS contrôle et Push notification à réaliser sont des éléments de la partie connectique du produit alarme.

1.5 Démarche suivie pour la réalisation du projet

Après avoir vérifié la présence des moyens matériels et logiciels nécessaires, il nous a été demandé, de développer les fonctionnalités offertes par le module 3G et de réaliser l'application de push notification

Pour la réalisation de ce projet, nous avons divisé notre travail en deux grandes parties.

- Partie I : Développement Driver 3G et service contrôle SMS.
- Partie II : Développement Application Push Notification.

Pour la première partie, notre travail consiste à :

- Etudier les différentes fonctionnalités du module 3G quectel M95, et l'installation de l'environnement du travail.
- Explorer les différentes fonctionnalités du module 3G Quectel M95 avec l'interface graphique proposé par le constructeur.
- Développer le driver du module 3G et l'implémenter sur la plateforme Nuvoton.
- Tester et valider la solution proposée Driver 3G.
- Développer le service contrôle SMS et l'implémenter sur la plateforme Nuvoton.
- Tester et valider le fonctionnement du service contrôle SMS

Pour la deuxième partie, notre travail consiste à :

- Etudier la fonctionnalité et le mécanisme de réalisation de Push Notification.
- Configurer le serveur Google Cloud Messaging afin de bénéficier de l'API responsable à la Push Notification.

- Développer l'application Android pour se connecter au serveur GCM et stocker le l'identifiant TOKEN fourni.
- Développer une application en C qui sera responsable d'envoyer une demande au serveur Google Cloud Messaging pour notifier l'utilisateur.
- Tester et valider l'application Push Notification.

1.6 Conclusion

Dans ce chapitre, nous avons essayé de mettre en évidence le cadre de réalisation du projet en présentant l'organisme d'accueil Comelit, le contexte du projet de stage et la méthodologie du travail adopté pour réaliser ce projet. Il s'agit du contexte des produits alarmes utilisés pour la surveillance. Nous avons cité l'environnement de travail pour la réalisation du produit. A la fin de ce chapitre nous avons présenté le cahier des charges et la démarche à suivie pour la réalisation du driver 3G ainsi que les services SMS contrôle et Push notification. L'étude approfondie pour le développement du driver 3G et le service SMS contrôle feront l'objet du prochain chapitre.

Chapitre 2

Conception et développement du Driver 3G et du service Contrôle SMS

2.1 Introduction

Le deuxième chapitre est dédié à l'étude des différentes fonctionnalités à réaliser avec le module 3G et qui mènera au développement du Driver 3G.

Dans la première partie du chapitre nous présentons en particulier le module 3G Quectel M95, sa communication avec le microcontrôleur du produit alarme Novoton. Nous soulignons également son utilité dans notre projet. Dans la deuxième partie de ce chapitre, nous présentons l'architecture du Driver 3G développé ainsi que sa conception, nous présentons aussi le service contrôle SMS. Les tests de validation des fonctionnalités développées du Driver 3G sont présentés dans la dernière section du chapitre.

2.2 Présentation du module 3G Quectel M95

Le GSM est un communicateur cellulaire qui est intégré dans le module 3G. Il le communicateur principal ou secondaire pour le produit alarme. En effet, la connexion sans fil apporte la flexibilité nécessaire pour l'installation d'un système de sécurité et représente une méthode de transmission de signal d'alarme sûre. Nous détaillons dans cette partie le module 3G et son utilité dans le produit alarme.

2.2.1 Besoins fonctionnels du Module 3G

Le module 3G possède un rôle important dans le produit alarme. En effet, en cas d'une intrusion ou d'un problème technique de coupure Internet, le système alarme continue sa surveillance en utilisant le module 3G. Dans notre projet, nous l'avons utilisé pour réaliser certains types de communication, soit avec l'utilisateur ou avec la centrale de surveillance ARC (Alarm Receiving Center). Pour un produit alarme, le module 3G doit permettre les fonctionnalités suivantes:

1. Service Voice Guide : c'est un service qui contient un menu de choix vocal présenté lors d'un appel téléphonique. Il est utile pour l'utilisateur afin de manipuler son alarme.
2. Service SMS control : c'est un service identique au Voice Guide. Mais l'utilisateur utilise les SMS afin de contrôler son larme à distance.
3. Envoi d'E-mails.
4. Envoi des sockets TCP/UDP.
5. Envoi des requêtes Get/Post HTTP.
6. Envoi des MMS.
7. Communication avec l'ARC via le Protocol ContactID. Ce protocole réalise un appel téléphonique et envoie des codes DTMF (Dual Tone Multi Frequency).

Pour la réalisation de ces besoins, nous devons configurer le module 3G. Un Driver 3G sera alors la solution pour la configuration et la communication avec le module 3G.

La société Comelit a choisi le module 3G Quectel M95 pour l'intégrer dans le produit alarme. Nous présentons dans ce qui suit ce module.

2.2.2 Présentation du module 3G M95

Le Module 3G M95 est l'un des plus petits modules GSM/GPRS à quatre bandes. Il adopte le dernier chipset de MediaTek, qui est de taille 19,9 × 23,6 × 2.65mm. Il a un potentiel d'ultra basse consommation d'énergie et une plage de température étendue (-40°C jusqu'à +85°C).

Grâce à la technologie intégrée sur une petite surface, le Module M95 a été facilement intégré dans des applications embarquées à faible volume. Ce module est adapté aux applications qui possèdent des exigences strictes de coûts et d'efficacité [11]. On peut l'utiliser dans plusieurs applications :

- Les compteurs intelligents d'électricité.
- Service de trafic maritime.
- Point de vente sans fil.
- Suivi personnel.
- Les boitiers d'alarmes.

Le module M95 est construit avec la technologie QuecFOTA qui lui permet de faire des mises à jour de son firmware à distance. Il possède aussi des fonctionnalités supplémentaires intégrées comme le protocole TCP/IP, UDP, FTP, HTTP, décodage DTMF, et une transmission

rapide et fiable des données, de la voix et des SMS via le réseau GSM/GPRS. Le module est riche en fonctionnalités qui lui rendent approprié pour une large gamme d'applications [11]. La figure 2.1 présente le module 3G M95.

Figure 2.1 : Module 3G Quectel M95

La société Quectel a réalisé une plateforme d'évaluation aux développeurs qui utilise le module 3G pour faciliter leurs tests. Nous présentons l'architecture de la plateforme d'évaluation utilisée dans la figure 2.2 [12].

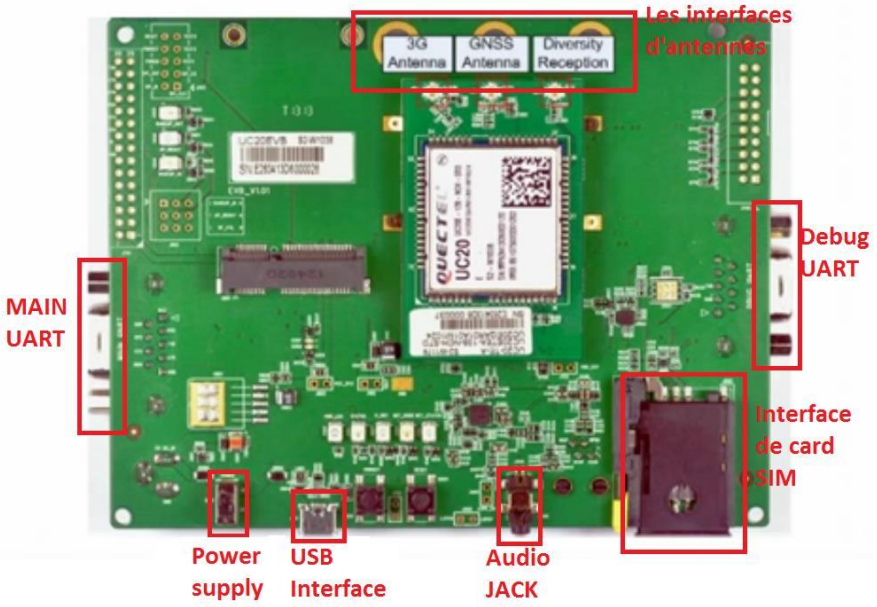


Figure 2.2 : Carte d'évaluation du Module 3G Quectel M95

Le module 3G Quectel M95 fournit plusieurs fonctionnalités de communication à travers le réseau GSM/GPRS y compris les fonctionnalités exigées par le cahier des charges du projet. Il est interfacé avec la plateforme de test du produit alarme via la communication série asynchrone UART (figure 1.10).

Nous détaillons dans ce qui suit la communication du module 3G Quectel M95 avec le produit alarme.

2.2.3 Communication avec le Module 3G

Pour communiquer avec le module 3G nous avons utilisé la communication UART présenté dans la figure 2.3. Le Nuvoton gère le module 3G et communique avec lui via la liaison série UART.

La communication UART du module 3G avec la carte Nuvoton est réalisé à travers les broches RX et TX qui sont inversement liées aux broches RX et TX du module 3G comme il est indiqué dans la figure 2.3.

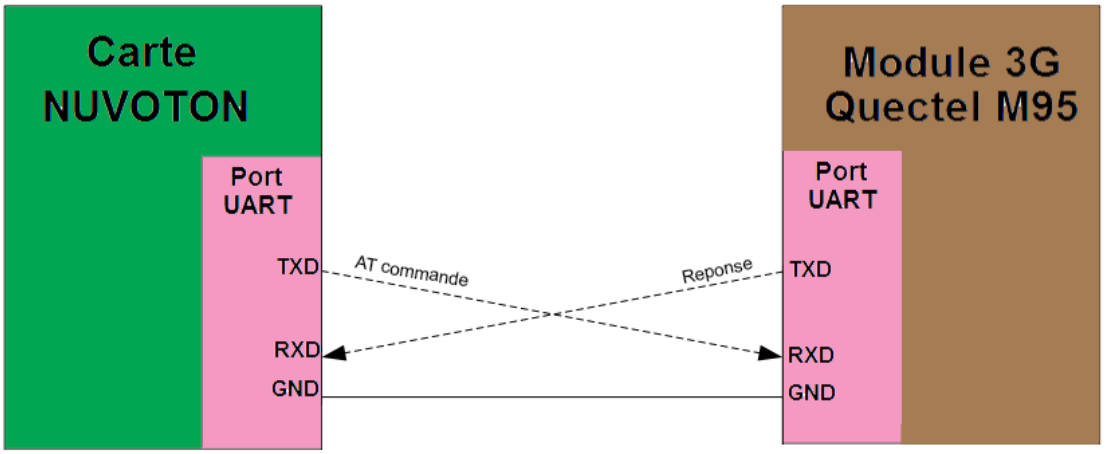


Figure 2.3 : Schéma de communication UART

Cette communication UART est utilisée pour commander le module 3G à travers l'envoie d'AT commandes.

La majorité des modems modernes et des modules extensibles, tel que le module 3G M95, disposent d'un jeu de commandes textuelles appelé le jeu AT. Ce jeu se compose de commandes simples préfixées par AT [13]. Elles ont été conçues spécialement pour la configuration et le contrôle des modules afin de réaliser des fonctions spécifiques.

Les commandes sont utilisées pour réaliser différentes opérations telles que raccrocher, appeler numéro, envoyer SMS, lire SMS, etc. Après réception de la commande, le module 3G retourne une réponse sous forme de chaine de caractères selon la commande envoyée.

En général, les fabricants fournissent avec le modem de la documentation suffisante pour sa configuration et son utilisation. Dans notre projet, nous avons utilisé les AT commande pour communiquer avec le module 3G Quectel M95. [11]

Voici quelques exemples d'AT commande pour ordonner le module 3G M95:

- « AT+CLCC » : lister les appels en cours.

- « AT+CMGS="<numéro>" »: envoyer un SMS au numéro indiqué.
- « ATD<numéro>; » : appeler le numéro indiqué.

Dans la partie suivante, nous décrivons la réalisation du driver responsable du fonctionnement des besoins prévus du module 3G.

2.3 Présentation du driver 3G et du service contrôle SMS

Dans cette partie, nous présentons les détails de développement du driver 3G et son architecture. A la fin de cette section, nous présentons le service Contrôle SMS et sa communication avec le driver 3G.

2.3.1 Présentation du Driver 3G

Il s'agit de développer un driver 3G ayant certaines fonctionnalités, nous commençons alors par expliciter ces dernières.

2.3.1.1 Rôle du driver 3G

Un driver est une couche logicielle qui permet de dialoguer avec le périphérique cible via une interface de communication (UART, SPI, I2C, USB, etc.) en échangeant des informations. C'est une composition de fonctions qui permet d'initialiser et d'exploiter les fonctionnalités offertes par un module.

Le driver 3G à concevoir, puis à développer doit être capable de réaliser la configuration des fonctionnalités suivantes :

- Envoi et réception SMS.
- Envoi et réception des appels téléphonique.
- Envoi et réception des sockets TCP/UDP.
- Envoi des requêtes Get/Post HTTP et HTTPS.
- Envoi des E-mail (SMTP).
- Envoi des messages MMS.
- Réception du code DTMF.
- Envoi requête USSD.
- Réalisation d'autres fonctionnalités (niveau du signal réseau, recherche des réseaux disponibles, information sur la carte SIM, détection de la carte SIM, etc.).

2.3.1.2 Présentation du fonctionnement du driver

Dans notre cas, nous allons réaliser un Driver 3G, qui communique avec le module 3G à travers une communication série UART principalement via les fonctions en C : **UART_Read()** et **UART_Write()**.

Nous avons réalisé notre driver pour qu'il soit capable de couvrir tous les besoins nécessaires et demandés. Pour cette raison nous avons changé et modifié certaines fonctions au fur et à mesure de l'avancement du projet et suite aux obstacles rencontrés.

Au début de notre projet, le Driver 3G était conçu pour réaliser des Appels, envoyer des SMS et envoyer des MMS à travers l'envoi des AT commandes spécifiques. Le driver 3G attend ensuite la réponse du module 3G qui peut être soit « OK », soit « ERROR ».

Dans la figure 2.4, nous avons réalisé un exemple de communication du Driver et du module 3G Quectel M95 à travers l'envoi d'une commande simple 'AT' pour vérifier si le module fonctionne convenablement en retournant la chaine de caractères 'OK'.

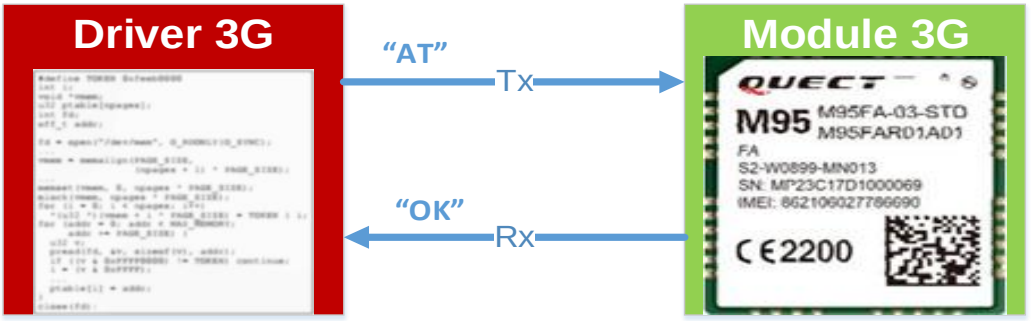


Figure 2.4: Exemple de Communication Driver et Module 3G

Cependant, le driver 3G peut aussi recevoir des événements d'une façon asynchrone (un appel téléphonique ou un SMS) par la réception également d'une chaine de caractères sans que nous réalisions une demande d'un service (envoi d'AT commande). Le problème se pose lorsqu'on reçoit en même temps la réponse suite à l'envoi d'une requête (AT commande) et aussi des données suite à une réception d'événements. Dans ce cas précis, des informations utiles seront perdues.

Pour remédier à ce problème, nous avons eu recours à l'utilisation de processus (thread). De cette façon il n'y aura pas de confusion entre données de deux origines différentes (réponse à une AT commande, réception d'un événement asynchrone).

Notre programme du driver 3G se compose essentiellement de deux parties pour gérer les fonctionnalités du module 3G :

- Partie 1 : recevoir et traiter les événements extérieurs en provenance du module 3G (Rx).
- Partie 2 : développer des fonctions qui exploitent les différents services fournis par le module 3G à l'aide de l'envoi des AT commandes via la communication série UART.

La figure 2.5 explique l'architecture fonctionnelle de notre driver 3G, qui sera détaillée par la suite.

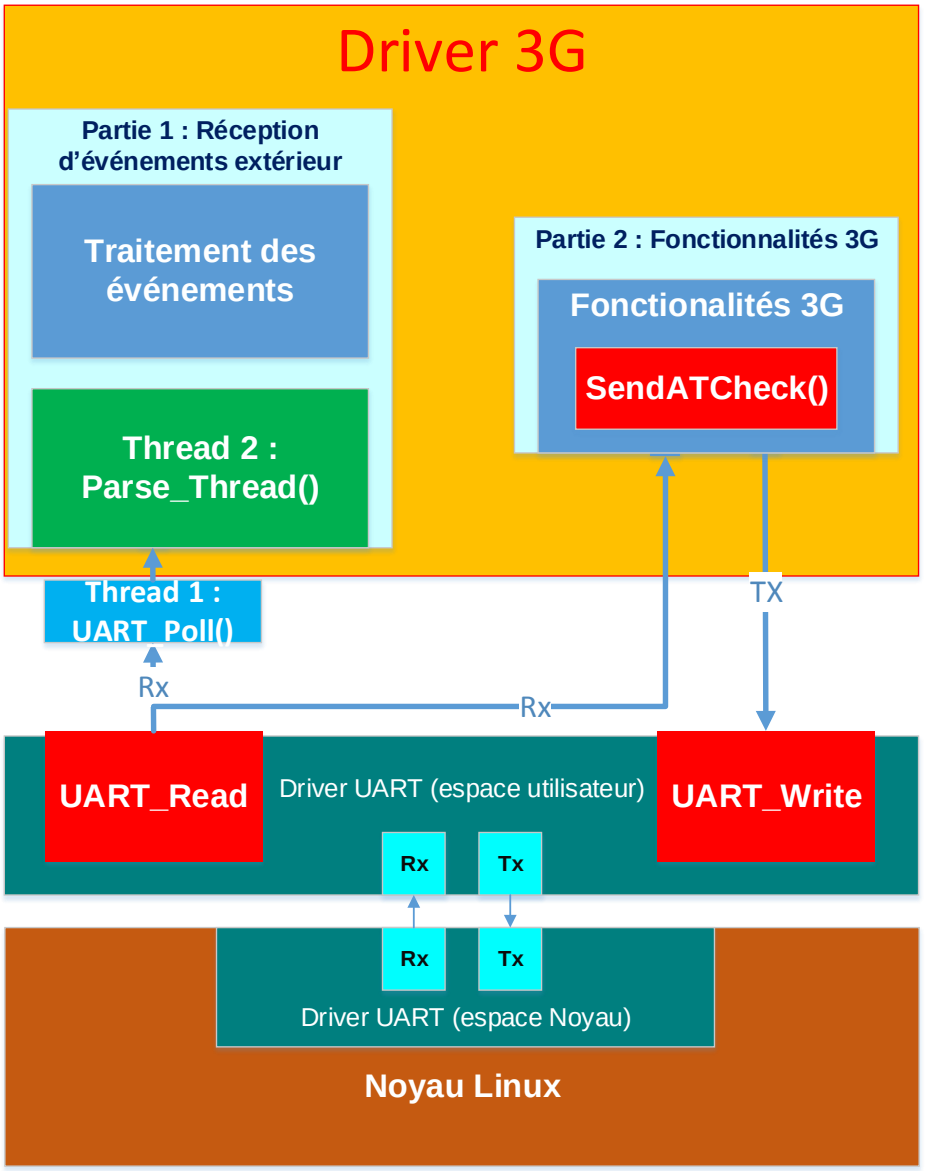


Figure 2.5 : Communication logiciel entre driver 3G et UART

Le driver 3G est un programme écrit en C qui s'exécute sur le microcontrôleur N3292x de la plateforme Nuvoton. Ce driver doit communiquer avec le module 3G via le driver UART en

mode transmission et réception à l'aide de deux fonctions développées, **UART_Read()** et **UART_Write()**.

Toute chaîne de caractères reçue depuis le module 3G finit forcément par les caractères « \r\n » qui sont représentés en hexadécimale par (\r : 0xa) et (\n : 0xd). Pour lire une chaine de caractères complète, il faut lire depuis le port UART jusqu'à la rencontre de cette chaine de caractères.

La communication avec le port UART se réalise via un fichier qui s'appelle descripteur de fichier « File Descriptor ». Toute lecture et écriture s'effectue à travers ce fichier de système. Pour cette raison la lecture et l'écriture du port UART sont en exclusion mutuelle soit lecture, soit écriture. Le port UART est une ressource partagée par excellence et l'utilisation d'un Mutex est primordiale.

Pour assurer la lecture ou l'écriture du port UART, nous avons utilisé les fonctions suivantes pour manipuler le fonctionnement des Threads lors d'accès au File Descriptor de l'UART:

- La fonction **pthread_mutex_init()** pour initier le Mutex lors de l'initialisation du Driver 3G.
- La fonction **pthread_mutex_destroy()** pour détruire le Mutex lors de la terminaison du Driver 3G.
- Les fonctions **pthread_mutex_lock()**, **pthread_mutex_trylock()** et **pthread_mutex_unlock()** pour protéger la ressource partagée UART.

Nous détaillons dans la suite, le fonctionnement de chacune des deux parties du driver 3G illustrées dans la figure 2.5.

2.3.1.2.1 Partie 1 : Système du processus de traitement et réception de flux de données.

La première partie de notre programme est constituée de deux Threads qui tournent d'une façon infinie et indépendamment des autres fonctionnalités du driver et détectent toute chaine de caractères envoyée depuis le module 3G via le port UART d'une façon asynchrone. La première s'appelle **UART_Poll()**, la deuxième **Parse_Thread()**.

Dans le tableau 1, nous présentons quelques exemples de données détectées par les deux Threads pour les traiter indépendamment du programme principal.

Tableau 1 : Exemple de chaine des événements reçue par module 3G

Chaine de caractères reçue	Information sur chaine reçue
“ RING “	Réception d’un appel téléphonique
“ NO CARRIER “	Coupure Appel téléphonique
“ NO ANSWER “	Pas de réponse du destinataire lors d’un appel
"BUSY"	Le destinataire est occupé lors d’un appel
“ +CMTI : "SM" “	Réception d’un nouveau SMS
“+QIURC: \"closed\"”	Socket TCP/IP est fermé
“+QIURC: \"incoming full\"”	Les lignes de sockets TCP/IP sont chargées
"+QIURC: \"recv\" "	Ouverture d’une ligne de connexion TCP/IP
"+QTONEDET:"	Réception de numéro sous forme DTMF
"+CPIN: READY"	le code pin est activé
"+CPIN: NOT READY"	le code pin n’est pas activé
"+QIND: SMS DONE"	Indique l’initialisation du serveur du service SMS
"+QSIMSTAT: 1,0"	La carte SIM n’est pas insérée
"+QSIMSTAT: 1,1"	La carte SIM est insérée
"+QSMTPPUT:"	Accusé de réception d’envoi du mail
"+QMMSEND:"	Accusé de réception d’envoi du MMS
"+CMGS:"	Accusé de réception d’envoi du SMS
"+QSMTPPUT :"	Accusé de réception d’envoi du mail

Le tableau 1 représente une liste de chaines de caractères reçues par le module 3G en cas de réception de données ou pour indiquer que l’envoi a atteint le destinataire. Ces chaines reçues sont des événements asynchrones externes qui sont détectés par le Thread **UART_Poll()**. Après la réception de la chaine de caractères, le Thread **Parse_Thread()** analyse la chaine reçue puis réalise des événements (Par exemples la lecture d’un SMS reçu **SMSInProcessing()**, la reconnaissance d’un numéro appelant **CallInProcessing()**) pour extraire les différentes informations fournies par le module 3G à propos de ces événements reçus,

Nous détaillons le traitement des threads à l’aide d’un schéma de la figure 2.6 qui représente le mécanisme générale du traitement de cette partie du Driver 3G.

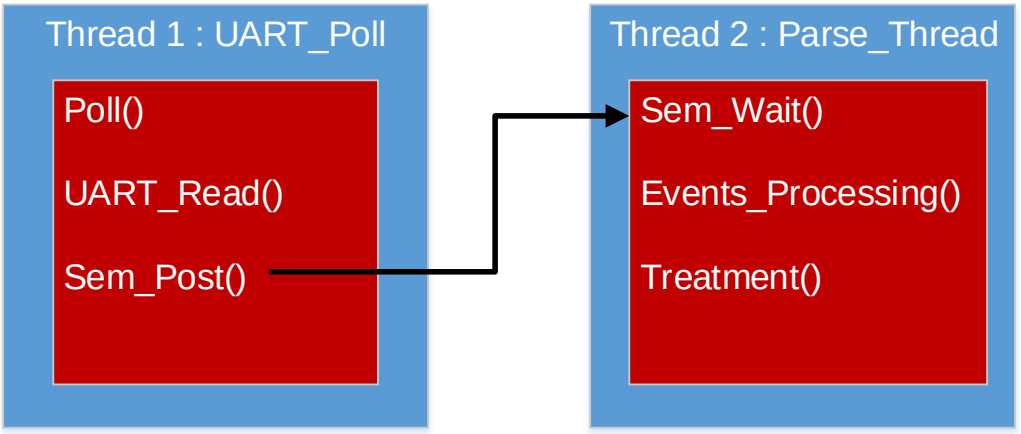


Figure 2.6 : Mécanisme d'échange de données entre deux Threads

Le Thread 2 reste bloqué dans la fonction **Sem_Wait(),** jusqu'à ce que le Thread 1 le libère avec la fonction **Sem_Post().**

Le Thread 1 nommé **UART_Poll(),** est en écoute permanente du descripteur de fichier UART à l'aide de la fonction **Poll().** Cette dernière surveille le descripteur de fichier passé en paramètre. Il réalise ensuite la lecture du port UART via la fonction **UART_Read()**. Puis, à l'aide de la fonction de sémaphore **Sem_Post()**, il libère le Thread 2 afin de réaliser la fonction **Events_Processing()**. Dans la fonction **Events_Processing()** nous comparons la chaine de caractères reçue sur le port UART avec les chaines de caractères listées par le tableau 1. Enfin, nous réalisons un traitement spécifique à l'événement survenu.

Pour chaque événement reçu, nous réalisons des fonctions spécifiques. Ensuite, nous informons le Middleware de l'événement survenu afin d'assurer la communication de données entre les différentes couches de notre environnement.

Nous avons utilisé le service du Middleware 'Telephony' pour recevoir les notifications du module 3G. Il réalise alors un traitement spécifique à chaque événement survenu et envoie les informations nécessaires à l'application graphique 'GUI' en bibliothèque QT.

La figure 2.7 présente la communication de données à travers les notifications entre les différentes couches de notre environnement.

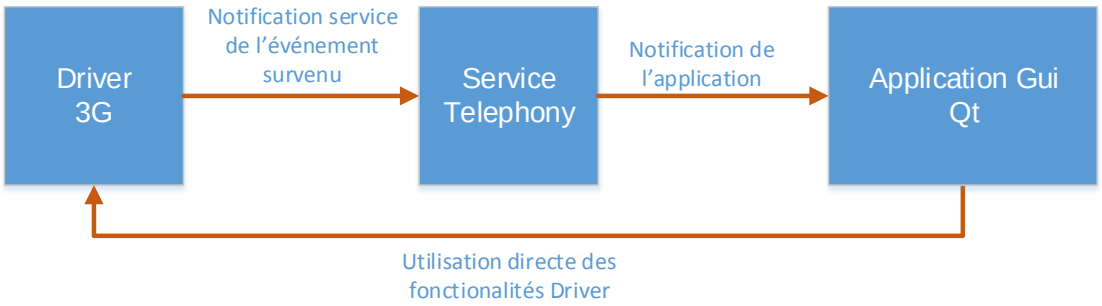


Figure 2.7 : Flux des événements et des actions

Principe de la méthode de notification

La méthode de notification dans le service 'Telephony' est basée sur 3 mécanismes qui sont détaillés dans la figure 2.8.

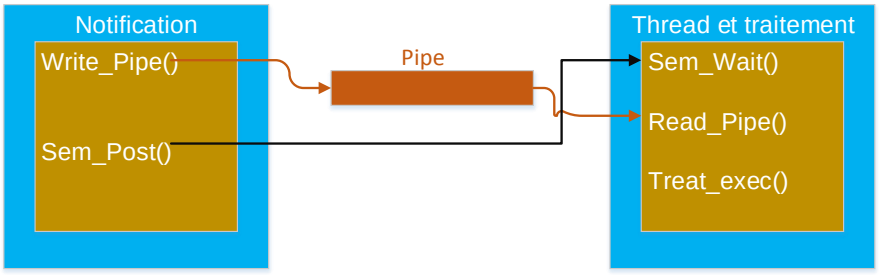


Figure 2.8 : Principe de synchronisation des fonctions

- une fonction de notification qui est appelée lors de la réception d'un événement depuis le driver 3G.
- une file d'attente 'Pipe' pour stocker les événements notifiés. Le 'Pipe' est utilisé en cas de réception de plusieurs notifications au même instant.
- un Thread qui va lire directement du Pipe et réalise le traitement spécifique à la notification survenue.

Le système de notification a deux rôles principaux :

- notification des événements asynchrones (réception Appel, SMS, DTMF, etc)
- une couche intermédiaire entre le Driver et les autres services, tel que la notification de l'application graphique des événements parcourus.

Nous détaillons dans la suite quelques exemples pour mieux comprendre la gestion des événements extérieurs.

- Lors d'une réception d'un appel téléphonique, le module 3G envoie la chaine de caractères 'RING' à notre carte Alarme, nous détectons cette chaine de caractères qui indique que nous avons reçu un Appel téléphonique. Dans ce cas, le Thread **Parse_Thread()** du driver 3G fait appel à la fonction **CallInProcessing()** pour qu'elle traite la réception d'appel et d'afficher le numéro de la personne appelante, et faire la notification en envoyant les détails de l'appelant.
- Si on reçoit la chaine de caractères 'CMTI', cela indique qu'on a reçu un SMS, nous devons alors extraire les détails de l'SMS et l'envoyer en notification vers la couche Middleware.
- Notre Thread capte aussi la réception des codes DTMF. Lorsque l'utilisateur appelle le boitier d'alarme, l'équipe Comelit à préparer un menu de choix vocal pour l'utilisateur. Lorsqu'il appelle le produit alarme. Il écoute un répondeur vocal qui lui donne le choix de choisir un numéro parmi une liste afin de réaliser un

traitement spécifique à ce choix. Par la suite à chaque détection d'un appui sur le clavier numérique, nous recevons le code DTMF réalisé par l'utilisateur.

2.3.1.2.2 Partie 2 : Les fonctionnalités 3G

Cette partie de fonctionnalité 3G se focalise sur la réalisation d'un ensemble de fonctions assurant différents services par le module 3G. Chaque fonction représente une liste d'AT commande qui a pour rôle de réaliser un traitement donné.

Comme exemple de fonctionnalité 3G, nous citons: appeler numéro, décliner appel entrant, envoyer SMS, Lire SMS reçu, envoyer MMS, envoyer et recevoir HTTPS, envoyer des requêtes TCP, envoyer Mail.

Dans le tableau 2, nous présentons quelques fonctions traitées par le Driver 3G via l'envoi d'AT commandes synchrones.

Tableau 2 : liste de fonctionnalités réalisé par le driver 3G

Fonction réalisée	Détail de la Fonction
CallAnswer()	Fonction qui répond à un appel téléphonique
CallDeny()	Fonction qui rejette un appel téléphonique
Send_SMS()	Fonction responsable d'envoyer SMS
SMS_Read()	Fonction qui lit un SMS enregistré dans la carte SIM.
Send_MMS()	Fonction responsable d'envoyer MMS
DTMFInit()	Initialise la fonctionnalité DTMF pour décodage
Get_HTTP()	Fonction pour recevoir du code HTML.
Send_Mail()	Fonction pour envoyer un Mail
Send_TCP()	Fonction pour envoyer des données TCP.
ChangeBaudeRate()	Change le baudrate du module 3G
Send_USSD()	Fonction qui réalise l'opération USSD

Pour les fonctions indiquées dans le tableau 2, nous réalisons dans chacune d'elles un envoi d'une liste d'AT commandes pour configurer le module 3G de façon à réaliser un traitement donné, et aussi de traiter les résultats de réception des AT commandes envoyés.

La figure 2.9 détaille le mécanisme d'envoi et de réception d'une seule AT commande.

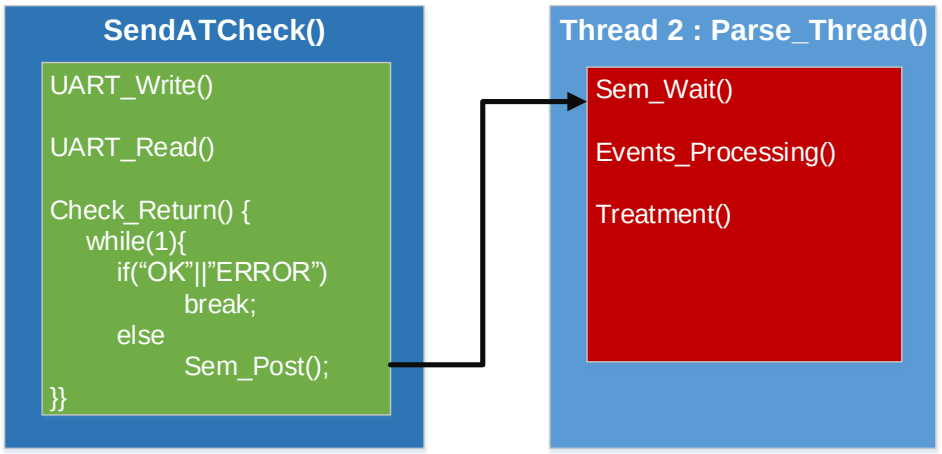


Figure 2.9 : Mécanisme fonctionnalité Driver 3G

Lors de la réception du résultat d'AT commande, nous testons si elle est 'OK' ou 'ERROR'. Mais si on lit une chaine différente de 'OK' ou 'ERROR' cela signifie que nous avons reçu un événement asynchrone et nous devons le traiter avant de lire la réponse de notre AT commande. Nous devons alors appeler la fonction **Sem_Post()** pour libérer le Thread **Parse_Thread()** responsable de la réception et du traitement des événements asynchrones reçus.

Nous détaillons deux exemples du tableau 2 pour mieux comprendre le fonctionnement des fonctions de traitement 3G, le premier est l'envoi de SMS et la fonction s'appelle Send_SMS(), le deuxième est l'envoi de Mail et la fonction se nomme Send_Mail().

- **Envoi SMS**

Pour envoyer un SMS nous avons intégré les AT commande du tableau 3 dans la fonction Send_SMS(). Cette dernière analyse les réponses reçues afin d'accomplir le traitement adéquat.

Tableau 3 : Listes d'AT commande pour envoyer un SMS

Traffic du port UART		**Explication**
AT commande	**Résultat**	
AT+CMGF=1	OK	Régler le format de message SMS en mode texte
AT+CSCS="GSM"	OK	Régler le mode d'écriture GSM
AT+CMGS="55940854" > Bonjour ISI	+CMGS: 25 OK	Le produit vous retourne '>' pour écrire ton message Appuyez **<CTRL+Z> '0x1A'** pour envoyer le message

- **Envoi Mail**

La fonction responsable à l'envoi d'un mail se nomme Send_Mail(). Cette fonction renferme les AT commandes du tableau 4 et aussi réalise des traitements suite aux réponses reçues.

Tableau 4 : Liste d'AT commande pour envoyer un E-mail

Traffic du port UART		**Explication**
AT commande envoyé	**Résultat reçu**	
Etape 1: Configuration et activation du contexte PDP (Packet Data Control).		
AT+QICSGP=1,1,"INTERNET.TN","","",1	OK	Configuration APN (Access Point Name)
AT+QIACT=1	OK	Active le context PDP (Network Link)
AT+QSMTPCFG="contextid",1	OK	Régler l'envoi du mail sur le Channel 1.
Etape 2: Configuration du serveur SMTP (Simple Mail Transfer Protocol.) server et du compte utilisateur		
AT+QSMTPCFG="ssltype",1	OK	Activation de l'encryptage des données
AT+QSMTPCFG="smtpserver","smtp.gmail.com",465	OK	Paramétrage de l'adresse du serveur Mail
AT+QSMTPCFG="account","aminebarrak@live.","**********"	OK	paramétrage du compte utilisateur et son mot de passe.
AT+QSMTPCFG="sender","amine barrak"," aminebarrak@live"	OK	Paramétrage du nom et adresse mail utilisateur
Etape 3: Configuration du contenue du Mail		
AT+QSMTPDST=1,1,aminebarrak@live.com	OK	adresse du destinataire
AT+QSMTPSUB=0,"TEST SMTP"	OK	objet du mail
AT+QSMTPBODY=0,28,120 CONNECT Bonjour, c'est un test alarm	+QSMTPBODY: 28 OK	Ajouter le contenu d'email après l'envoi de QSMTPBODY et la réception de 'CONNECT'
AT+QFUPL="RAM:smtp.txt",100,200,1 CONNECT	+QFUPL: 100,707 OK	téléchargement du fichier d'attachement « smtp.txt » à la mémoire RAM du module 3G
Etape 4: Envoyer email.		
AT+QSMTPPUT=300	OK	Envoi du Mail.
	+QSMTPPUT: 0,0	Accusé de réception du mail
Etape 5: Effacer le contenue du email et désactiver le Contexte PDP		
AT+QSMTPCLR	OK	effacement du contenue du Mail
AT+QFDEL="RAM:smtp.txt"	OK	Suppression du fichier d'attachement 'smtp.txt'
AT+QIDEACT=1	OK	désactiver le contexte associé numéro 1.

Après la présentation et l'explication du fonctionnement du Driver 3G, Nous nous focalisons dans la section suivante sur l'application de contrôle du produit alarme à travers les SMS.

2.3.2 Présentation du Service contrôle SMS

Pour faciliter la communication et le paramétrage de l'alarme, nous avons créé le service SMS contrôle pour donner la main à notre client afin de manipuler et de commander son alarme à distance [14].

Le service SMS contrôle est une solution pour contrôler l'état de l'alarme depuis leur téléphones portable. Dans ce cas le téléphone portable de l'utilisateur devient une télécommande, nous pouvons recevoir les messages d'alerte, d'alarme quel que soit son emplacement dans le monde. Nous pouvons par exemple actionner l'alarme à distance, par exemple activer le chauffage de la maison et contrôler la température.

La figure 2.10 représente la communication avec GSM pour l'envoi et la réception des SMS, dans le but de contrôler l'alarme.

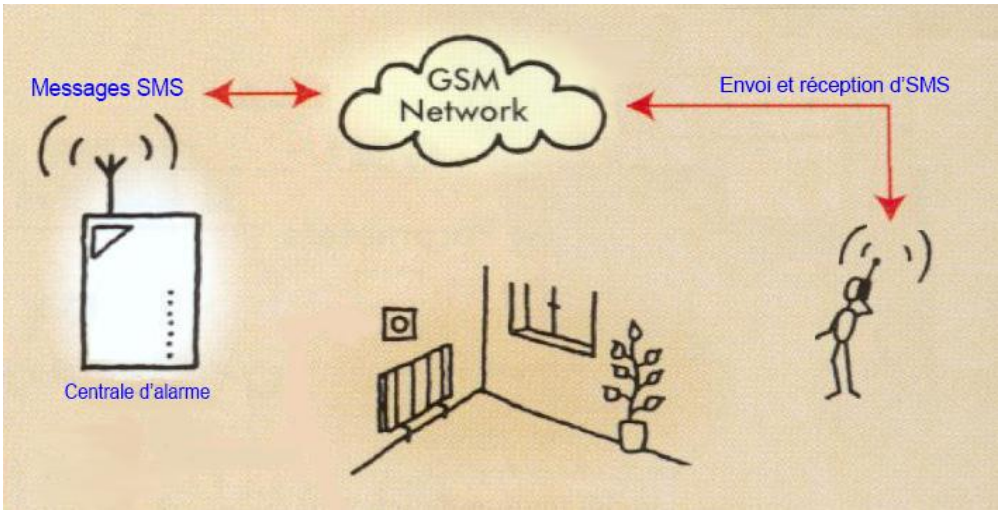


Figure 2.10 : Contrôle SMS de la centrale d'alarme avec l'utilisateur

Le service contrôle SMS nous permet de réaliser la surveillance des équipements électriques partagés dans la maison.

L'avantage du service contrôle SMS est qu'il est fonctionnel lors de coupure de la connexion Internet ou de la coupure du courant. Nous utilisons l'infrastructure GSM existante. Les SMS envoyés peuvent être reçus si nous utilisons une carte SIM et un module GSM. Dans le cas du produit alarme Comelit, nous utilisons le module 3G pour l'envoi et la réception des SMS. Nous avons utilisé les fonctions du Driver 3G pour la configuration de l'envoi et la réception des SMS et les traiter avec le service Contrôle SMS localisé dans la couche Middleware et spécifiquement dans le service de notification 'Telephony'.

2.3.2.1 Détail technique du service contrôle SMS

Pour contrôler le produit alarme via un téléphone mobile ou tout autre appareil utilisé pour l'envoi de SMS, l'utilisateur peut envoyer des requêtes contenant des actions de commandes. Le module envoie une réponse à la suite de l'action effectuée. Les commandes s'écrivent sous la forme suivante:

Question: « code action attribut »

Réponse: « action: réponse »

Si le code d'écriture SMS n'est pas valide, le produit alarme ne répond pas, mais si la commande est incomplète, il répond par « ERROR ».

Le tableau 5 représente les mots clé des requêtes du service SMS contrôle.

Tableau 5 : Identification des termes techniques pour le service contrôle SMS

Terme technique	Description
code	Le code utilisateur qui l'identifie de manière unique
action	la commande que vous souhaitez effectuer
attributs	si elle est présente, elle représente la liste des zones touchées, ils sont séparés par un espace
réponse	la réponse qui suit l'action requise

Le service contrôle SMS est programmé pour réaliser les fonctionnalités suivantes :

- Renvoyer l'état des zones.
- Activer le système d'alarme sur les secteurs désigné.
- Désactiver l'alarme sur les secteurs spécifiés.
- Renvoyer l'état des secteurs désigné.
- Renvoyer l'état des sorties indiquées.
- Activer/ désactiver les sorties spécifiées (s'ils sont présent).
- Demander le crédit restant sur la carte SIM.
- Arrêter l'alarme.

Dans ce qui suit, nous détaillons le principe de communication du service contrôle SMS et sa relation avec le Driver 3G.

2.3.2.2 Principe SMS contrôle et sa relation Driver 3G

Le service contrôle SMS, utilise le Driver 3G pour recevoir et envoyer des SMS. Les requêtes de SMS reçues de l'utilisateur sont traité par le service de notification 'Telephony'. Ce dernier nous informe de la réception des SMS et des événements survenus. Nous traitons le

message reçu selon le prototype de question (code action attribut) du service contrôle SMS. Selon chaque type de commande reçu nous réalisons un traitement spécifique à l'action demandé. Par la suite nous envoyons un SMS contenant une réponse à la demande effectuée via la fonction **Send_SMS()** du Driver 3G.

Dans la figure 2.11, nous représentons la communication du service Contrôle SMS et son positionnement dans le service 'Telephony' ainsi que sa communication avec le Driver 3G et les appels des fonctions d'envoi SMS et de réception SMS, ainsi que la fonction de **Send_USSD()** qui est utilisé pour connaitre le solde restant dans la carte SIM du produit alarme.

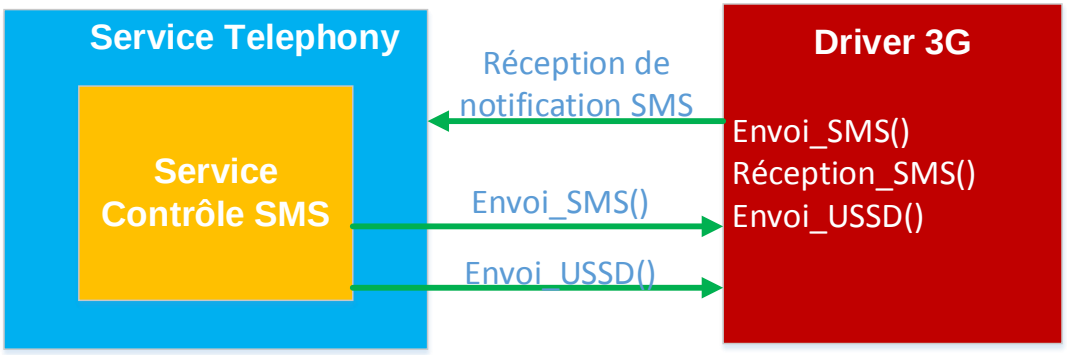


Figure 2.11 : Communication service contrôle SMS et Driver 3G

Dans ce qui suit, nous détaillons la conception réalisée de notre Driver 3G ainsi que du service contrôle SMS.

2.4 Conception du driver 3G et du service Contrôle SMS

Nous allons présenter la conception de notre Driver 3G et de l'application Contrôle SMS, par des organigrammes qui peuvent clarifier leurs fonctionnements.

2.4.1 Organigramme du Driver Module 3G

Le Driver 3G est basé essentiellement sur 2 fonctionnalités élémentaires :

- Une fonction **SendAtCheck()** pour l'envoi des AT commandes, puis la récupération de la réponse du module 3G.
- Une fonction **UARTPoll()** qui détecte les événements externes. Cette fonction qui détecte les événements envoyés par le module 3G à travers l'UART. A la différence avec la fonction SendATCheck(), cette fonction n'envoie jamais d'AT commande.

Nous allons décrire ces deux fonctionnalités par des organigrammes pour bien comprendre leur fonctionnement.

2.4.1.1 Organigramme de la fonction SendAtCheck()

Dans l'organigramme de la figure 2.12, nous allons décrire le fonctionnement de la fonction **SendAtCheck()** via un exemple réel d'une AT commande qui réalise un appel téléphonique.

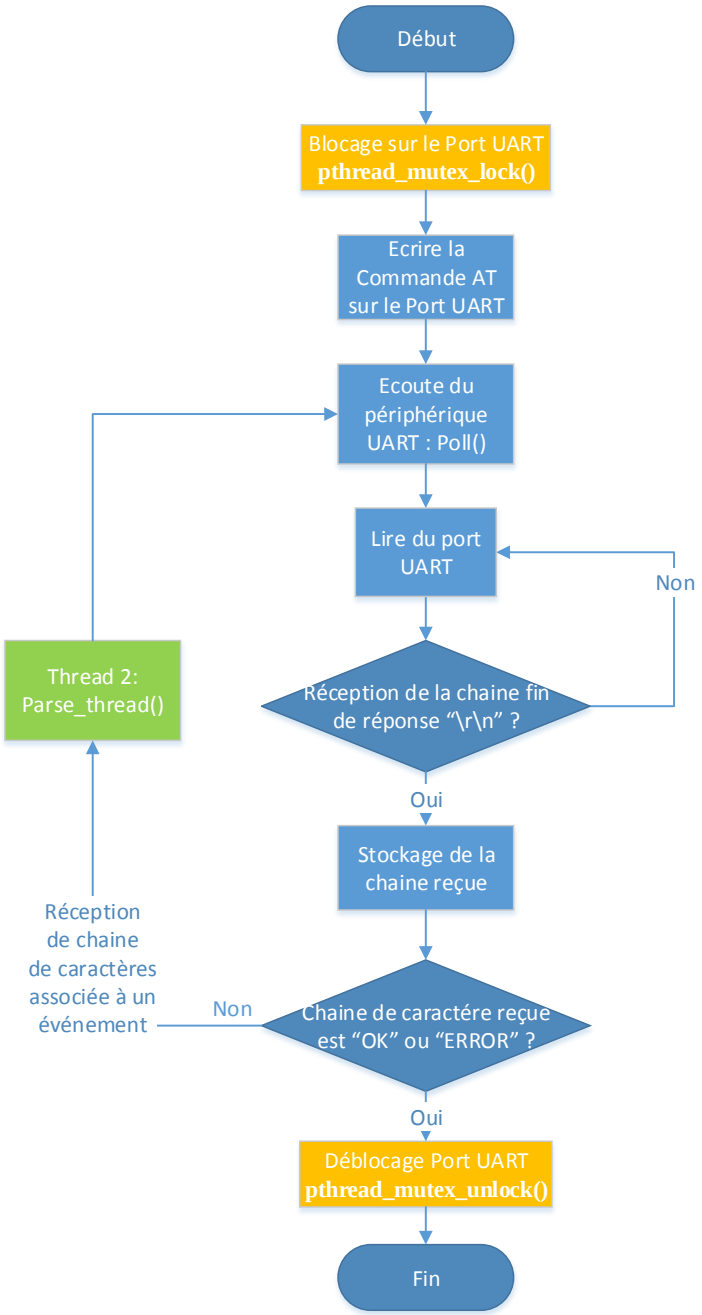


Figure 2.12: Organigramme de la fonction SendATCheck()

Tout d'abord, avant d'utiliser le Port UART, nous avons réalisé un blocage avec la fonction **pthread_mutex_lock()** pour le protéger contre l'utilisation d'une autre fonction. Par la suite on écrit la commande AT suivante pour appeler mon numéro par exemple « ATD55940854; ». Nous attendons ensuite la réponse du module 3G via le port UART avec la fonction **Poll()** qui permet d'écouter le trafic sur le port UART.

Lors de la réception d'une réponse, nous lisons cette chaine de caractères, et nous vérifions si elle se termine par « \r\n » qui marque la fin d'une chaine de caractères en provenance du module 3G. Si non on répète la procédure de lecture jusqu'à la réception de toute la chaine.

Nous vérifions cette chaine reçue, si elle est « OK » ou « ERROR ». Si non, il s'agit d'une chaine de caractères associée à un événement asynchrone externe. Pour cette raison, nous passons cette chaine de caractères au thread 2 spécifique à la détection des événements externes **Parse_thread()** via la fonction **Sem_Post()**. On retourne ensuite à l'écoute sur le port UART pour recevoir la réponse attendu du module 3G et en fin de ce cycle on débloque le port UART avec la fonction **pthread_mutex_unlock()**.

Dans l'organigramme de la figure 2.13, nous détaillons le fonctionnement de la fonction du Thread 2 **Events_Processing()**.

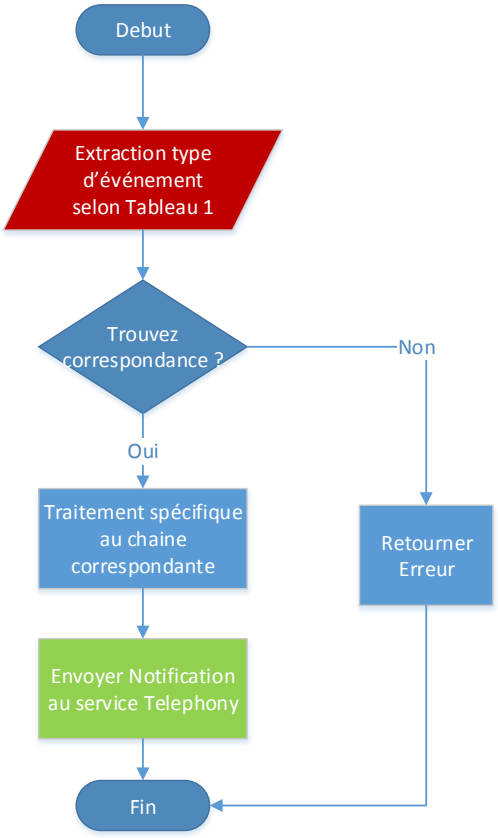


Figure 2.13 : Organigramme du mécanisme Events_Processing()

La fonction **Events_Processing()** du thread 2 **Parse_thread()** prend en paramètre la chaine de caractères reçue. Elle la compare à la liste de chaines de caractères indiquée dans le tableau 1. Lorsqu'elle trouve une égalité elle réalise alors un traitement spécifique à cet événement, et par la suite elle envoie une notification de cet événement reçu à la couche Middleware au service de « Telephony ».

2.4.1.2 Organigramme du Thread UARTPoll()

Dans l'organigramme de la figure 2.14, nous décrivons le fonctionnement de la fonction Thread 1 **UARTPoll()** et sa communication avec le Thread 2 **Parse_thread().**

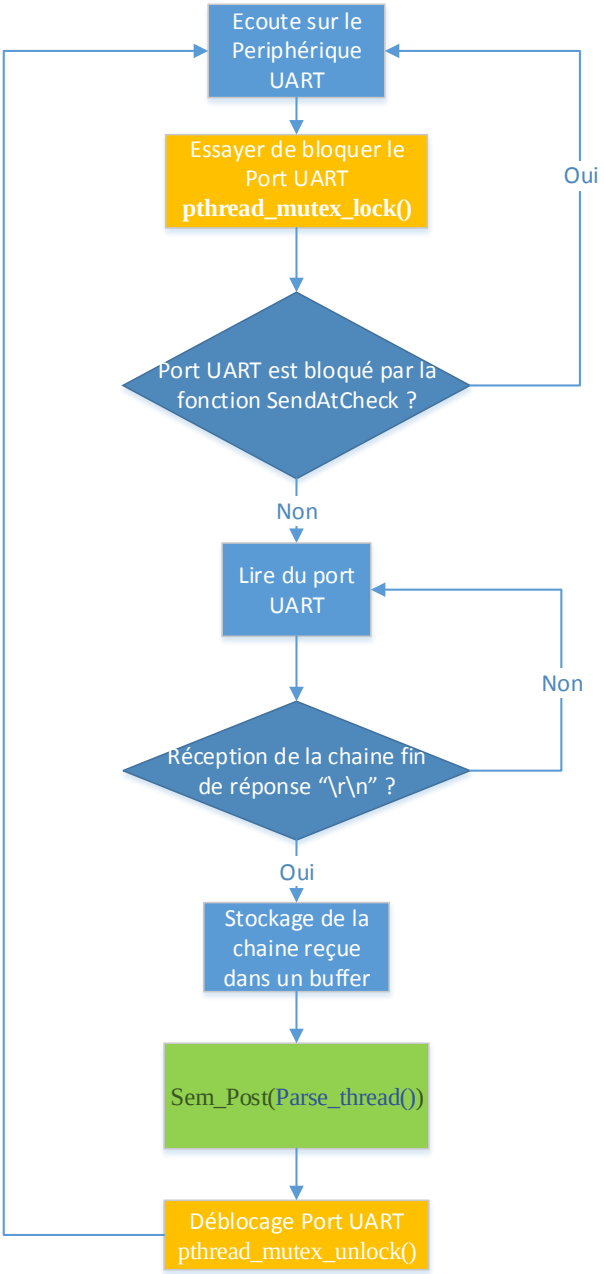


Figure 2.14 : Organigramme du mécanisme UART_POLL

L'objectif du Thread **UART_Poll()** est essentiellement l'écoute du port UART pour détecter la réception d'une chaine de caractères.

Lorsque le thread détecte qu'il y a une chaine de caractères reçue. Dans ce cas, il essaye de bloquer le Port UART pour le protéger contre une autre utilisation à l'aide de la fonction **pthread_mutex_trylock(),** mais lorsqu'elle le trouve déjà bloqué par la fonction **SendAtCheck()** elle sort directement de la boucle et retourne à l'état d'écoute du port. Dans le cas où elle trouve le port UART libre, elle lit du port jusqu'à la rencontre de la chaine '\r\n' qui marque la fin de la réponse. Elle renvoie ensuite cette chaine à la fonction **Events_Processing()** du thread 2 pour détecter l'événement externe décrit précédemment. Enfin, elle débloque le port UART avec la fonction **pthread_mutex_unlock()** et retourne à l'écoute.

Après la description des organigrammes nécessaires du driver 3G. Nous passons, à la description du service Contrôle SMS.

2.4.2 Organigramme du service Contrôle SMS

Dans l'organigramme de la figure 2.15, nous décrivons un exemple de traitement du service contrôle SMS. Il s'agit de la commande de requête « demande de crédit » de la carte SIM du produit alarme.

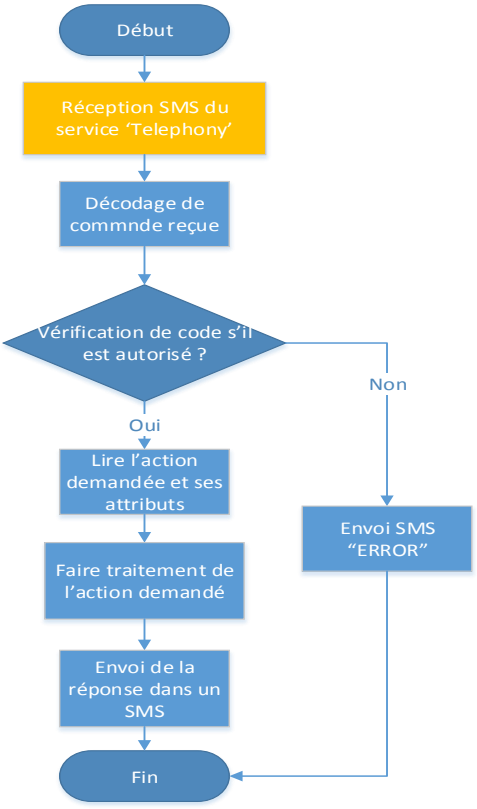


Figure 2.15 : Organigramme service contrôle SMS

Nous allons décrire cet organigramme via un exemple réel d'une commande de demande crédit, cette commande permet de savoir le crédit restant dans la carte SIM du produit alarme.

Pour réaliser une requête de demande de crédit, l'utilisateur doit envoyer la requête suivante : « code credit »

Tout d'abord le module 3G reçoit un SMS, et on détecte la réception d'un événement. La fonction **Events_Processing()** du thread 2 déterminera son type qui est la réception d'SMS. Dans ce cas, nous envoyons une notification au service « Telephony » lui informant qu'il s'agit de la réception d'SMS en lui fournissant tous les détails nécessaires (date, contenue SMS, numéro de l'envoyant). Après la réception de ces détails par le service « Telephony », nous déterminons les chaines « code », « action » et « attribut » s'il existe de la requête reçue jusqu'à la détection de la fin de du SMS. Nous détectons qu'il s'agit de l'action « Credit ». Nous réalisons une fonctionnalité du Driver 3G qui envoie l'AT commande responsable de la requête USSD. Nous retournons la réponse de la fonction **Send_USSD()** à notre utilisateur via la fonctionnalité d'envoi SMS du Driver 3G sous la forme « crédit: Réponse_USSD ».

Dans la section 5, nous présentons les résultats de test et validation les fonctionnalités développées ainsi que les threads développés.

2.5 Tests et validation

Dans cette partie du chapitre, nous présentons quelques tests de fonctionnement du Driver 3G et de l'application contrôle SMS développés.

La phase de test est effectuée au fur et à mesure du développement du Driver. En effet, lorsqu'on ajoute une nouvelle fonctionnalité, on la valide avec les tests convenables. Lors de l'exécution du test et à l'aide des traces ajoutées dans le code, nous pouvons cerner les parties du code valides et les fonctionnalités parcourues.

Toutes les traces d'exécution de notre code et de tout le code de l'application dans l'alarme sont gérées par un service de la couche Driver. Ce service permet d'organiser l'affichage des traces du code avec le prototype suivant :

[Temps][Couche][Fichier source][La fonction en exécution][Ligne d'exécution][Trace]

C'est une suite de fonctionnalités « #define » qui permettent de définir des prototypes à utiliser lors de l'affichage. Ils utilisent des couleurs différentes afin de distinguer entre les traces d'information, des erreurs et des traces utilisées dans les fichiers de tests.

2.5.1 Test et validation du Driver 3G

Nous avons réalisé un fichier de tests relatif au driver 3G, pour valider ses différentes fonctionnalités. La figure 2.16 illustre ce test.

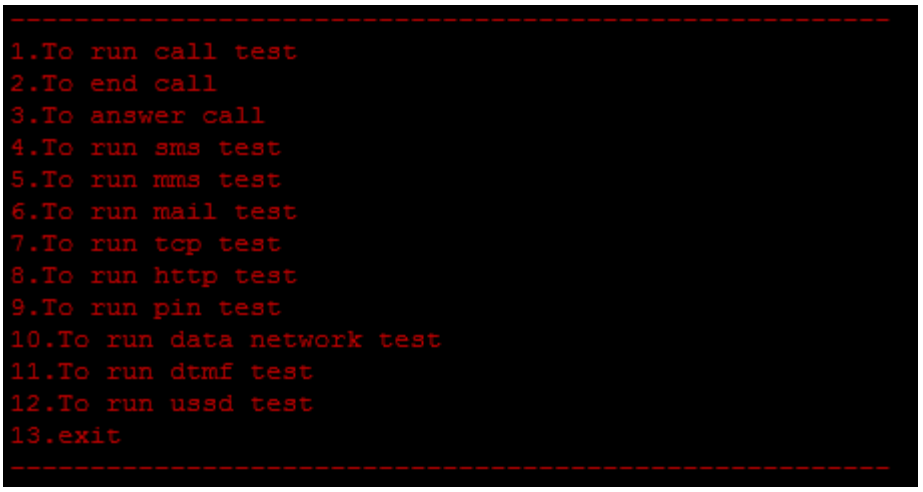


Figure 2.16 : Test Global

On peut à chaque fois choisir l'une des fonctionnalités indiquées dans la figure 2.16. Chacun de ces choix fait appel à une des fonctionnalités du Driver 3G.

2.5.1.1 Test et validation de fonctionnalités

Dans cette partie de test et validation du Driver 3G, nous allons présenter les résultats de test des fonctionnalités suivantes:

- Réalisation d'un appel téléphonique.
- Envoi d'un E-MAIL.

2.5.1.1.1 Test et validation d'un Appel Téléphonique

Dans le menu du choix global du Driver 3G décrit précédemment, nous avons choisi de réaliser un test d'appel d'un numéro, on sélectionne alors « To run call test » de la figure 2.16. La figure 2.17 montre le test réalisé et les traces indiquant l'état de fonctionnement du Driver.

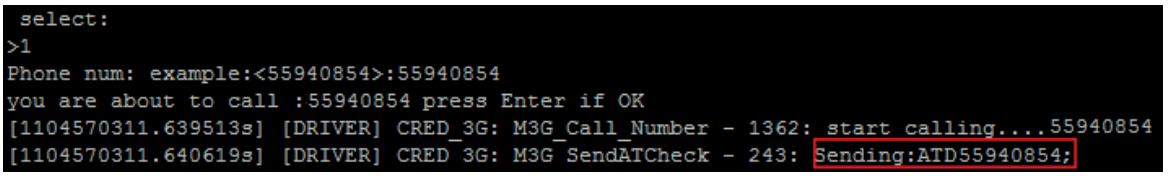


Figure 2.17 : Réalisation d'un appel téléphonique

Nous remarquons l'envoi de l'AT commande ATD suivi du numéro. L'appel est ensuite réalisé.

2.5.1.1.2 Test et validation de l'envoi d'un E-MAIL

Dans cette partie de test, nous réalisons l'envoi d'un e-mail. Nous devons d'abord configurer l'APN (Access Point Name) spécifique à l'opérateur téléphonique afin de réaliser une connexion au réseau internet et avoir une adresse IP. La figure 2.18 montre la configuration réalisée pour avoir une adresse IP.

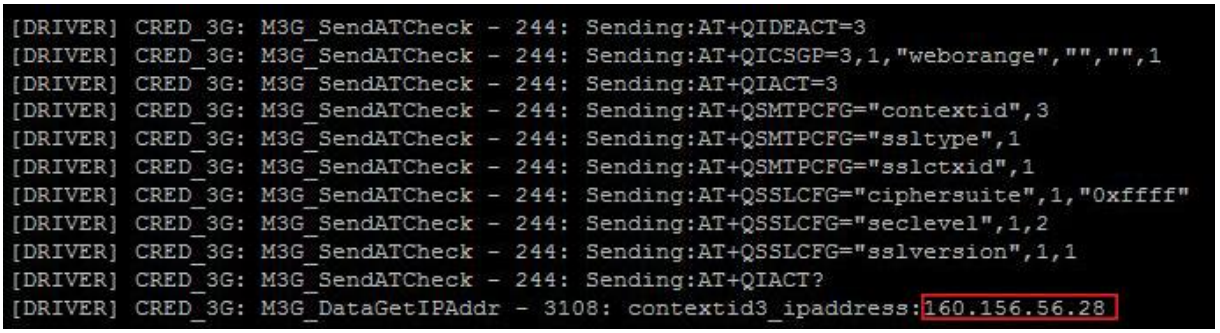


Figure 2.18 : Configuration pour avoir adresse IP

Par la suite, nous pouvons effectuer l'envoi des AT commandes pour configurer l'envoi du Mail. Nous devons configurer l'adresse du serveur et son port (exemple gmail: srvaddr : smtp.gmail.com, srvport : 465), l'adresse mail de l'émetteur, son nom et son mot de passe (nous avons caché le mot de passe de l'email de test Comelit).

La figure 2.19 montre la configuration du serveur Gmail et de l'émetteur de l'Email.

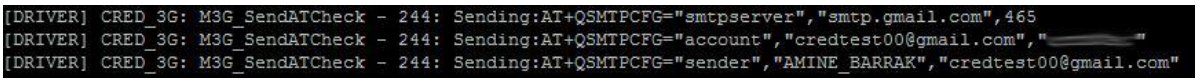


Figure 2.19 : Configuration du serveur Gmail et de l'émetteur de l'Email

Nous devons aussi configurer l'adresse mail du destinataire, le sujet de l'email, le contenu de l'email et le chemin de l'image et du fichier en attachement. La figure 2.20 montre la configuration nécessaire du contenu de l'Email et de l'envoi.

```
[DRIVER] CRED_3G: M3G_SendATCheck - 244: Sending:AT+QSMTPCLR
[DRIVER] CRED_3G: M3G_SendATCheck - 244: Sending:AT+QFDEL="RAM:*"
[DRIVER] CRED_3G: M3G_SendATCheck - 244: Sending:AT+QSMTPDST=1,1,"aminebarrak@live.com"
[DRIVER] CRED_3G: M3G_SendATCheck - 244: Sending:AT+QSMTPSUB=0,"TEST SMTP"
[DRIVER] CRED_3G: M3G_SendAT - 337: Sending:AT+QSMTPBODY=1,10,300
[DRIVER] CRED_3G: M3G_Send_Mail - 1701: body_text uploading
[DRIVER] CRED_3G: M3G_SendAT - 337: Sending:AT+QFUPL="RAM:test text file.txt",520,200,1
[DRIVER] CRED_3G: M3G_AppendFileToRam - 1770: file uploading
[DRIVER] CRED_3G: M3G_SendATCheck - 244: Sending:AT+QSMTPATT=1,1,"RAM:test_text_file.txt"
[DRIVER] CRED_3G: M3G_SendAT - 337: Sending:AT+QFUPL="RAM:pics.jpg",64561,200,1
[DRIVER] CRED_3G: M3G_AppendFileToRam - 1770: file uploading
[DRIVER] CRED_3G: M3G_RxReadOnce - 135: M3G_RxReadOnce Timout
[DRIVER] CRED_3G: M3G_SendATCheck - 244: Sending:AT+QSMTPATT=1,2,"RAM:pics.jpg"
[DRIVER] CRED_3G: M3G_SendATCheck - 244: Sending:AT+QSMTPPUT=300
```

Figure 2.20 : Configuration du contenu de l'Email et son envoi

Après une configuration juste et complète des paramètres d'envoi du Mail et des différentes configurations avec l'envoi des AT commande correspondantes, le mail arrive convenablement à sa destination avec les fichiers d'attachement. Par la suite nous recevons sous forme d'événement que le mail est bien arrivé au destinataire. Dans ce cas, nous notifions le service Telephony que l'Email est bien envoyé au destinataire. La figure 2.21 montre la réception de l'accusé de réception de l'Email et la notification du service 'Telephony'.

```
[DRIVER] CRED_3G: M3G_SendATCheck - 244: Sending:AT+QSMTPCLR
[DRIVER] CRED_3G: M3G_SendATCheck - 244: Sending:AT+QFDEL="RAM:*"
[DRIVER] CRED_3G: M3G_EmailOutProceessing - 250: mail send OK
[MW] CRED_TELEPHONY: Ts_Notify_Thread - 1869: CRED_TS_EMAIL
[MW] CRED_TELEPHONY: Mail_EventProcessing - 1697: CRED_TS_OUTGOING:email send OK
```

Figure 2.21 : Accusé de réception de l'Email

La figure 2.22 montre la bonne réception de l'Email au destinataire avec les fichiers d'attachement.

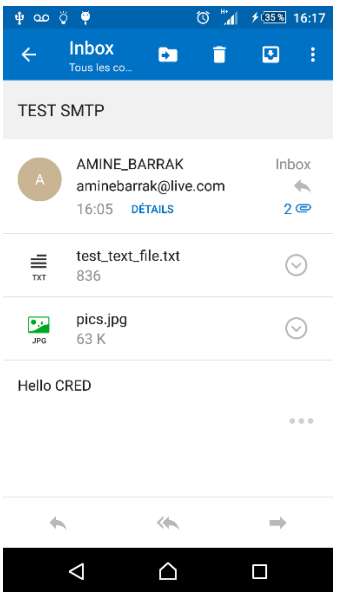


Figure 2.22 : Réception d'un l'Email

2.5.1.2 Test et validation des événements

Dans cette section nous présentons le résultat de test et validation de deux exemples de réception d'événements externes et qui sont détectés par la fonction **Events_Processing()** du thread 2 :

- Enlèvement de la carte SIM.
- Réception de code DTMF.

2.5.1.2.1 Test et validation de l'enlèvement de la carte SIM

Lorsqu'on enlève la carte SIM du module 3G, un événement asynchrone sera envoyé vers l'UART. Cet événement sera alors détecté et traité par le Driver 3G à travers les deux threads UART_Poll() et Parse_Thread().

La figure 2.23 indique l'état de changement et d'enlèvement de la carte SIM ainsi que les traces d'exécution du code.

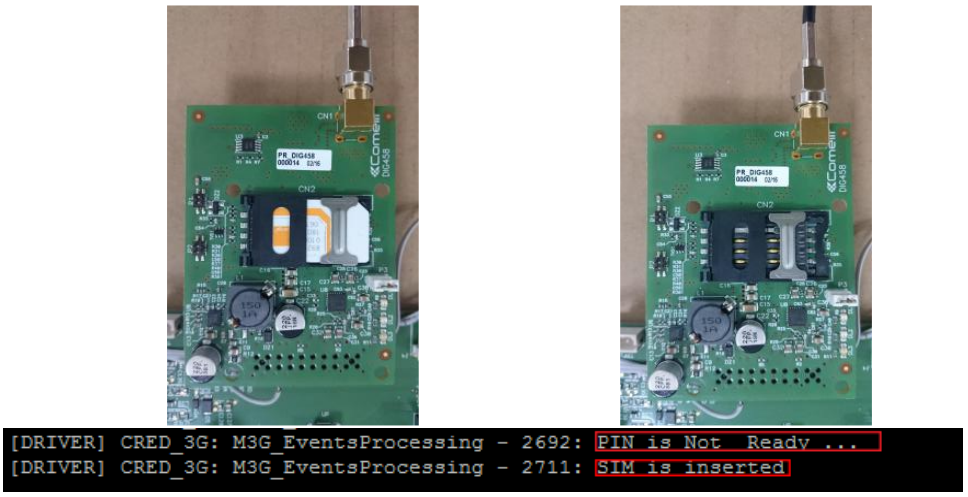


Figure 2.23 : Test d'insertion et d'enlèvement de la carte SIM

Dans la trace de la figure 2.23, lors du retrait de la carte SIM, le module génère un événement asynchrone et la fonction **Events_Processing()** du thread **Parse_Thread()** le détecte. Dans ce cas nous affichons la trace indiquée dans la figure « Pin is Not Ready … ». Lorsqu'on retourne la puce, il s'agit d'un événement asynchrone, la fonction le détecte et nous affichons la trace « SIM is inserted ».

2.5.1.2.2 Test et validation de réception de code DTMF

Le module 3G supporte la réception des caractères DTMF. Pour bénéficier de la réception du code DTMF, nous réalisons son initialisation avec une configuration d'AT commande.

Lorsque nous réalisons un appel téléphonique vers un appareil mobile. Nous pouvons détecter les codes DTMF réalisé dans l'appareil mobile. La figure 2.24 montre la réalisation d'un appel téléphonique et nous tapons les numéros 7 et 8 sur le clavier.

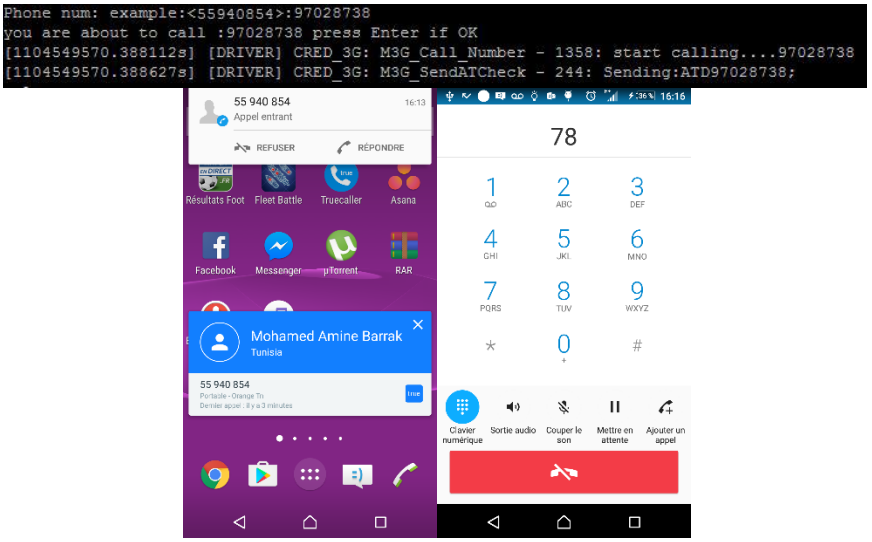


Figure 2.24 : Test d'envoi code DTMF lors d'un appel téléphonique

Dans la figure 2.25 nous montrons la réception du code DTMF envoyé par l'appareil mobile. Nous détectons les numéros 7 et 8 et le driver du module 3G envoie une notification au service 'Telephony' des codes DTMF reçus.

```
[DRIVER] CRED_3G: M3G_DTMFInProceessing - 2367: DTMF=7
[MW] CRED_TELEPHONY: Ts_Notify_Thread - 1919: CRED_TS_DTMF
[MW] CRED_TELEPHONY: DTMF_EventProcessing - 1782: CRED_TS_INCOMMING DTMF:7
[DRIVER] CRED_3G: M3G_DTMFInProceessing - 2367: DTMF=8
[MW] CRED_TELEPHONY: Ts_Notify_Thread - 1919: CRED_TS_DTMF
[MW] CRED_TELEPHONY: DTMF_EventProcessing - 1782: CRED_TS_INCOMMING DTMF:8
```

Figure 2.25: Réception du code DTMF

2.5.2 Test et validation du service contrôle SMS

Dans cette partie de test du service contrôle SMS, nous détaillons quelques exemples de commandes d'action pour contrôler le produit alarme.

Le produit alarme installé dans une maison peut la décomposer en 8 secteurs et chaque secteur peut contenir au maximum 8 zones.

Nous détaillons dans cette section les résultats de test suivants :

- Activation d'alarme dans tous les secteurs (arm)
- Détection de l'état de tous les secteurs (status)
- Demande du crédit restant dans la carte SIM de l'alarme (credit)

2.5.2.1 Envoi de commande armer les secteurs

Lorsque l'utilisateur du produit alarme veut activer l'alarme dans tous les secteurs de la maison, il réalise la commande suivante :

```
1111 arm
```

La réponse de l'alarme contient le résultat de la requête reçue indiquant si l'alarme est activée dans tous les secteurs.

```
arm: PERFORMED ACTION
```

Dans la figure 2.26, nous présentons la réception du SMS contenant une commande « arm » envoyée par l'utilisateur du produit alarme.

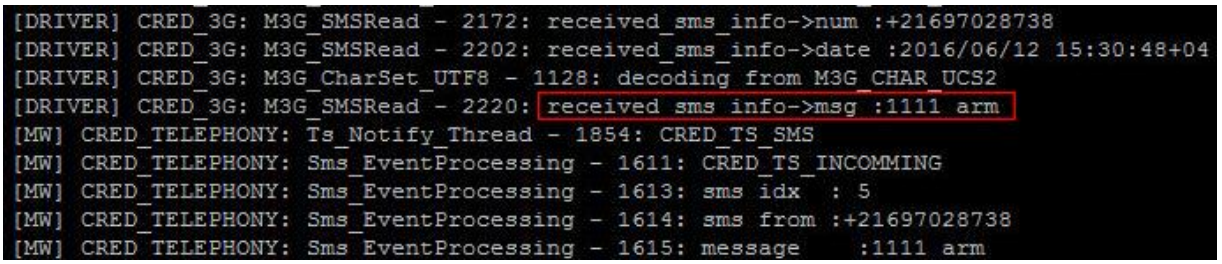


Figure 2.26 : Réception requête arm

La figure 2.27, présente le traitement réalisé pour répondre l'utilisateur de la requête « arm ».

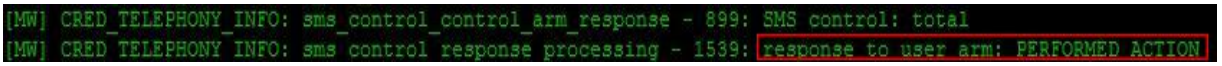


Figure 2.27 : Résultat de traitement requête arm

La figure 2.28, présente l'envoi du message de l'alarme vers l'utilisateur contenant la réponse de sa requête « arm ».

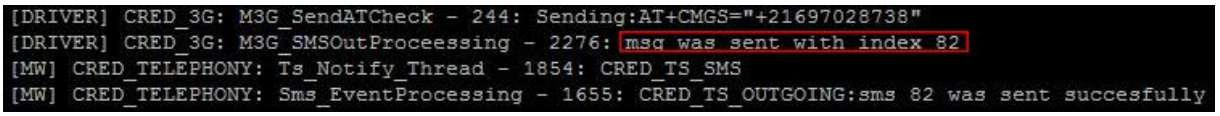


Figure 2.28 : Envoi du message contenant le résultat de requête arm

2.5.2.2 Envoi de commande état des secteurs

La commande état des secteurs renvoie l'état de tous les secteurs de la centrale d'alarme. La réponse contient les secteurs de la maison suivis de leurs états. Les états possibles des secteurs du produit alarme sont :

- X : le secteur d'alarme n'est pas encore prêt.
- I : indique que toutes les zones d'un secteur sont insérées.

- P : indique une insertion partielle des zones d'un secteur.
- D : le secteur d'alarme n'est pas activé.

Lorsque l'utilisateur du produit alarme veut consulter l'état des secteurs de la maison, il envoie la commande état des secteurs suivante :

```
1111 status
```

Le produit alarme réalise une vérification des secteurs de la maison et envoie une réponse indiquant l'état de chaque secteur.

```
status: 1 I 2 I 3 I 4 I 5 I 6 I 7 I 8 I
```

Dans la figure 2.29, nous présentons la réception du message de la requête état des secteurs.

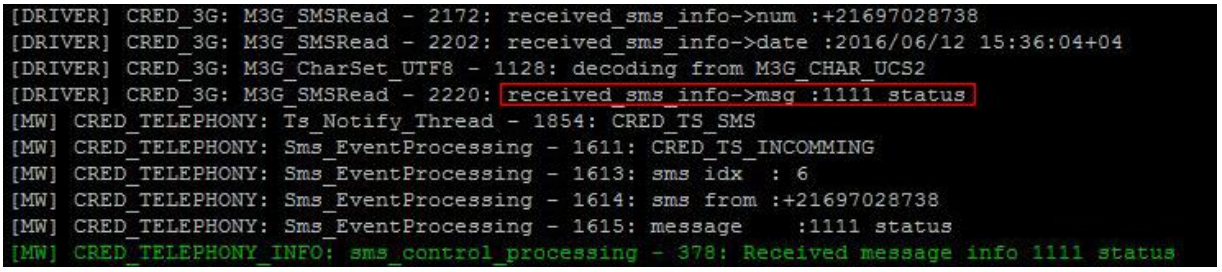


Figure 2.29 : Réception de la requête état des secteurs

La figure 2.30 présente le traitement réalisé afin de fournir à l'utilisateur la réponse de la requête « status ».

Figure 2.30 : Résultat du traitement de la requête état des secteurs

La figure 2.31 montre l'envoi d'un SMS contenant le résultat de la commande d'état des secteurs « status ».

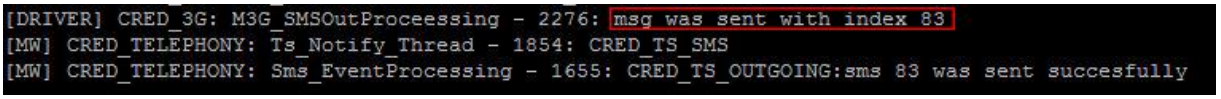


Figure 2.31 : Envoi du message contant le résultat de la requête état des secteurs

La figure 2.32 montre la trace d'envoi des SMS contenant les requêtes « arm » et « status » au produit alarme, ainsi que les résultats retournés.

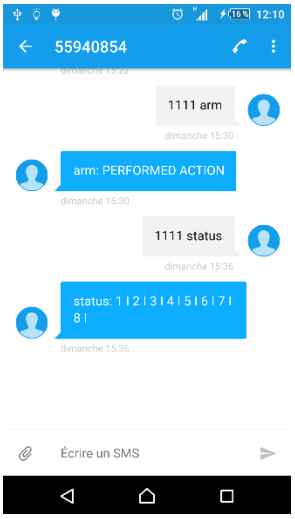


Figure 2.32 : Trace d'envoi des requêtes "arm" et "status" avec leurs résultats

2.5.2.3 Envoi de commande demande Crédit

L'utilisateur du produit alarme peut consulter le crédit restant dans la carte SIM du produit alarme via la requête « credit ». L'utilisateur envoie la commande suivante :

```
1111 credit
```

Le produit alarme réalise le traitement de la requête reçue et retourne un SMS à l'utilisateur contenant le résultat de la commande USSD pour savoir le crédit restant dans la carte SIM du produit alarme. L'exemple suivant montre un résultat de commande « credit ».

```
SUBMITTED FROM 40916: TIM: Traffic residue 14:14 EUR inc VAT, last
updated SMS 20/10 at 17:04
```

Dans la figure 2.33, nous présentons la réception d'un message contenant la requête crédit restant dans la carte SIM de l'alarme.

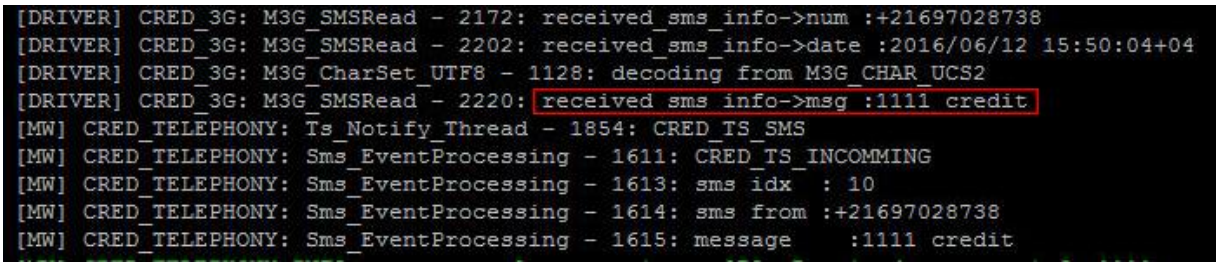


Figure 2.33 : Réception de la requête crédit

La figure 2.34 présente le traitement réalisé afin de fournir à l'utilisateur la réponse de la requête « credit ».

```
[DRIVER] CRED_3G: M3G_SendATCheck - 244: Sending:AT+CUSD=1,"*101#",15
[DRIVER] CRED_3G: M3G_CharSet_UTF8 - 1116: decoding from M3G_CHAR_GSM
[MW] CRED_TELEPHONY_INFO: sms_control_credit_response - 1427: *101#---------->Votre solde est de 4.765 dt

[DRIVER] CRED_3G: M3G_SendATCheck - 244: Sending:AT+CUSD=2
[MW] CRED_TELEPHONY_INFO: sms_control_response_processing - 1539: response to user Votre solde est de 4.76
2#
```

Figure 2.34 : Réalisation de commande USSD

La figure 2.35 montre l'envoi d'un SMS contenant le résultat de la commande crédit de l'alarme.

```
[DRIVER] CRED_3G: M3G_SMSOutProceessing - 2276: msg was sent with index 87
[MW] CRED_TELEPHONY: Ts_Notify_Thread - 1854: CRED_TS_SMS
[MW] CRED_TELEPHONY: Sms_EventProcessing - 1655: CRED_TS_OUTGOING:sms 87 was sent succesfully
```

Figure 2.35 : Envoi du message contenant le résulat de commande credit

La figure 2.36 présente la trace d'envoi de la requête « credit » ainsi que la réception du crédit restant dans la centrale d'alarme.

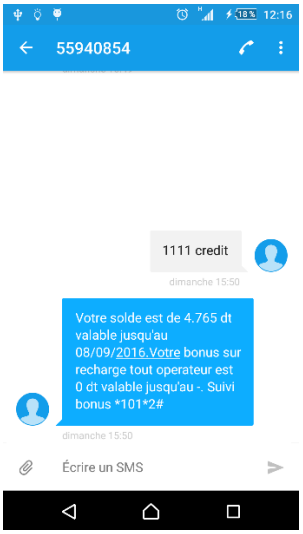


Figure 2.36 : Trace d'envoi de commande "credit" et le résultat

2.6 Conclusion

Dans ce chapitre, nous avons présenté le module 3G du produit alarme et les étapes de conception et de développement d'un driver pour la configuration de ce module 3G et du service contrôle SMS pour le contrôle à distance du produit alarme. Nous avons présenté en particulier les organigrammes détaillant le fonctionnement du Driver 3G et du service contrôle SMS. Nous avons détaillé également les tests de validation de certaines fonctionnalités développées dans le driver 3G. On cite la fonction de réalisation d'appel et envoi d'un e-mail. Pour la détection et le traitement d'événements, nous l'avons testé pour l'enlèvement de puce et réception de code DTMF. Nous avons également validé le service contrôle SMS. Dans le chapitre suivant nous allons détailler la seconde partie de notre projet de fin d'études qui concerne le développement de l'application Push Notification.

Chapitre 3

Conception et Développement de l'application Push Notification

3.1 Introduction

Dans ce chapitre, nous décrivons les étapes de développement du service push notification pour un système d'alarme. Nous présentons par la suite les étapes pour réaliser l'application Push Notification sur android. Nous détaillons les résultats de tests d'envoi des messages push du produit alarme vers l'application développée sur un smart phone android.

3.2 Présentation du mécanisme Push Notification

Dans cette section, nous présentons en premier lieu le concept général du mécanisme Push Notification, ensuite, nous décrivons le modèle existant de communication en utilisant l'API GCM (Google Cloud Messaging) du serveur Google pour la réalisation du Push Notification. Et nous allons finir par présenter le modèle d'architecture de notre application.

3.2.1 Concept Push Notification et GCM

Pour garder le contact actif avec les clients du produit alarme, nous devons leur apporter de l'information utile en temps réel. La notion du temps réel est alors primordiale dans notre application.

Les applications de notification doivent être rapides, concises et répondent à un besoin urgent où une information importante à communiquer avec les utilisateurs. L'application Push est souvent décrite comme un canal de communication privilégié pour les entreprises. Son principe est de notifier les utilisateurs en temps réel. L'application ne doit pas être en cours d'exécution pour recevoir une notification[15].

Cette application mobile possède de nombreux avantages :

- Les notifications bénéficient d'un important taux d'ouverture et d'utilisation.
- L'envoi de notification est gratuit.
- Une invitation à fermer le pop-up ou à découvrir plus de contenu.

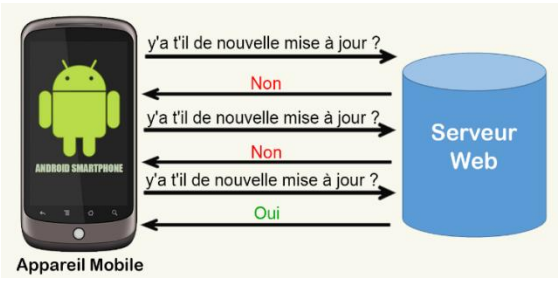
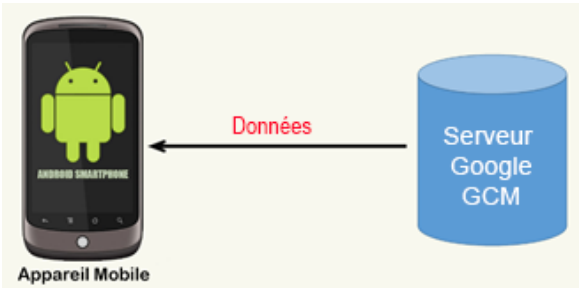
Google Cloud Messaging (GCM) représenté par le sigle de la figure 3.1 est un service pour envoyer et recevoir les notifications Push vers et à partir des applications Android. C'est est un service ouvert qui permet aux développeurs d'envoyer des messages entre les serveurs et les applications clientes.

Figure 3.1 : Google Cloud Messaging

GCM est utile lorsque de nouvelles données sont disponibles sur le serveur, il les gère dans les files d'attente de messagerie et délivre ces messages à l'application cible. L'API GCM représente une particularité par rapport aux autres applications qui font une demande au serveur sur un intervalle de temps régulier.

Dans le tableau 6 nous allons réaliser une comparaison entre une application qui utilise le GCM et une autre avec la méthode de Polling pour souligner les avantages offerts par l'API GCM.

Tableau 6 : Comparaison entre méthodes GCM et Polling

Réception de données avec la méthode de Polling	**Réception de Données avec la méthode GCM**
 **Figure 3.2: Méthode Polling**	 **Figure 3.3 : Méthode GCM**
• Simple à mettre en œuvre. • L'appareil interrompe périodiquement le serveur pour de nouvelles données. • Toujours maintenable lors d'une mise à jour.	• Connexion initialisé par le Serveur Web. • Connexion constante. • Moins de consommation éléctrique. • Plus difficile à mettre en œuvre.

D'après le tableau 6, la méthode GCM permet d'économiser d'une façon considérable, la consommation électrique depuis la batterie de l'utilisateur. En effet, on évite les demandes d'interrogation régulière au serveur pour chercher s'il y a de nouvelles données.

GCM ne fournit aucune interface graphique et aucun algorithme prédéfini de prise en charge des messages. Au lieu de cela, il passe simplement le message brut directement à l'application qui a toute la liberté sur la manière de le traiter. Par exemple, l'application peut afficher une notification système ou une interface personnalisée. Elle peut aussi simplement synchroniser des données sans rien afficher.

Nous allons décrire dans ce qui suit les étapes nécessaires pour réaliser et activer l'API du GCM dans le projet Google.

3.2.2 Architecture générale de la communication GCM

Le mécanisme général de communication du serveur GCM décrit par la figure 3.4 se résume par le fonctionnement suivant [15]:

1. L'appareil Android envoie le Sender ID et l'ID d'application au serveur GCM pour l'enregistrement.
2. Lors de l'inscription réussie, le serveur GCM produit un identificateur appelé TOKEN spécifique pour l'appareil Android.
3. L'appareil Android envoie ce TOKEN d'inscription au serveur local.
4. Le serveur local stocke le TOKEN dans la base de données pour une utilisation ultérieure.

   a) Chaque fois qu'une notification est déclenché par le site, le serveur envoie le message à notifier au serveur de GCM avec le TOKEN enregistré.
   b) Le serveur GCM envoie ce message à l'appareil particulière en utilisant ce TOKEN.

Dans le tableau 7 nous définissons les termes techniques utilisés précédemment.

Tableau 7 : Définition des termes techniques du GCM

Termes techniques	Description
Sender ID	Une valeur numérique unique générée lors de la configuration du projet GCM.
API Key	Une API enregistrée sur le serveur d'application qui fournit un accès au service Google.
ID d'application	Un identifiant pour l'application client.
TOKEN	Un identifiant donné par GCM à l'application client qui lui permet de recevoir des messages. Il se compose de 152 caractères.

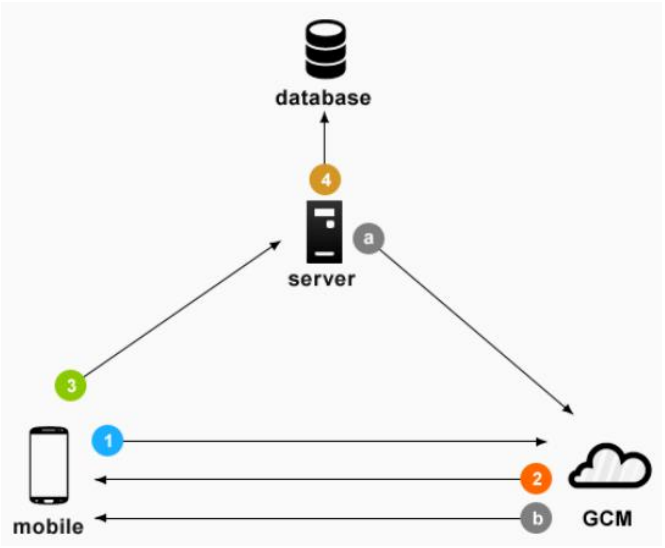


Figure 3.4 : Architecture de communication générale du mécanisme Push Notification

Dans la partie suivante, nous décrivons notre propre solution. La spécificité de notre architecture du mécanisme Push Notification est qu'elle ne contient pas de serveur suite à la contrainte Matériel qui impose de ne pas utiliser un serveur. Le produit alarme contient assez de charge et principalement huit caméra en streaming pour cette raison il ne supportera pas un serveur de plus. Tout le traitement des utilisateurs est réalisé au niveau de l'alarme.

3.2.3 Architecture proposée

Nous avons réalisé l'architecture de notre application Push Notification qui est décrite par la figure 3.5.

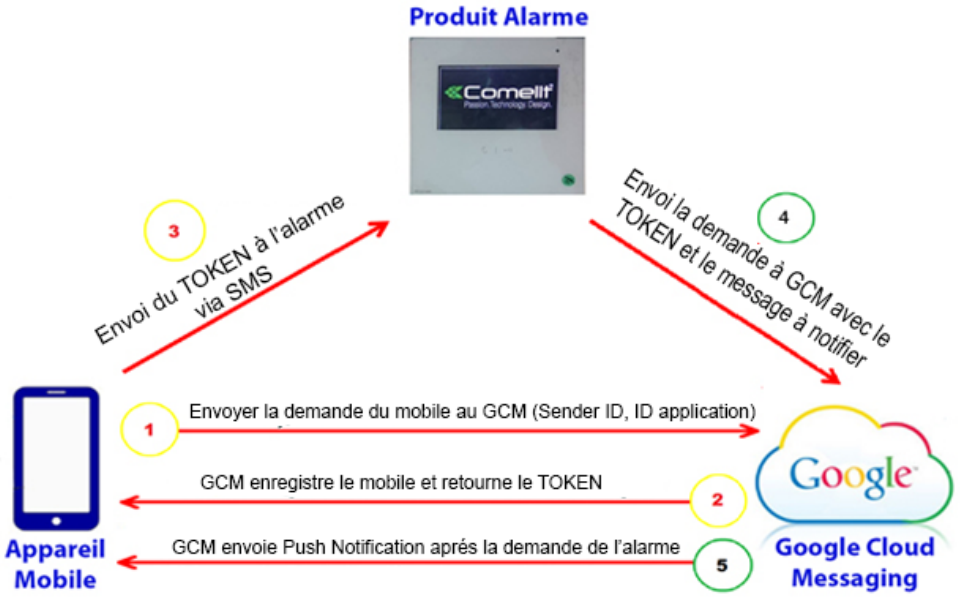


Figure 3.5 : Nouvelle architecture du mécanisme Push Notification

Pour réaliser le Push Notification à la clientèle de Comelit, nous allons détailler les étapes décrites dans la Figure 3.5 :

1. L'appareil mobile envoie le Sender ID et l'ID de l'application au serveur GCM pour faire l'enregistrement.
2. Après un enregistrement réussi, le serveur GCM retourne le TOKEN spécifique à l'appareil mobile.
3. L'appareil Android envoie le TOKEN à la boite d'alarme.
4. Lors du déclenchement d'alarme, la boite d'alarme utilise le TOKEN enregistré pour chaque client et envoie une demande au GCM qui contient le message à notifier.
5. Après avoir obtenu une demande d'envoyer Push Notification, GCM envoie la notification d'alarme à l'appareil Android spécifique au TOKEN.

Notre propre mécanisme d'envoi de Push Notification, nous a fourni une amélioration par rapport aux applications qui utilisent des serveurs et du stockage dans une base de données. En effet, nous avons optimisé le trafic réseau, ainsi que la connexion vers la base de données et les requêtes SQL nécessaires permettant de choisir les clients à notifier.

3.3 Présentation du développement du mécanisme Push Notification

Trois étapes sont nécessaires pour recevoir les notifications sous Android :

1. Créer un projet dans le "Google Developper Center" pour récupérer le Sender ID. Spécifier pour ce projet l'utilisation du service "Google Cloud Messaging for Android", et générer l'API Key.
2. Gérer la réception et le traitement des notifications dans l'application Android.
3. Développement du service Push dans le produit alarme qui est responsable de déclencher les notifications.

Nous détaillons dans les sections suivantes, chacune de ces étapes citées.

3.3.1 Projet Google et API Google Cloud Messaging

Google Clé API est nécessaire pour interagir avec le serveur GCM. En effet, il utilise cette clé afin d'identifier notre service Push du produit alarme. Pour avoir le Google clé API, nous devons créer un projet Google dans le "Google Developer Console" et activer l'API GCM.

Pour bénéficier du service Google Cloud Messaging, nous avons réalisé les étapes suivantes :

1. Avoir un compte Google gmail.
2. Créer un projet sous Google Developer Console.
3. Obtenir l'ID de l'application de notre projet Alarme.
4. Activer le service GCM pour notre projet Alarme.
5. Créer un Identifiant et choisir clé API, par la suite créer une clé serveur.
6. Obtenir l'API key pour l'utiliser dans notre service Push.

Dans la figure 3.6, nous illustrons la création du projet alarme pour bénéficier de l'API Google Cloud Messaging.

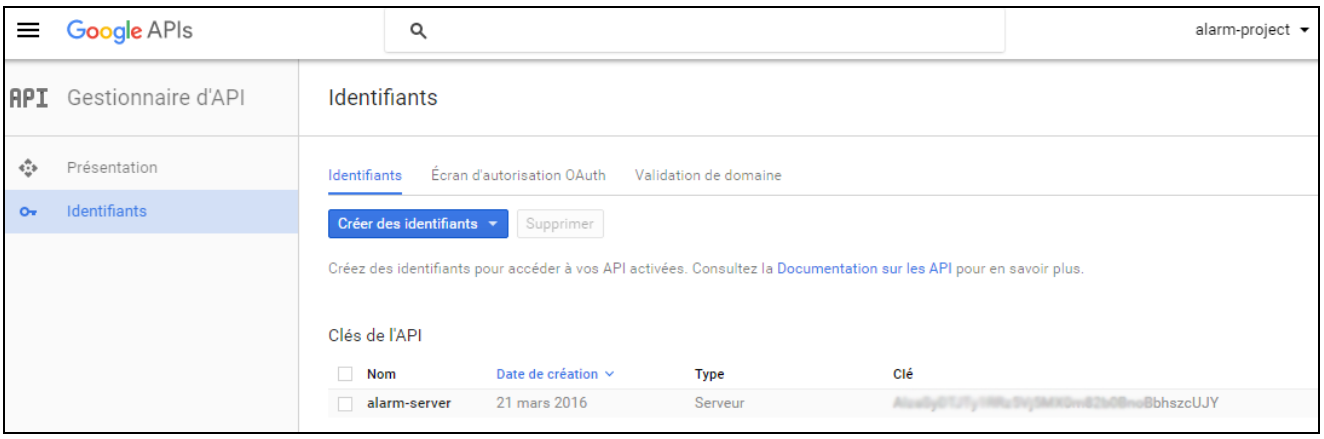


Figure 3.6 : Google Clé API

La figure 3.6 montre la création d'un projet Google, et l'ajout de l'API GCM à notre projet nommé « alarm-project ». Pour des raisons de confidentialité imposées par Comelit, la clé sera masquée.

Avec la figure 3.7, nous montrons la génération des Sender ID et du server API Key. Ensuite nous réalisons la génération d'un fichier Json qui contient des détails de configuration, tels que les clés et les identifiants, les services google que nous utilisons. Nous ajoutons ce fichier dans le projet Android de l'application Push Notification.

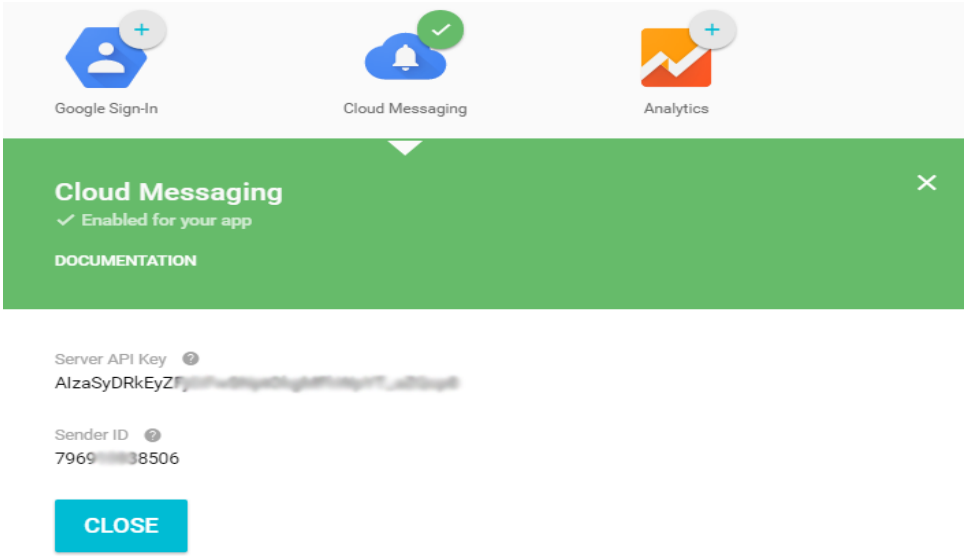


Figure 3.7 : Génération du Server API key

Nous avons masqué le Sender ID et le API Key pour des raisons de confidentialité.

3.3.2 Présentation de l'application client Android Push Notification

Pour développer l'application Android Push Notification, il faut que l'Android SDK (Logiciel development kit) renferme :

- Google Play Services
- Google APIs
- Le projet Google Play Services Lib en tant que dépendance Android

L'application Client Android est nécessaire dans notre projet pour recevoir le Push Notification. Elle a deux rôles essentiels, le premier est de se connecter à Google pour recevoir le TOKEN spécifique à l'appareil téléphonique. Le deuxième consiste à recevoir le message Push lors de son envoi depuis le GCM.

Nous avons réalisé l'application Client Android afin de fournir les tâches suivantes :

- Authentification pour accéder à l'application. L'utilisateur a trois tentatives pour s'identifier, si non l'application ne lui permettra plus de s'identifier. Après la récupération du TOKEN de Google, nous l'envoyons via un SMS à l'alarme.
- Enregistrement avec GCM avec la via la fonction getToken().
- Rafraichissement du TOKEN obtenu du serveur GCM après avoir effectué une désinscription puis une inscription au serveur GCM.

- Enregistrement local des informations nécessaires tel que le TOKEN à l'aide du service SharedPreferences. Dans notre application, il est nécessaire d'utiliser des variables qui doivent être gardées en mémoire même suite à une fermeture. Notre solution a portée sur l'utilisation du ShredPreferences qui est la solution la plus simple à implémenter.
- Création du bouton « Unregister » pour faire la désinscription du serveur GCM avec la fonction **DeleteToken()**.
- Création du bouton de rafraîchissement « Refrech » pour récupérer un nouveau TOKEN du serveur GCM. Dans ce cas, elle envoie vers l'alarme un SMS contenant son nouveau TOKEN.

L'application que nous avons conçue est dynamique et permet d'afficher des messages différents selon le type de notification reçu.

3.3.3 Présentation du service Push dans le produit alarme

Nous avons réalisé le service Push de manière qu'il soit flexible permettant de faire une mise à jour du TOKEN lors de son changement et d'envoyer la notification directement lors du besoin.

Le service Push se base essentiellement sur deux principales fonctionnalités :

- Mise à jour du TOKEN
- Réalisation du Push et manipulation de la table des utilisateurs.

Nous allons décrire avec détail les deux fonctionnalités du service Push dans ce qui suit.

3.3.3.1 Fonctionnalité de mise à jour TOKEN

La mise à jour des TOKENs spécifiques aux utilisateurs enregistrés dans l'alarme est réalisée via la réception de SMS contenant le nouveau TOKEN.

La figure 3.8 explique la relation du Service Push avec le service de notification 'Telephony' ainsi que l'utilisation des fonctions d'envoi et de réception d'un SMS du Driver 3G.

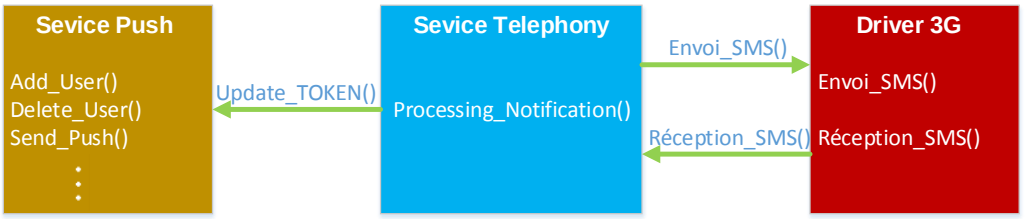


Figure 3.8 : Communication service Push avec service Telephony et Driver 3G

Lors de la détection d'une réception de message SMS, nous réalisons une vérification s'il s'agit d'un SMS spécifique au service Push ou un autre service qui utilise les SMS. L'utilisateur réalise son authentification et nous envoie un SMS pour enregistrer son TOKEN dans le produit alarme. Nous vérifions alors s'il s'agit d'un numéro déjà enregistré ou non. S'il est enregistré, nous demandons de nous envoyer son TOKEN afin de l'enregistrer dans l'alarme.

Ce service reste alors en écoute. Lorsque le service reçoit un SMS de mise à jour du TOKEN, le nouveau TOKEN écrasera l'ancien.

3.3.3.2 Description des fonctionnalités du service Push

Le service Push nous permet de réaliser certaines fonctionnalités permettant la gestion des utilisateurs :

- Ajout d'un nouvel utilisateur.
- Suppression d'un utilisateur.
- Suppression de tous les utilisateurs.
- Envoyer un Push Notification à tous les utilisateurs.
- Envoyer un Push Notification à un utilisateur spécifié par son numéro de téléphone.

Ce service Push se caractérise aussi par une flexibilité lors du redémarrage de l'alarme. Les données des utilisateurs seront stockées dans un fichier binaire. Le fichier s'actualise à chaque ajout ou suppression d'un utilisateur ainsi lors du changement du TOKEN.

Lorsqu'on initialise l'alarme, nous cherchons la présence du fichier pour ajouter les utilisateurs déjà inscrits.

3.3.4 Communication Service Push et application client Android

Dans cette partie, nous décrivons les différents changements apportés au service Push et à l'application Android afin de les synchroniser. Nous expliquons également la communication entre l'application et le produit alarme.

Dans la figure 3.9, nous décrivons la communication d'authentification entre l'application Android et le service Push et l'envoi du TOKEN obtenu de Google Cloud Messaging à notre plateforme d'alarme.

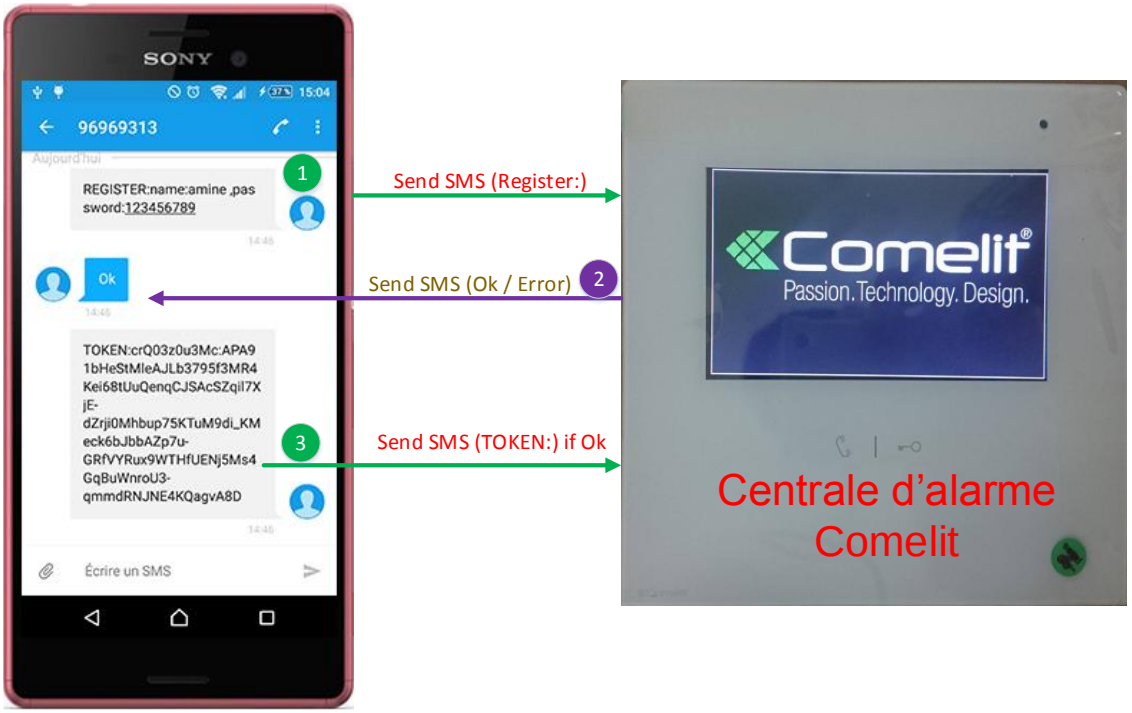


Figure 3.9 : Communication SMS application Android et Service Push

La figure 3.9 illustre des traces de transfert d'SMS réalisées entre l'application Android, et le service Push dans la plateforme Nuvoton.

Tout d'abord, afin de communiquer et envoyer le TOKEN à la plateforme Nuvoton, nous réalisons l'ajout d'un nouveau utilisateur dans le produit alarme. Il sera identifié par son numéro de téléphone unique. Dans ce cas l'application Android, réalise une identification de l'utilisateur via une interface d'authentification dans laquelle l'utilisateur entre son nom et le mot de passe de son produit alarme. Un SMS sera alors envoyé de l'application Android vers l'alarme et contient le nom de l'utilisateur ainsi que le mot de passe de la boite alarme. C'est la communication N°1 de la figure 3.9.

Lorsque la boite alarme détecte cet SMS via le Driver 3G sous forme d'événement :

- Le service 'Telephony' vérifie qu'il s'agit d'un SMS d'authentification avec la chaine de début de l'SMS « REGISTER : »
- Il vérifie que l'SMS reçu est en provenance du numéro déjà enregistré.
- Le produit alarme retourne un SMS « Ok » vers le client si le mot de passe est vérifié ou « Error » dans le cas contraire. Cette communication est la N°2 de la figure 3.9.

L'application reste en attente jusqu'à l'arrivé d'un SMS de l'alarme indiquant le succès d'authentification ou un échec, sinon elle sortira après un délai d'attente.

Si l'authentification est vérifiée, elle envoie le TOKEN reçu de GCM vers l'alarme pour l'enregistrer spécifiquement à cet utilisateur, c'est la communication N°3. Sinon, l'utilisateur a jusqu'à 3 essais en total pour réaliser son authentification.

Lorsque l'utilisateur se désinscrit du serveur GCM, et s'enregistre de nouveau, un SMS automatique sera envoyé à l'alarme contenant le nouveau TOKEN. Les informations propre à cet utilisateur seront mises à jours automatiquement dans l'alarme.

Lorsque le produit alarme veux notifier un utilisateur, il envoie une demande au serveur GCM contenant le TOKEN spécifique à l'utilisateur cible.

3.4 Conception de notre application

Le langage de modélisation unifié UML (Unified Modeling Language) est un langage de modélisation graphique à base de pictogrammes conçus pour fournir une méthode normalisée pour visualiser la conception d'un système.

Durant cette section nous allons décrire le système de Push Notification avec les diagrammes de modélisation UML (cas d'utilisation, séquence et classe).

3.4.1 Diagramme de cas d'utilisation

La figure 3.10 présente le diagramme de cas d'utilisation du mécanisme d'envoi du Push Notification. Sur ce diagramme, il existe trois acteurs principaux :

- Utilisateur de l'application Push Notification sur les Smartphones Android. Il permet de réliser l'authentification dans l'application Android qui se réalise que si l'utilisateur est déjà ajouté dans le produit alarme. Après authentification, Il reçoit le TOKEN du serveur Google Cloud Messaging et l'envoie dans un SMS à la boite d'alarme. A partir de cet instant, il peut recevoir une notification de l'alarme en cas de besoin.
- Responsable du produit alarme qui manipule la liste des utilisateurs dans l'alarme.
- Le systéme d'alarme qui permet le rafraichissement de du TOKEN lorsqu'il est reçu par l'utilisateur.

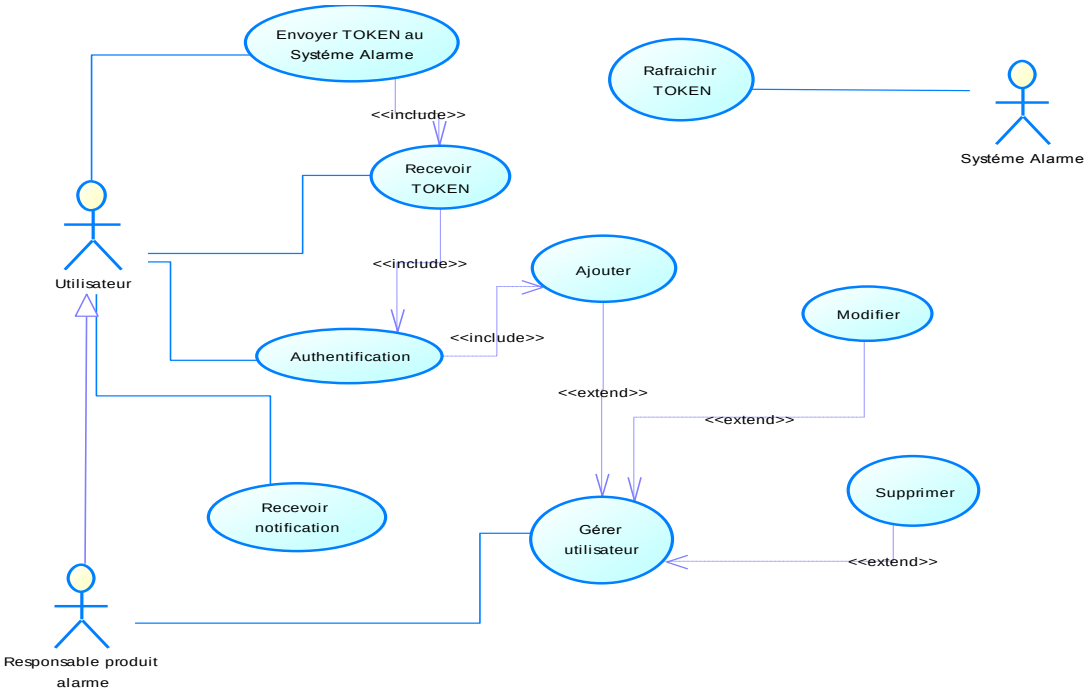


Figure 3.10 : Diagramme de cas d'utilisation

3.4.2 Diagrammes de séquence

Pour bénéficier des services de l'application, l'utilisateur doit remplir un formulaire d'authentification où il saisit son nom et le mot de passe de la boite d'alarme. Ces données sont envoyées via un SMS à la boite d'alarme qui sera chargée de vérifier l'existence de l'utilisateur. Par la suite, si les identifiants sont valides, la boite d'alarme envoie un SMS qui contient « Ok » à l'application sinon elle envoie « Error ».

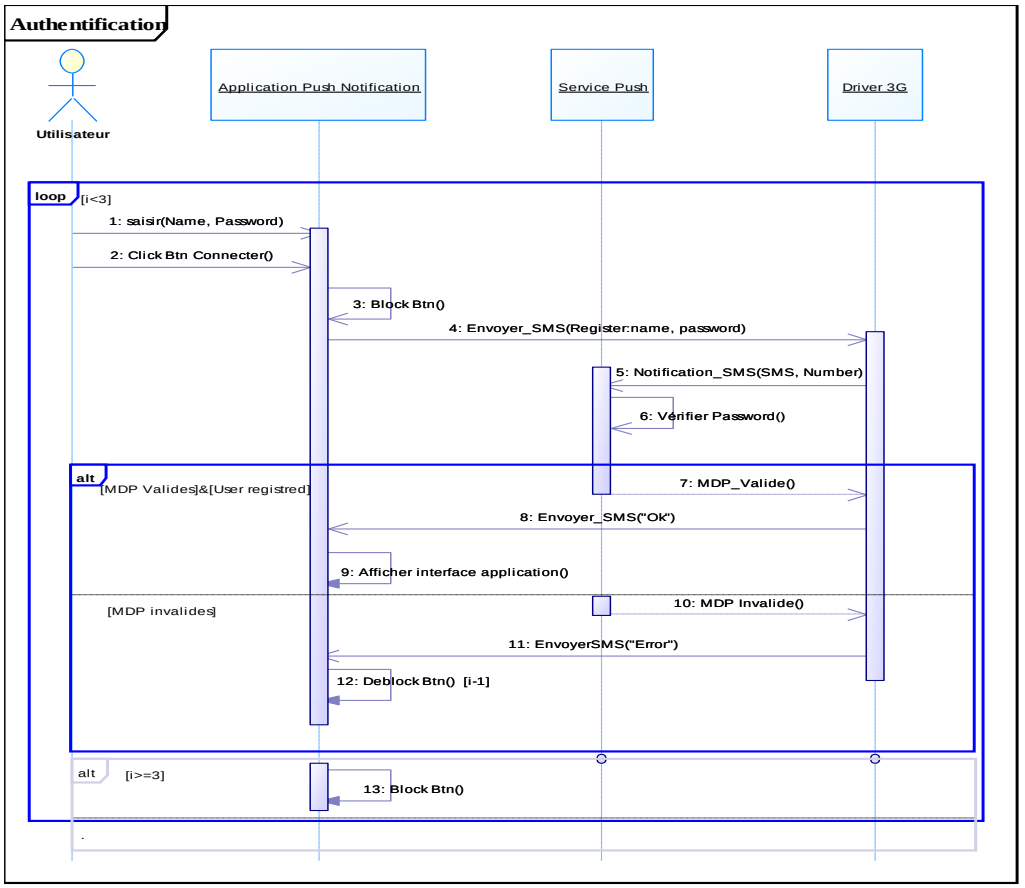


Figure 3.11 : Diagramme de séquence du cas d'utilisation « Authentification »

Après une authentification réussie, l'utilisateur se connecte au serveur Google Cloud Messaging avec le SenderID et l'ID d'application qui lui génère un TOKEN. L'utilisateur envoie un SMS à la boite alarme pour enregistrer son nouveau TOKEN. En cas de besoin, le service Push envoie un message à l'utilisateur sous forme de Push Notification à travers l'API GCM.

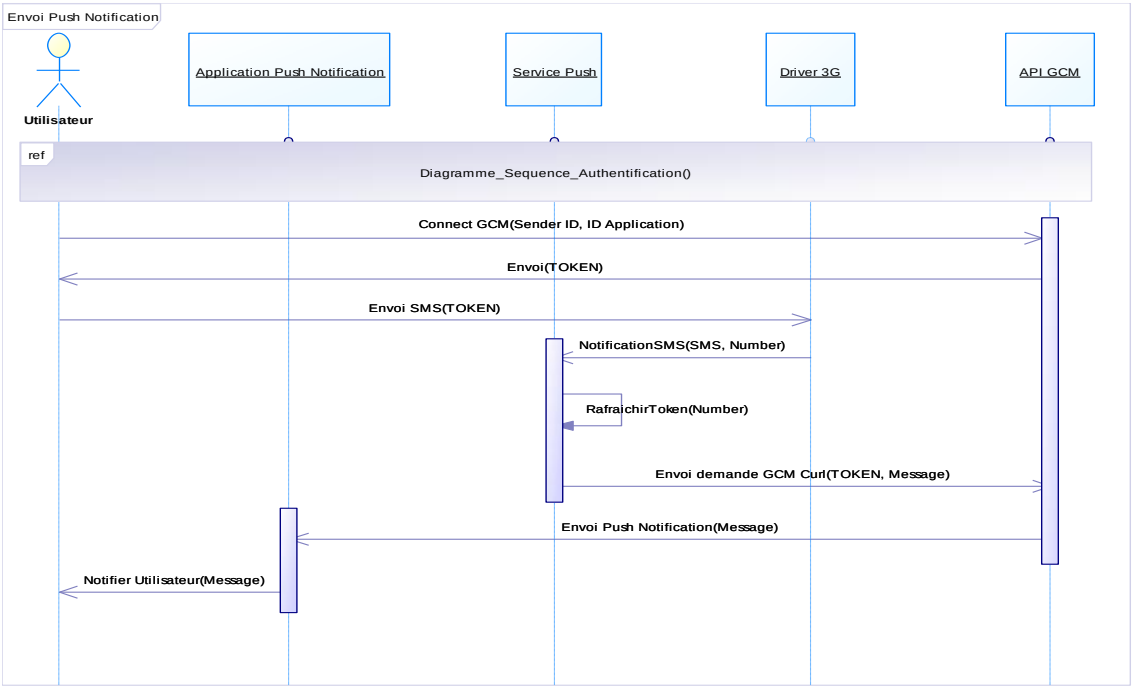


Figure 3.12 : Diagramme de séquence de « l'envoi Push Notification »

3.4.3 Diagramme de classe

Le diagramme de classe est l'un des diagrammes statiques d'UML. Il permet de décrire la structure d'un système informatique tout en montrant les différentes classes, leurs attributs, leurs méthodes ainsi que les relations entre eux.

Le diagramme de classe de la figure 3.13 décrit la communication entre les différentes fonctions de l'application Push Notification.

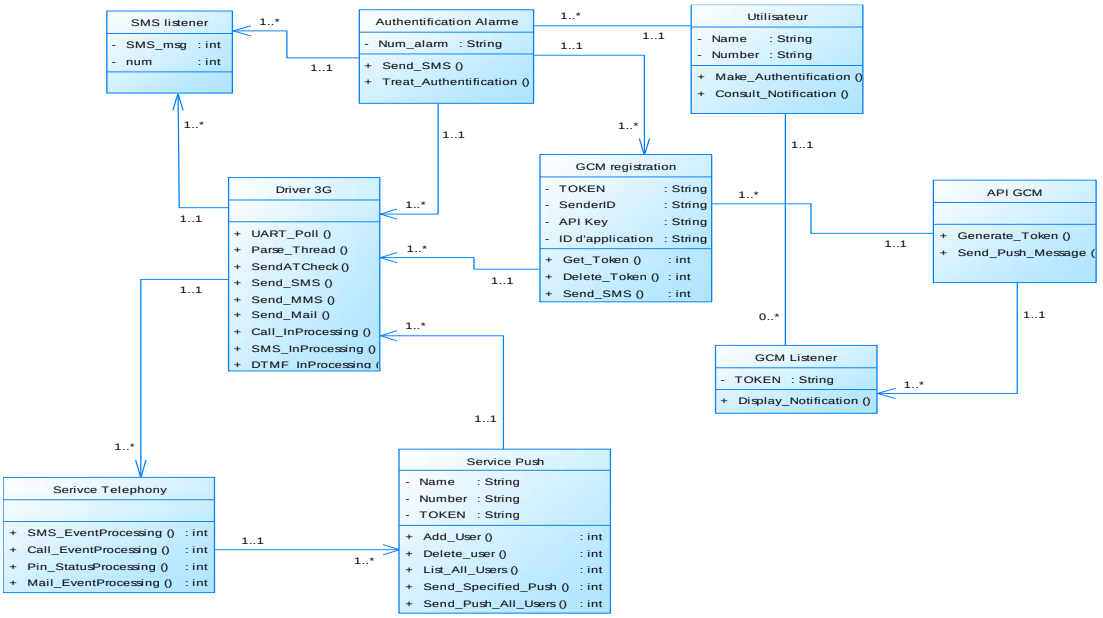


Figure 3.13 : Diagramme de classe

3.5 Test et validation

Dans cette section, nous validons et testons les différentes fonctionnalités de l'application Android Push Notification ainsi que les traces illustrant la communication entre les différents acteurs du produit alarme.

3.5.1 Test et validation de l'application Android

Pour accéder au service permettant la connexion au Google Cloud Messaging, il faut passer par l'interface d'authentification. Nous allons présenter dans cette section les différentes interfaces de notre application.

3.5.1.1 Authentification de l'application

Nous avons créé une interface d'authentification, pour accéder au service qui permet d'acquérir le TOKEN depuis le serveur GCM.

Lors du lancement de l'application, l'interface de la figure 3.14 apparait. Nous saisissons alors le nom de l'utilisateur et le mot de passe de l'alarme puis nous appuyons sur le bouton login. Ensuite, l'application bloque le bouton login et on remarque la présence du bar de progression (figure 3.15). En effet, le téléphone envoie un SMS à l'alarme sous le format suivant : « REGISTER:name:<name>,password:<password> ».

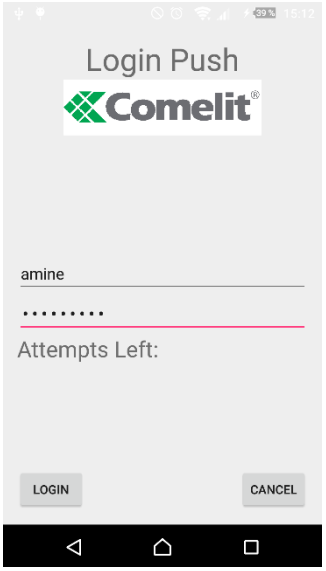


Figure 3.14 : Interface d'authentification

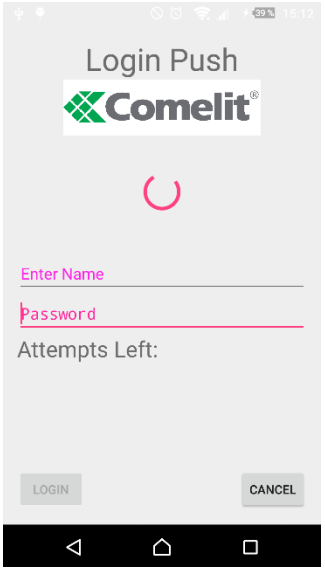


Figure 3.15 : Envoi du message d'authentification à l'alarme

L'alarme vérifie le mot de passe reçu, et elle le trouve erroné. Alors, elle retourne un SMS indiquant une mauvaise authentification à l'utilisateur contenant « Error ».

Lorsque l'utilisateur reçoit l'SMS indiquant la mal authentification, il peut se ré-identifier avec un nombre maximale d'authentification égale à 3 comme l'indique la figure 3.16.

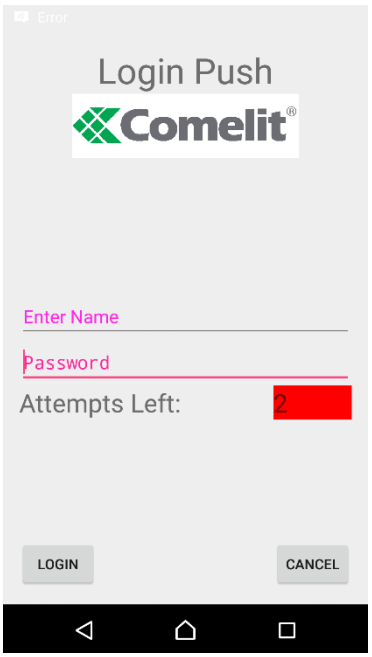


Figure 3.16 : Authentification Erronée

La figure 3.17 représente le cas d'une mauvaise authentification de l'utilisateur. En effet un message « Error » est envoyé depuis le produit alarme.

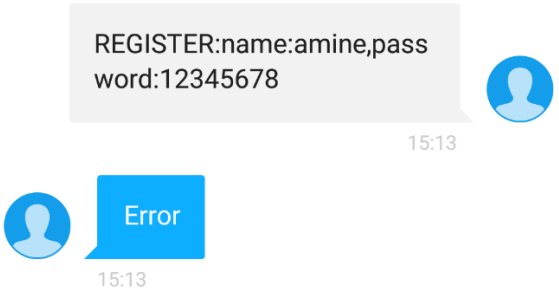


Figure 3.17 : Echange d'SMS avec l'alarme

3.5.1.2 Connexion avec GCM et Réception TOKEN

Lorsque l'authentification de l'utilisateur est bien réalisée, une nouvelle interface s'affiche (figure 3.18) afin qu'il se connecte à Google Cloud Messaging et reçoit le TOKEN. L'application Android envoie alors un message SMS à l'alarme contenant le TOKEN déjà acquis du GCM (figure 3.19).

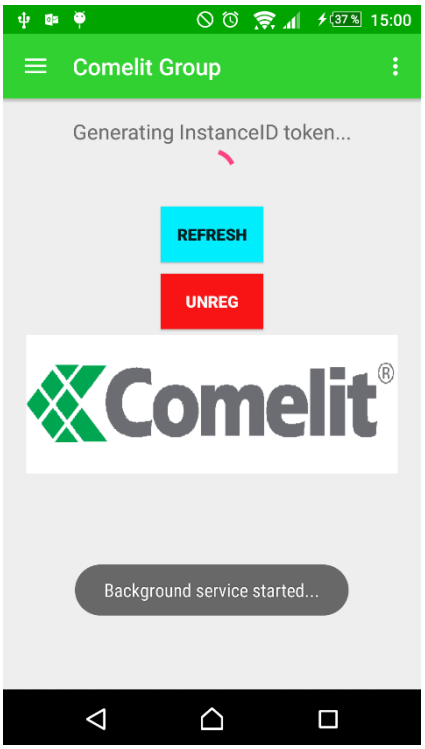


Figure 3.18 : Génération du nouveau TOKEN

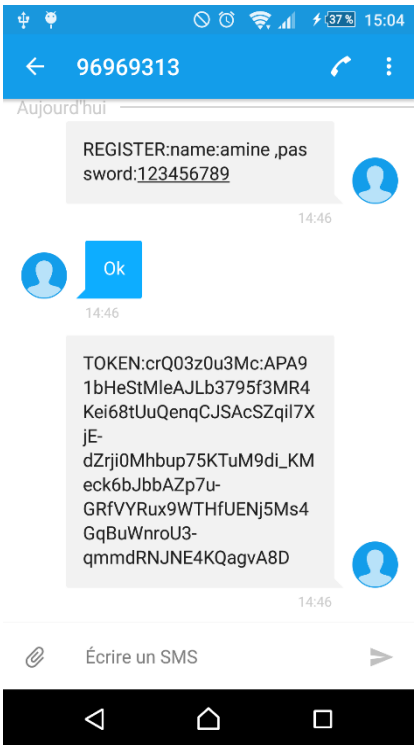


Figure 3.19 : Echange d'SMS avec l'alarme

Lorsque l'utilisateur veut réaliser une désinscription du service GCM, il clique sur le bouton « UNREG » de la figure 3.20. Dans ce cas, l'application réalise le traitement pour effacer le TOKEN. Lorsque le TOKEN actuel sera effacé du GCM, nous affichons un message indiquant la désinscription réussite comme il est indiqué sur la figure 3.21.

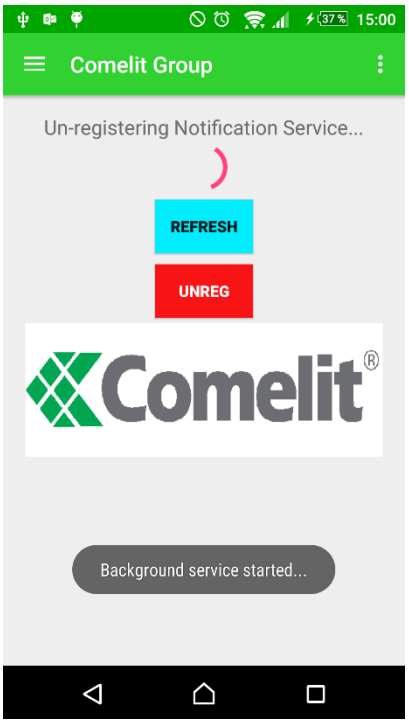


Figure 3.20 : Lancement de désinscription de GCM

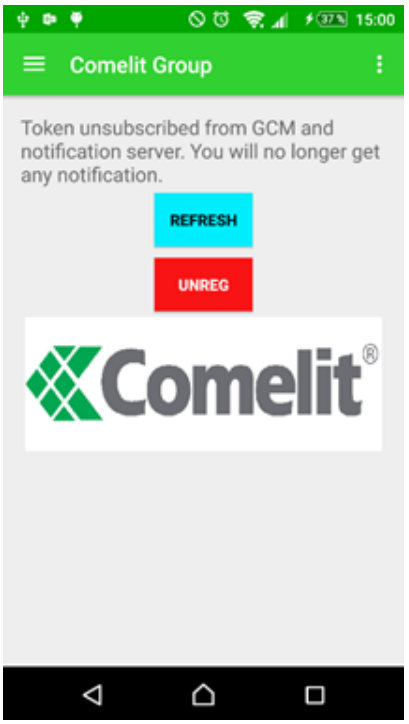


Figure 3.21 : Désinscription réussie de GCM

3.5.1.3 Rafraichissement du TOKEN

Lorsque l'utilisateur réalise une désinscription du serveur Google Cloud Messaging avec le bouton « UNREG », et réalise un enregistrement de nouveau avec le bouton « REFRESH ». Il obtiendra un nouveau TOKEN comme il est indiqué dans la figure 3.22. A cet instant, l'application envoie un SMS contenant le nouveau TOKEN à l'alarme pour rafraichir celui déjà enregistré dans l'alarme.

Dans la figure 3.23, nous montrons le message SMS envoyé du nouveau TOKEN vers l'alarme.

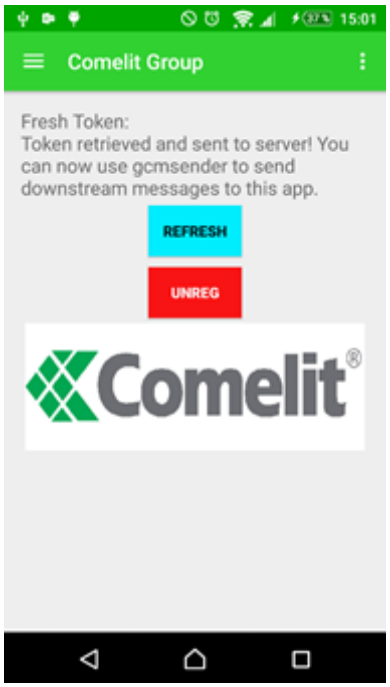


Figure 3.22 : Rafraichissement du TOKEN

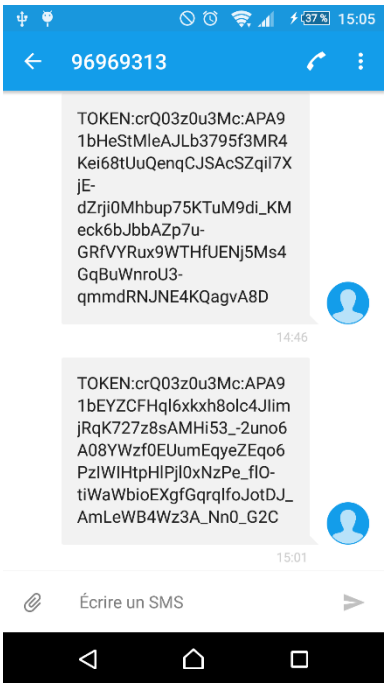


Figure 3.23 : Envoi du TOKEN à l'alarme

3.5.2 Test et validation du service Push dans l'alarme

Dans la figure 3.24, nous présentons la liste du menu de choix qui représente les fonctionnalités possibles que nous pouvons réaliser afin de manipuler le service de Push.

Figure 3.24 : Test des fonctionnalités du service Push

La liste du menu de choix contient les fonctionnalités suivantes :

1. Ajouter un utilisateur.
2. Lister les utilisateurs.
3. Effacer utilisateur.
4. Envoyer un Push à un utilisateur spécifique.

5. Envoyer un Push à tous les utilisateurs.
6. Recevoir une nouvelle adresse ip.
7. Mettre à jour le fichier d'enregistrement des utilisateurs.
8. Mettre fin au service Push.
9. Effacer tous les utilisateurs.
10. Effacer tous les SMS enregistrés dans la carte SIM du module 3G.
11. Sortir du menu de choix

Nous avons enregistré deux utilisateurs avec leur TOKEN dans notre liste des utilisateurs comme il est indiqué dans la figure 3.20, le premier s'appelle amine et le deuxième s'appelle amen. L'enregistrement du TOKEN se fait automatiquement lors de l'enregistrement.

Nous allons envoyer un message Push à toute la liste des utilisateurs comme il est indiqué dans la figure 3.25.

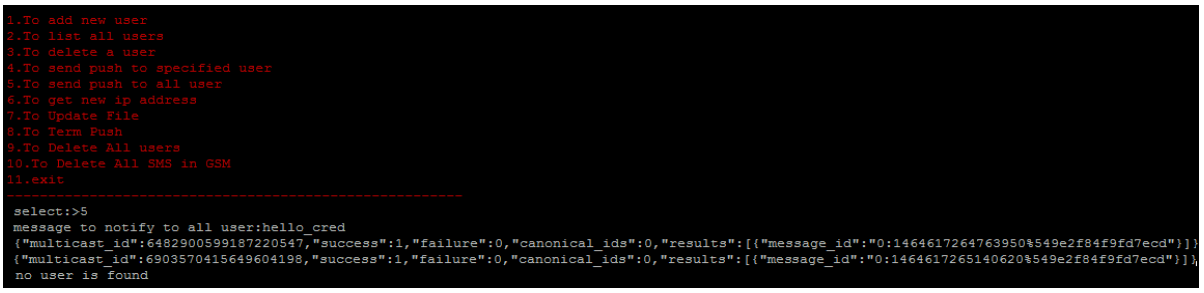


Figure 3.25 : Envoi message Push aux utilisateurs

Le message envoyé est 'hello_cred' dans un Push aux utilisateurs enregistrés avec leurs TOKEN.

Nous recevons instantanément, le message Push dans les deux appareils téléphoniques spécifiques aux deux utilisateurs enregistrés.

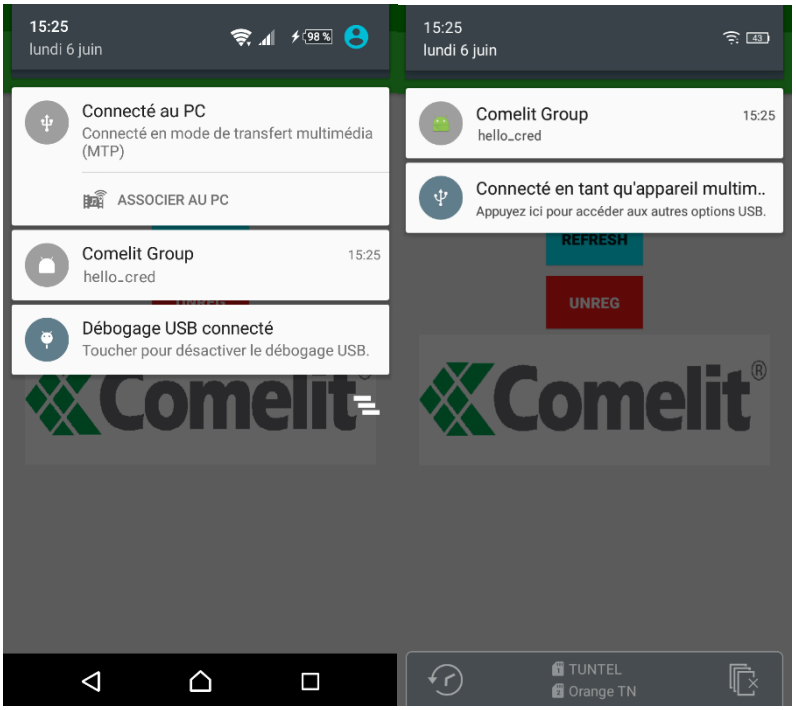


Figure 3.26 : Réception de notification sur le téléphone de l'utilisateur amine

Figure 3.27 : Réception de notification sur le téléphone de l'utilisateur amen

3.6 Conclusion

Nous avons présenté dans ce chapitre les étapes de réalisation de l'application Push Notification. Nous avons détaillé le développement des différentes parties de l'application. Ensuite, nous avons réalisé la conception afin de détailler l'échange entre les différents modules de notre système. Enfin, nous avons tester et valider notre application développée à travers le présentation de résultats d'exécution.

Conclusion générale

Le présent rapport est rédigé dans le cadre de la préparation d'un projet de fin d'études à l'Institut Supérieur d'Informatique, en vue de l'obtention de diplôme d'Ingénieur en Génie Informatique des Systèmes Industrielles.

Notre projet a consisté à la mise en œuvre et l'implémentation du Driver 3G et de l'application Push Notification, sur une plateforme à base de microcontrôleur ARM appelé Nuvoton pour la société Comelit Tunisie. Pour se faire, nous avons commencé par étudier le Module 3G quectel M95 et sa communication avec la plateforme Nuvoton afin de comprendre son fonctionnement et identifier les fonctionnalités à réaliser dans le produit. Ensuite, nous avons procédé à l'implémentation du Driver 3G sur la plateforme Nuvoton en utilisant l'environnement BUILDROOT pour le développement du code. En utilisant les fonctionnalités envoi et réception de SMS du driver 3G, nous avons développé le service contrôle SMS pour assurer le pilotage du système d'alarme. Par la suite, nous avons réalisé l'application de Push Notification qui se base sur le serveur Google Cloud Messaging permettant d'envoyer des messages Push au smartphones android en temps réel. Enfin, nous avons validé nos codes à travers des tests d'intégrité.

Comme perspectives de notre travail, nous envisageons de réaliser un API comparable à Google Cloud Messaging qui sera spécifique à la clientèle des produits Comelit.

Ce projet nous a été bénéfique sur le plan professionnel et personnel. Sur le plan professionnel, nous avons eu l'occasion d'aiguiser nos connaissances en langage C, à l'environnement Buildroot et à la plateforme de développement Nuvoton et de travailler sur un projet de grande envergure. Sur le plan personnel, nous avons appris à bien travailler en équipe, à se faire des concessions et de côtoyer l'environnement professionnel.

Bibliographie

[1]http://www.lavoixdunord.fr/Locales/Cambrai/actualite/Cambrai/2011/11/12/article_les-alarmes-font-elles-encore-peur-aux-c.shtml [Accès le 22-Février-2016]

[2] http://pro.comelitgroup.com/int#[Accès le 5-Mars-2016]

[3]http://docplayer.fr/5825839-Initiation-a-la-technologie-secondaire-projet-et-guide-pedagogique-module-3-production-electromecanique-le-systeme-d-alarme-janvier-2002.html[Accès le 18-Mars-2016]

[4]http://www.businesscoot.com/le-marche-des-alarmes-et-du-materiel-de-videosurveillance-51/[Accès le 23-Février-2016]

[5] http://www.anti-cambriolage.fr/alarme/normes/[Accès le 12-Avril-2016]

[6]http://www.saintjocaudan.fr/les_ressources/technologie/quatrieme/securit_

maison/fonctionnement_systeme_alarme.htm[Accès le 5-Mai-2016]

[7] http://www.microchip.ua/nuvoton/OTHER/N3292x-DevelopmentBoard.pdf[Accès le 15-Mars-2016]

[8] http://wenku.baidu.com/view/ddd03147bd64783e09122be5.html###[Accès le 25-Mai-2016]

[9] http://free-electrons.com/doc/training/embedded-linux/embedded-linux-slides.pdf [Accès le 15-Mars-2016]

[10] E*nvironnement buildroot*. http://linux.developpez.com/tutoriels/embarque-buildroot/ [Accès le 23-Février-2016]

[11]http://www.quectel.com/product/prodetail.aspx?id=7[Accès le 5-Avril-2016]

[12] https://www.youtube.com/watch?v=rhuibuzF888 [Accès le 25-Fevrier-2016]

[13] http://repo.hackerzvoice.net/depot_madchat/reseau/Commande%20AT.htm [Accès le 5-Avril-2016]

[14] http://www.musitel.com/EN/alarmes_commandes_sms_fr.html [Accès le 5-Avril-2016]

[15] http://charusat.net/NCSCA2016/NCSCA-2016_Conference-proceeding/22.pdf [Accès le 15-Mars-2016]