Université de Perpignan

Rapport de stage

Effectuée du 01/01/2016 au 30/06/2016

à

Oroora Clean Technology Sarl.

Tunis, Tunisie

Développement d'un Système de Monitoring des Installations Solaires Photovoltaïques

Par:

Montassar Jomâa

Encadré par:

Lotfi Chaarabi, Maitre de Conférences, Ecole Nationale d'Ingénieurs de Tunis

Table of Contents

1. Introduction

1.1 La croissance du marché du solaire photovoltaïque

Le solaire photovoltaïque connait depuis une dizaine d'années une croissance fulgurante à deux chiffres. La croissance est alimentée par les baisses des coûts des systèmes ainsi que les différents mécanismes d'incitations (fiscales et subventions). Figure 1 présente l'historique de la capacité annuelle installée au niveau mondiale qui a atteint 40 GW pour l'année 2014. Le récent démantèlement des mécanismes d'incitations et de subventions dans certains pays clés (Allemagne, Espagne) n'ont pas affecté le rythme de croissance au niveau mondial. Il est prévu que le rythme sera maintenu pour les prochaines années comme le prédit l'association européenne du solaire photovoltaïque SolarPower Europe. Les prévisions des capacités annuelles et cumulées pour la période 2015-2019 sont données en figure 2 et 3. Il est prévu que la capacité annuelle va doubler dans les 4 prochaines années.

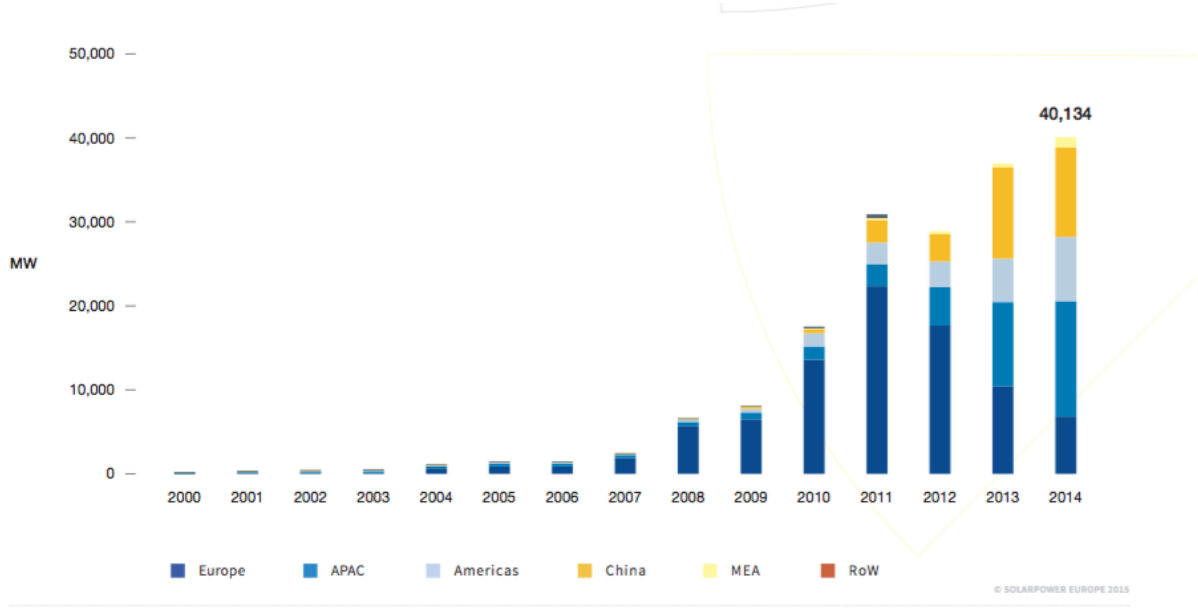

FIGURE 1: HISTORIQUE DE LA CAPACITE ANNUELLE INSTALLEE AU NIVEAU MONDIALE. SOURCE : SOLARPOWER EUROPE

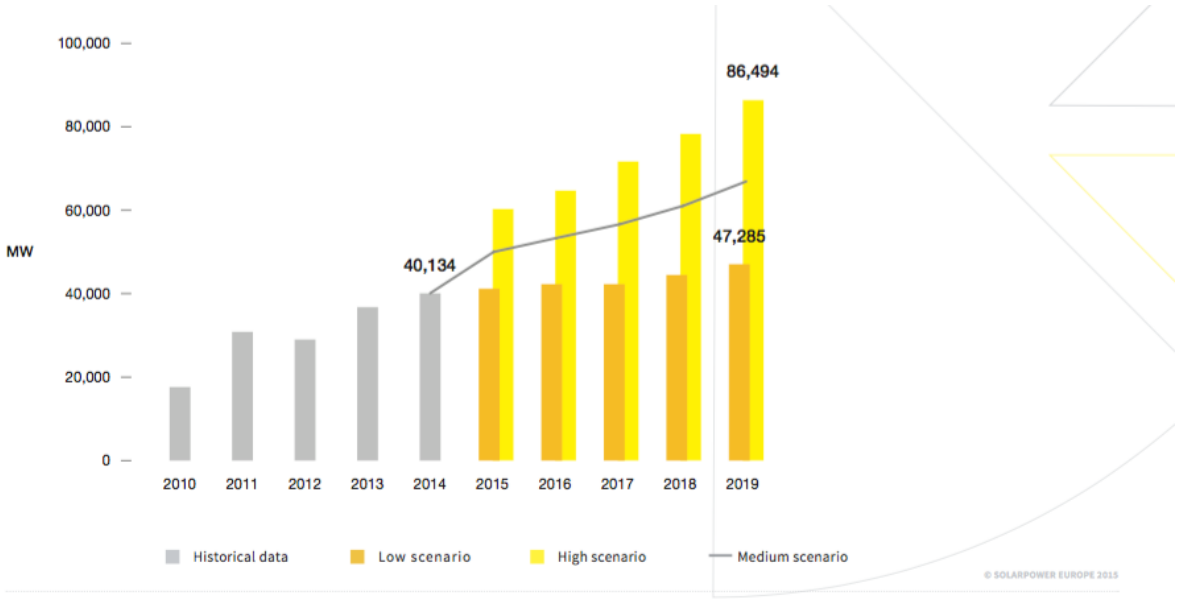

FIGURE 2: PREVISION DE LA CROISSANCE DES CAPACITES ANNUELLES INSTALLEES POUR LA PERIODE 2015-2019. SOURCE : SOLARPOWER EUROPE.

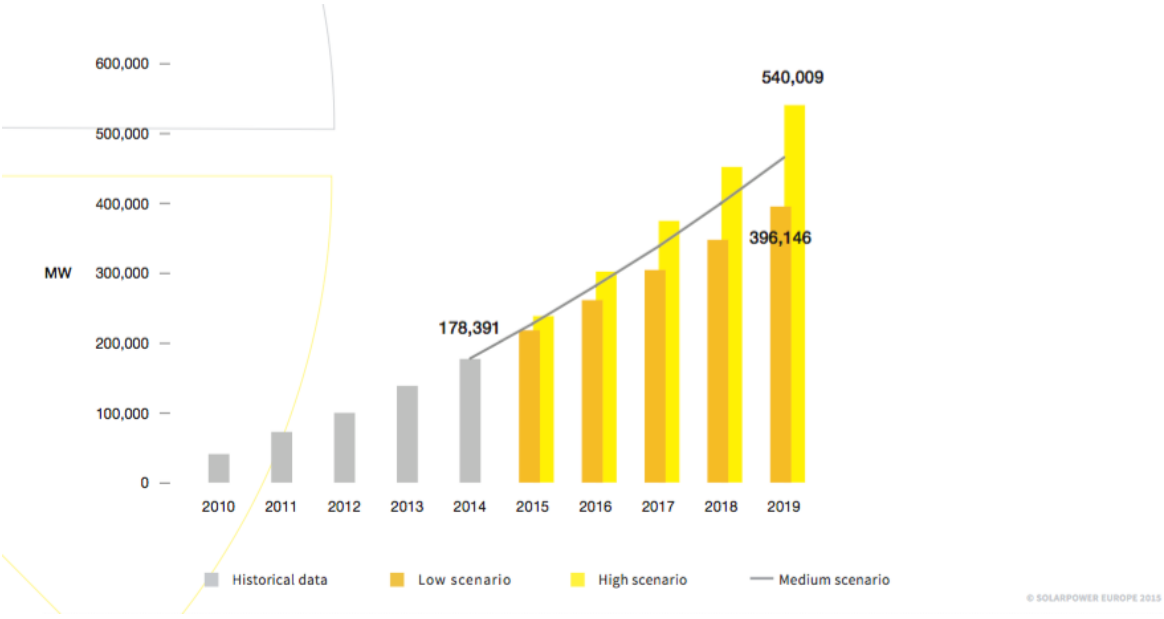

FIGURE 3 : PREVISION DE LA CAPACITE SOLAIRE INSTALLEE EN CUMULEE POUR LA PERIODE 2015-2019. SOURCE : SOLARPOWER EUROPE.

Les secteurs résidentiel, commercial et industriel représentent à peu près 50 % du part de marché. Ces systèmes sont généralement de petite et moyenne taille typiquement inférieure à 1 MWc. La majorité de ces systèmes sont de type connecté au réseau électrique sous les deux formes les plus connues ; net-metring et Feed-in-Tariff. Dans certains pays comme : le Danemark, les Pays-Bas ; la république Tchèque et la Belgique, le secteur résidentiel domine le marché avec un part de marche qui dépasse les 50 % (Figure 4).

Net-metering : (comptage net ou facturation nette) est un service accordé par un fournisseur d'électricité à un consommateur, lui permettant de compenser partiellement sa

consommation d'électricité facturée par la production d'une installation solaire ou autres qu'il exploite sur son site de consommation. De façon plus générale, il consiste à mesurer et facturer la consommation nette en permettant au compteur de ces consommateurs de tourner à l'envers lorsque leur production dépasse leur consommation, ou encore en utilisant deux compteurs, l'un enregistrant la production, l'autre la consommation, et en effectuant la soustraction sur la facture.

Feed-in-Tariff : est un mécanisme de politique visant à accélérer les investissements dans les énergies renouvelables. Elle y parvient en proposant des contrats d'achat d'électricité à prix fixe pour les producteurs d'énergie renouvelable. Ce mécanisme se différencie par rapport au net-metering par la possibilité d'une compensation financière pour l'investisseur ce qui n'est généralement pas possible dans le cas du net-metering.

Résidentiel : est tout système installé sur le toit ou au sol ayant une capacité crête inferieure à 10 kWc.

Commerciale : est tout système installé sur le toit ou au sol ayant une capacité crête inferieure à 250 kWc.

Industriel : est tout système installé sur le toit ou au sol ayant une capacité crête inférieure à 1 MWc.

Utility scale : est tout système installé au sol ayant une capacité crête supérieure à 1 MWc.

Grid-parity : La parité réseau se produit lorsqu'une nouvelle source d'énergie peut produire de l'énergie à un coût moyen actualisé de l'électricité (Levelezid Cost of Energy - LCOE) qui est inférieur ou égal au prix d'achat d'électricité du réseau électrique. Le terme est le plus souvent utilisé lors de l'examen des sources d'énergie renouvelables, notamment l'énergie solaire et l'énergie éolienne. Atteindre la parité réseau est considéré comme le point où une source d'énergie devient un concurrent pour le développement généralisé sans subventions ou le soutien du gouvernement.

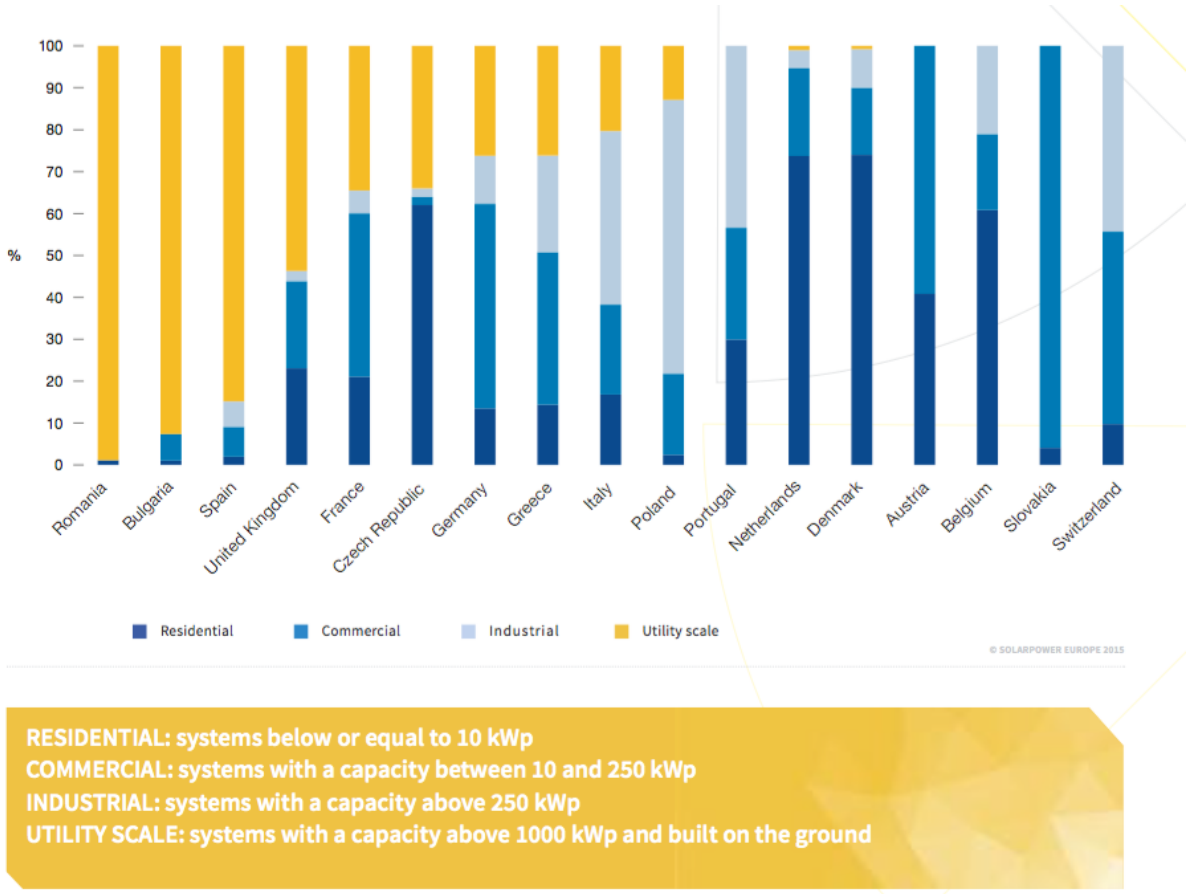

FIGURE 4: DISTRIBUTION DES DIFFERENTS MARCHES DE SOLAIRES (RESIDENTIEL, COMMENRCIAL, INDUSTRIEL AND UTILITE) DANS LES PRINCIPAUX MARCHES EUROPEENS. SOURCE: SOLARPOWER EUROPE.

SolarPower Europe prévoit que ces deux marchés (roof top et utility scale) vont continuer à évoluer dans la période 2015-2019 avec un part de marché légèrement en faveur de cette dernière, Figure 5. La baisse des coûts des systèmes va rendre le solaire économiquement viables dans de nouveaux marches émergents comme la région MENA (Midle East and North Africa) et l'Amérique Latine.

Autre facteur important est que le solaire en particulier et les énergies renouvelables en générale sont entrain de gagner des parts significatifs du marché dans les nouvelles installations toutes technologies confondues. Il est important également de noter que les technologies traditionnelles ; nucléaire, charbon et fuel, sont en train de perdre des parts comme le montre la Figure 6. Cette tendance va s'accélérer les prochaines années suite a la signature historique par les pays industriels du nouvel accord au COP21 de Paris.

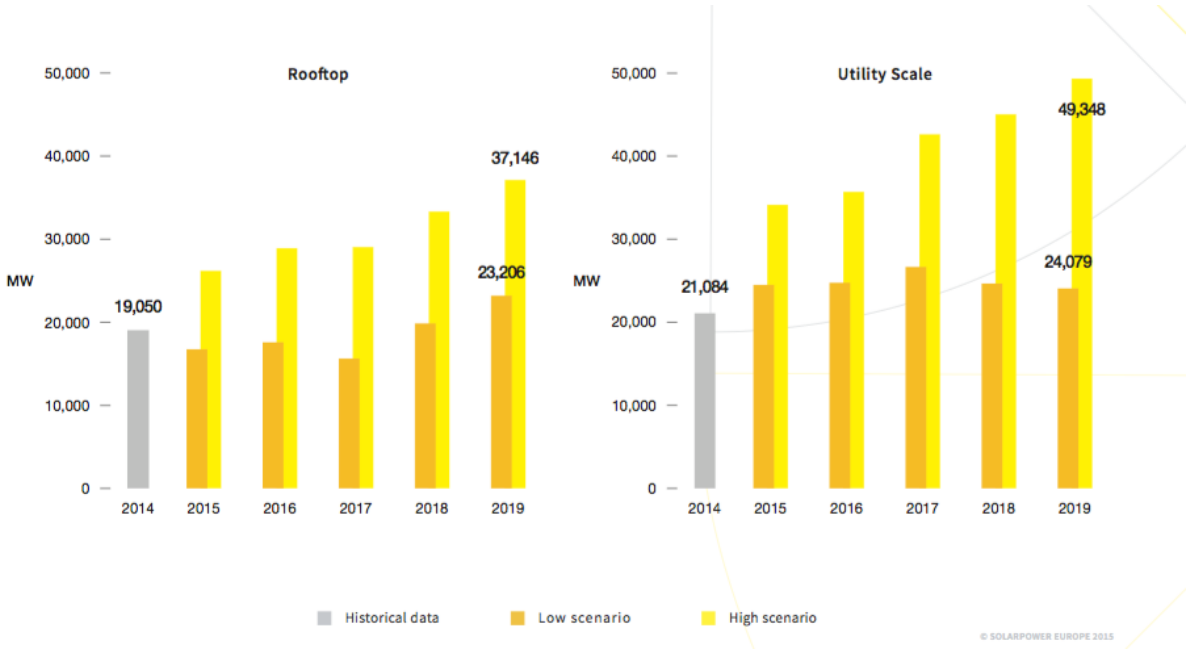

FIGURE 5: PREVISION DE L'EVOLUTION DES SECTEURS SOLAIRES SOLAIRES (MONTE SUR LE TOIT ET ECHELLE D'UTILITE) POUR LA 2015-2019. SOURCE: SOLARPOWER EUROPE.

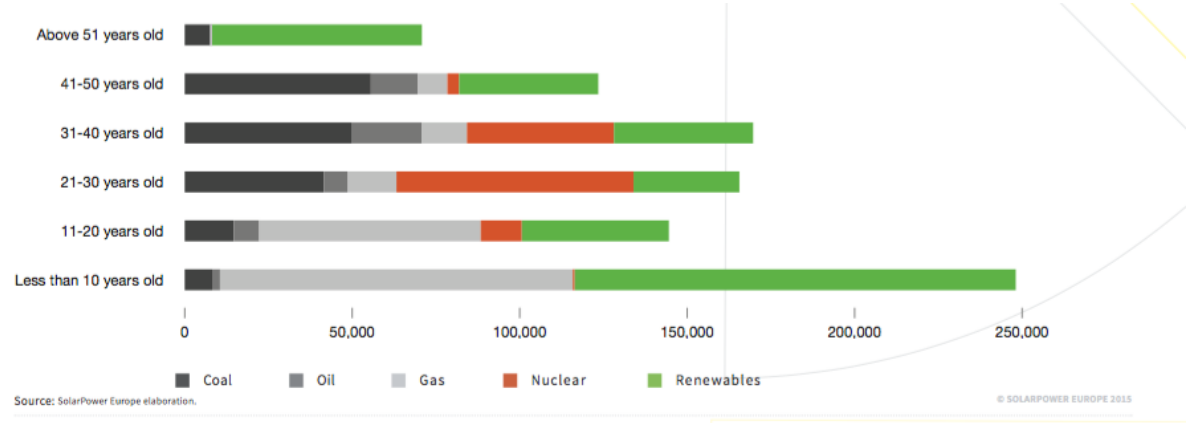

FIGURE 6: L'AGE DES DIFFERENTS TECHNOLOGIES UTILISEES DANS LA PRODUCTION D'ELECTRICITE EN EUROPE. SOURCE: SOLARPOWER EUROPE.

1.2 L'importance du monitoring

La nature intermittente de l'énergie solaire et sa sensibilité aux facteurs environnementaux (irradiation, température) ont rendu le recours à des systèmes de surveillance essentiel pour un bon fonctionnement du système et pour optimiser le retour sur investissement. Ils existent différentes solutions techniques à ce problème qu'on peut catégoriser en fonction de l'échelle de finesse dans la surveillance :

- Surveillance au niveau du panneau ; les paramètres sont mesurés à l'échelle de chaque panneau

- Surveillance au niveau du string ; les paramètres sont mesurés à l'échelle d'un groupement de panneaux
- Surveillance au niveau du système ; les paramètres sont mesurés à l'échelle d'un système

Optez pour l'une des solutions citées en haut dépendra de la complexité et la taille du système. D'une façon générale pour les systèmes de taille importante, le monitoring se fait au niveau du panneau ou au niveau de string. L'objective est de pourvoir identifier rapidement le panneau ou l'ensemble de panneaux ayant un rendement faible et planifier les opérations de maintenance nécessaires. En l'occurrence, quand il s'agit d'un système de petite taille, de type résidentiel, le monitoring se fait au niveau du système. Ce choix est dû à des raisons économiques. Equiper chaque panneau par les capteurs nécessaires aura un impact sur le coût total du système et ainsi le choix se limite à un nombre limiter de capteurs.

On s'intéresse dans ce travail aux systèmes de monitoring utilisés pour les installations résidentielles de petite taille. En fait, actuellement la majorité des systèmes de types résidentiels ne sont pas munis d'un système de monitoring. Un tel système permettra de suivre les performances du système et éventuellement décider des opérations de maintenance. En outre, en cas où le système offre la surveillance de la consommation électrique domestique, des économies d'énergies de 10 %-15 % ont été enregistrées. L'utilisateur par l'observation des métriques de sa propre consommation, il devient conscient de son propre mode de consommation et ainsi il améliorera son comportement (changement d'appareils énergivore, utilisation de lampes économiques, …).

1.3 Présentation de l'entreprise

Oroora Solar offre des services en ingénierie spécialisée dans les installations photovoltaïques et opère principalement en Tunisie. Les installations sont typiquement des systèmes connectés au réseau sur la base du net-metering. Oroora Solar est entrain de développer un système de monitoring des installations solaires via internet. A terme, le système mesurera la production électrique de l'installation PV ainsi que la consommation électrique de la maison en question et envoie les données vers un serveur dans le cloud. Une application cloud analysera la performance de l'installation et envoie des alertes en cas de besoin vers le propriétaire de l'installation mais également vers l'équipe de maintenance pour programmer une intervention. L'application génèrera des rapports périodiques (hebdomadaire, mensuelle, etc.) sur les performances de l'investissement.

2 Conception du système

Pour qu'un système de surveillance trouve du succès, il faut qu'il soit :

- Simple à utiliser
- A faible coût
- S'adapte à différente configurations de systèmes

- Robuste et fonctionnant sous des conditions extrêmes (faibles utilisations d'énergies et fonctionne sous faibles connections d'internet ; gère la coupure d'internet)
- Les données doivent être facilement accessibles via différentes plateformes Mobile et Web.

Pour limiter le coût de la solution, il est important de limiter le nombre des composantes nécessaires à sa fabrication. Ainsi, il a été décidé de limiter les mesures à ces paramètres :

- L'énergie consommée
- L'énergie produite
- La température ambiante

La solution retenue pour ce travail se compose de trois parties distinctes :

- Un modem (les capteurs + une carte d'acquisition + un contrôleur)
- Une base de données
- Des applications mobiles et web

Figure 2 schématise la solution dans ces trois parties. Les détails de la solution seront traités dans le reste de ce rapport.

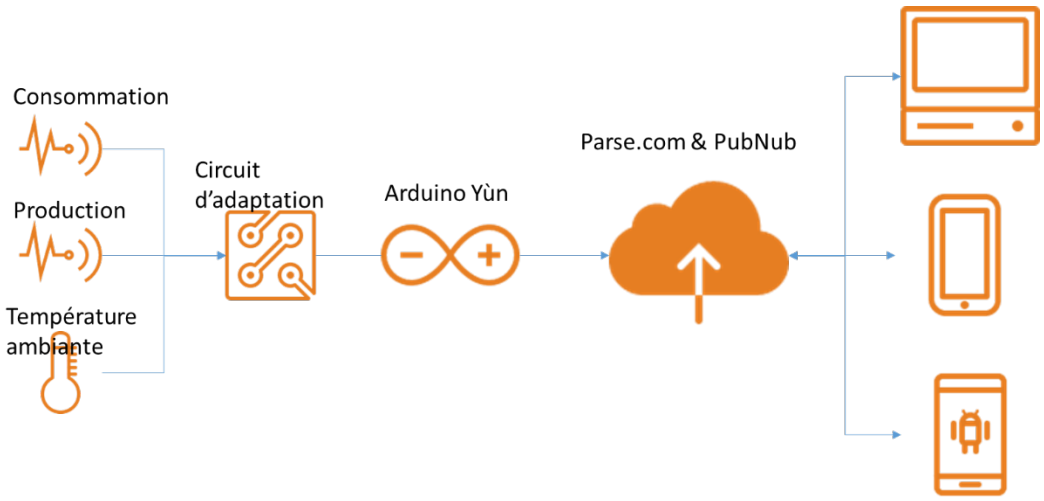

FIGURE 1 : SCHEMATIQUE DE LA SOLUTION DE MONITORING.

Cette solution suppose que la production et la consommation peuvent être surveillées séparément. Ainsi, la quantité exportée ou importée du secteur est simplement la différence entre la production et la consommation. La Figure xx présente le mode d'utilisation du modem avec un système photovoltaïque monophasé.

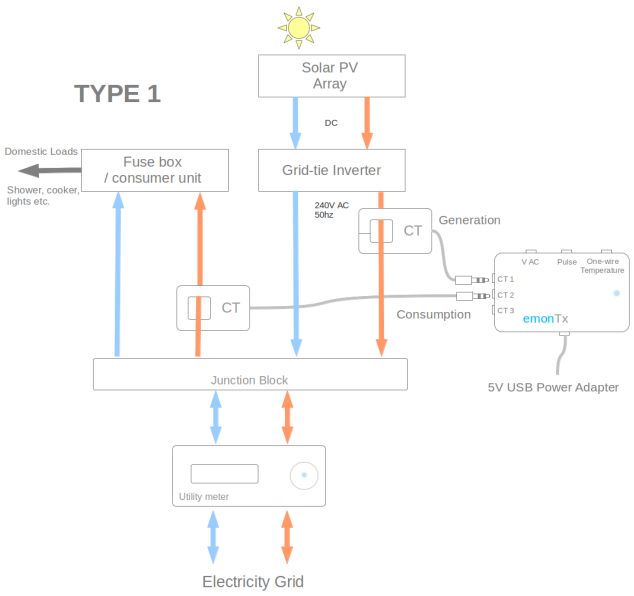

FIGURE 2: MODE D'UTILISATION DU MODEM DE SURVEILLANCE

On s'intéressera dans les prochaines parties a détaillé les choix techniques des trois parties de la solution ; modem, backend et application client.

3 Design du modem et des capteurs

Dans cette partie, les choix techniques adoptés dans la conception du modem sont présentés.

3.1 Capteur de courant

Le choix a été porté sur un transformateur de courant (TC). Ce choix est motivé par son faible coût, son accessibilité et la sécurité d'utilisation. Il existe de nombreuses TC sur le marché, comme celui représenté sur la figure 7, qui peut être utilisé pour effectuer la lecture du courant alternatif. Certains capteurs possèdent une résistance interne qui transforme le courant à une tension alternatif équivalente. On distingue deux types de capteurs TC suivant la nature de la sortie courant ou tension:

- SCT-013-000 [1] supporte un courant maximale de 100 A et fourni une sortie de 50 mA @ 100A (type de sortie: courant).
- SCT-013-030 [1] supporte un courant maximale de 30 A et fourni une sortie de 1 V @ 30A (type de sortie: tension).

On a choisit le capteur SCT-013-000 qui offre une sortie en courant. Ainsi on a ajouter une résistance de charge sur la carte d'acquisition pour transformer le courant en une tension de valeur maximale 5v qui correspond à la tension de référence du convertisseur analogique numérique de la carte Arduino. La valeur de la résistance de charge est calculée par la formule suivante :

$$R = \frac{v_{max}}{i_{max}}$$

Avec v_{max} la tension maximale convertie par le convertisseur analogique numérique de la carte Arduino qui vaut 5V. i_{max} correspond au courant maximale à la sortie du capteur. Sachant que les disjoncteurs domestiques utilisés en Tunisie sont de 16A ou de 32A ce qui correspond à un courant maximale en sortie du capteur de 8 mA ou de 16 mA. Ainsi la valeur de la résistance est de 620 Ω (valeur normalisée) pour les disjoncteurs de 16A et de 300 Ω (valeur normalisée) pour les disjoncteurs de 32A.

FIGURE 7: CAPTEUR DE COURANT TRANSFORMATEUR DE COURANT (TC)

Il existe d'autres modèles de capteurs de courant plus sophistiqués et plus précis comme le xxxx mais ils n'ont pas été testé dans ce travail. Il est recommandé de les considérer dans une prochaine version.

3.2 Capteur de tension

La tension est captée en utilisant un transformateur de tension qui fournie une image réduite et isolée de la tension secteur. Le transformateur (figure xxx) possède deux secondaires 2X6V. L'un des deux secondaires est utilisé comme capteur de tension l'autre pour alimenter la carte.

FIGURE 8: TRANSFORMATEUR DE TENSION

3.3 Capteur de température

L'efficacité des panneaux photovoltaïques se dégradent avec la montée de la température. Ainsi pour prendre en compte l'effet de ce paramètre, il a été décidé d'ajouter un capteur de

température à la solution. Comme le montre la Figure xx, l'efficacité des panneaux baissent de 7 % a 40 dégrée C. A terme, le système de surveillance doit prendre en compte cette dégradation en analysant la performance du système.

Il existe différents modèles de capteurs de températures (numériques et analogiques). Le choix a été porte sur un capteur numérique de type DS18B20. Ce choix est motivé par son faible coût ainsi que le niveau de précision qu'il peut offrir et son interfaçage simple avec la carte Arduino Yun. En fait, le fait qu'il n'est pas intégré dans la carte d'acquisitions directement il évitera toute altération des mesures du à la surchauffe de la carte d'acquisition.

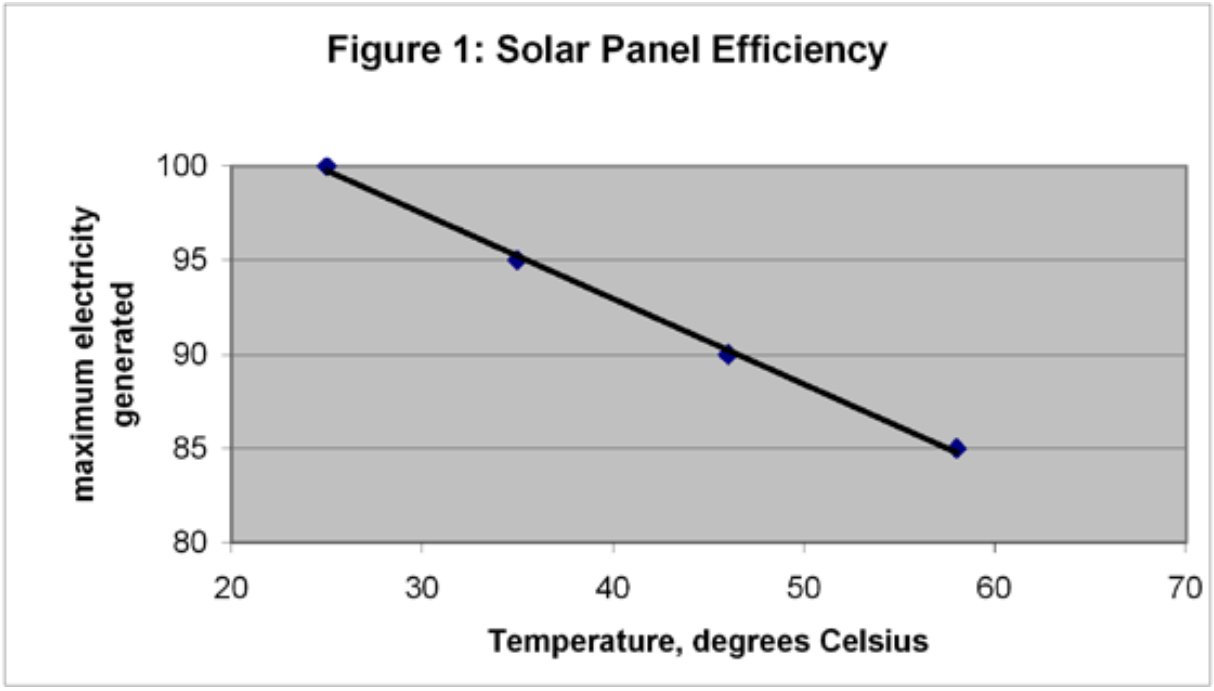

FIGURE 9: L'EFFICACITE DES PANNEAU SOLAIRES EN FONCTION DE LA TEMPERATURE.

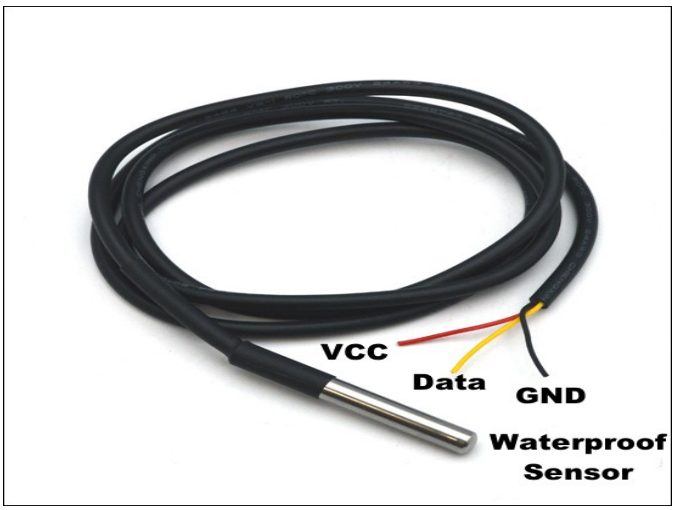

FIGURE 10 : CAPTEUR DE TEMPERATURE DS18B20

3.4 Choix du contrôleur

Il existe une large panoplie de contrôleurs conçus spécialement pour l'internet des objets. On note en particulier; xxxxxx. Pour ce travail, le choix s'est porte sur le Arduino Yun.

La carte Arduino Yun (Figure xx) est une carte Arduino open source mais possédant quelques fonctionnalités avancées pour réaliser des prototypes d'objets connectés (Internet of things). En effet, avec la carte Arduino Yun on peut faire communiquer des capteurs directement avec un réseau local ou internet, qu'il soit filaire ou wifi.

Le choix de la carte Arduino yun est motivé par son faible cout et l'intégration de la communication distante, via internet. La carte possède, également, un convertisseur analogique numérique avec une résolution de 12 bits et une tension de référence de 5V. Le convertisseur accepte 6 entrées analogiques multiplexées. Ainsi, la carte s'interface avec les capteurs courants, température et tension à travers ces entrées analogiques.

FIGURE 11: ARDUINO YUN (ARDUINO.CC)

Arduino Yun a 2 microprocesseurs qui peuvent communiquer. D'un côté, il y a le processeur ATmega qui gère les capteurs et les échanges de données entre eux, de l'autre il y a le processeur Linino, qui lui s'occupe de la relation de la carte avec internet et les réseaux. Pour pouvoir communiquer entre eux, ils utilisent un bridge (pont). Voici un schéma détaillant ces échanges :

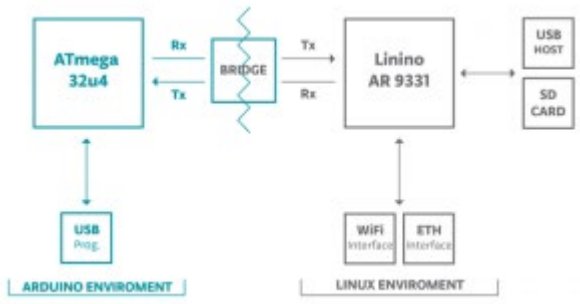

FIGURE 12: SCHEMATIQUE D'UN ARDUINO YUN (SOURCE: ARDUINO.CC)

3.5 Design du circuit d'adaptation

La carte d'acquisition assure un interfaçage adéquat entre les capteurs de tension et des courants et la carte Arduino Yun. La figure suivante illustre un schéma bloc de la carte d'acquisition :

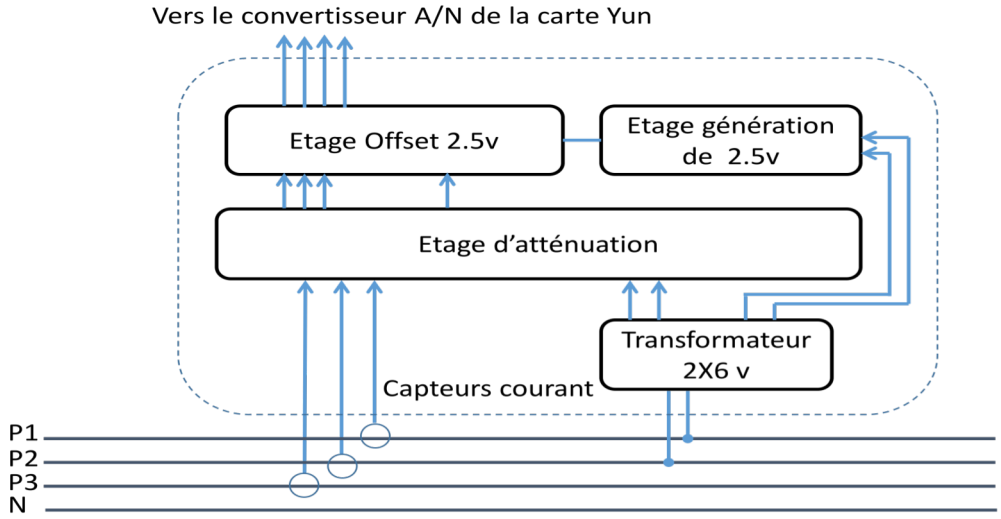

FIGURE 13 : SCHEMA BLOC DE LA CARTE D'ACQUISITION

L'étage d'atténuation assure l'adaptation des sorties des capteurs de courant et de tension vers la plage de tension -2.5V à 2.5V. Cet étage est constitué essentiellement par des résistances. La disposition et les valeurs des résistances sont expliqués par la suite.

L'image de la tension secteur est fournie par le premier secondaire d'un transformateur de tension 2X6V qui constitue un capteur de tension. Le second secondaire est utilisé pour alimenter la carte avec une tension 5V continue en passant par un étage de redressement et de régulation. La figure xx présente le schéma électrique de l'étage d'alimentation et d'atténuation de la tension captée. L'atténuation de la tension en sortie du transformateur est assurée par un diviseur de tension à l'aide de deux résistances R15 et R16. Les valeurs des résistances sont liées par la formule suivante :

$$V_{cca} = \frac{R15}{R15 + R16} V_{cc}$$

Avec V_{cc} la tension crête à crête à la sortie du transformateur : $V_{cc} = 2x\sqrt{2}x6V \approx 17V$ et V_{CCa} valeur de la tension atténuée : $V_{CCa}=5V$. On a choisit des valeurs de résistances élevée pour limiter le courant absorbé.

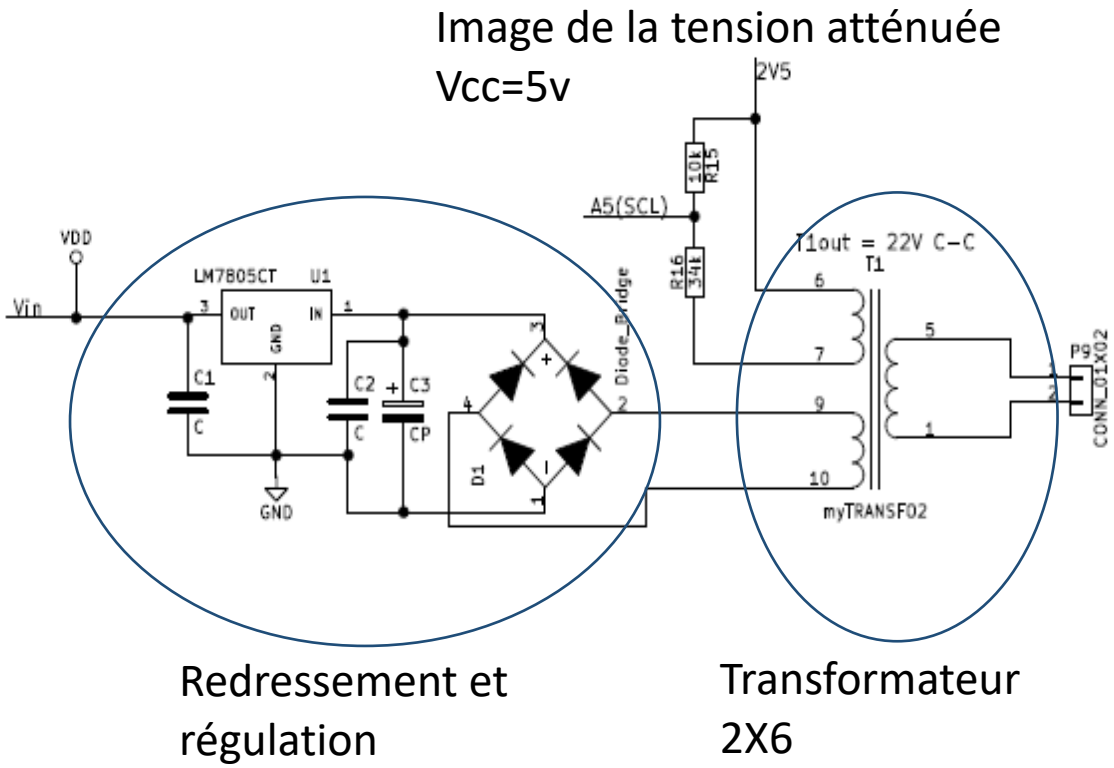

FIGURE 14: SCHEMA ELECTRIQUE DE L'ETAGE D'ALIMENTATION ET D'ATTENUATION DE LA TENSION

L'atténuation ou l'adaptation de la sortie du capteur du courant est assurée par une résistance de charge en parallèle avec la sortie du capteur (voir section 3.1.1). La figure xx illustre le schéma électrique de la résistance connecté avec le connecteur audio femelle du capteur courant.

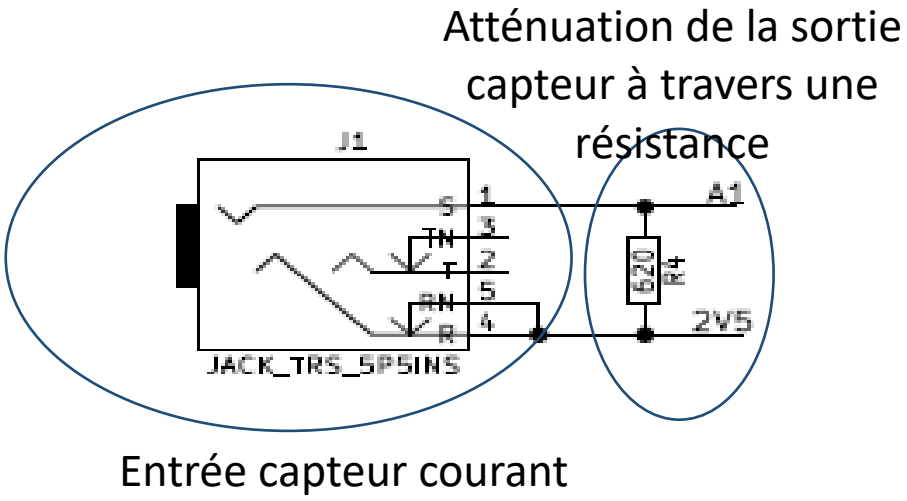

FIGURE 15: SCHEMA ELECTRIQUE DE LA RESISTANCE CONNECTE AVEC LE CONNECTEUR AUDIO

L'étage Offset 2.5v permet d'ajouter une composante continue de 2.5V à la sortie de l'étage d'atténuation. Ainsi, les tensions appliquées aux entrées du convertisseur analogique

numérique de la carte Arduino sont comprises entre 0V et 5V. La figure xx présente le schéma électrique de circuit de génération de 2.5V stable à partir de 5V.

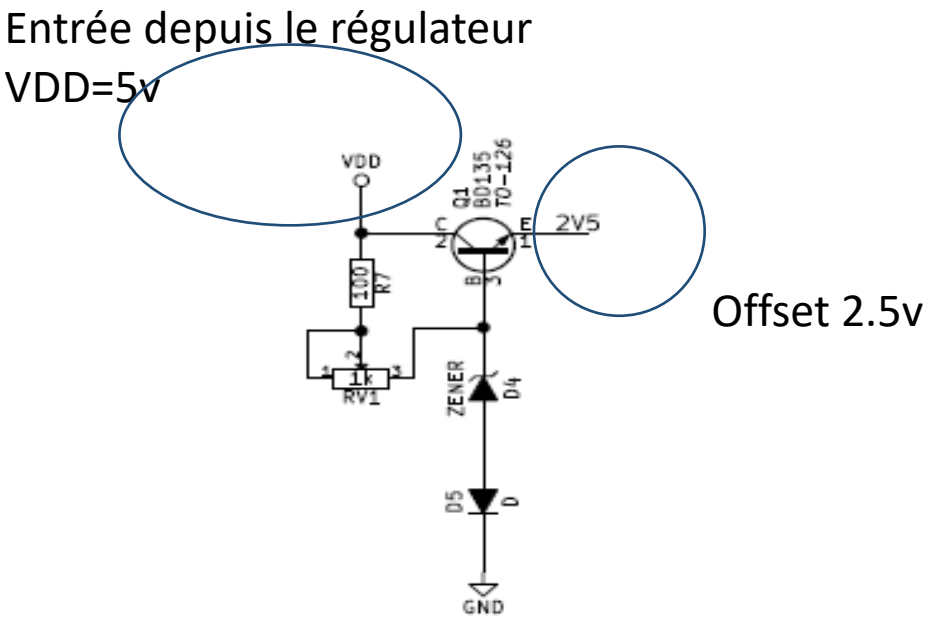

FIGURE 16: SCHEMA ELECTRIQUE DE CIRCUIT DE GENERATION DE 2.5V STABLE A PARTIR DE 5V

La diode zener permet d'avoir une tension de valeur 2.5V au niveau de la cathode. La résistance variable (RV1) permet d'ajuster le courant de la diode Zener pour avoir une tension fixe de 2.5V. Afin d'amplifier le courant à la sortie de la diode zener on utilise un transistor mais il introduit une chute de tension de 0.6V au niveau de la jonction BE. Pour compenser cette chute de tension on a ajouté la diode D5.

Un prototype a été préparé et assemblé. Le résultat final est donné en Figure xx.

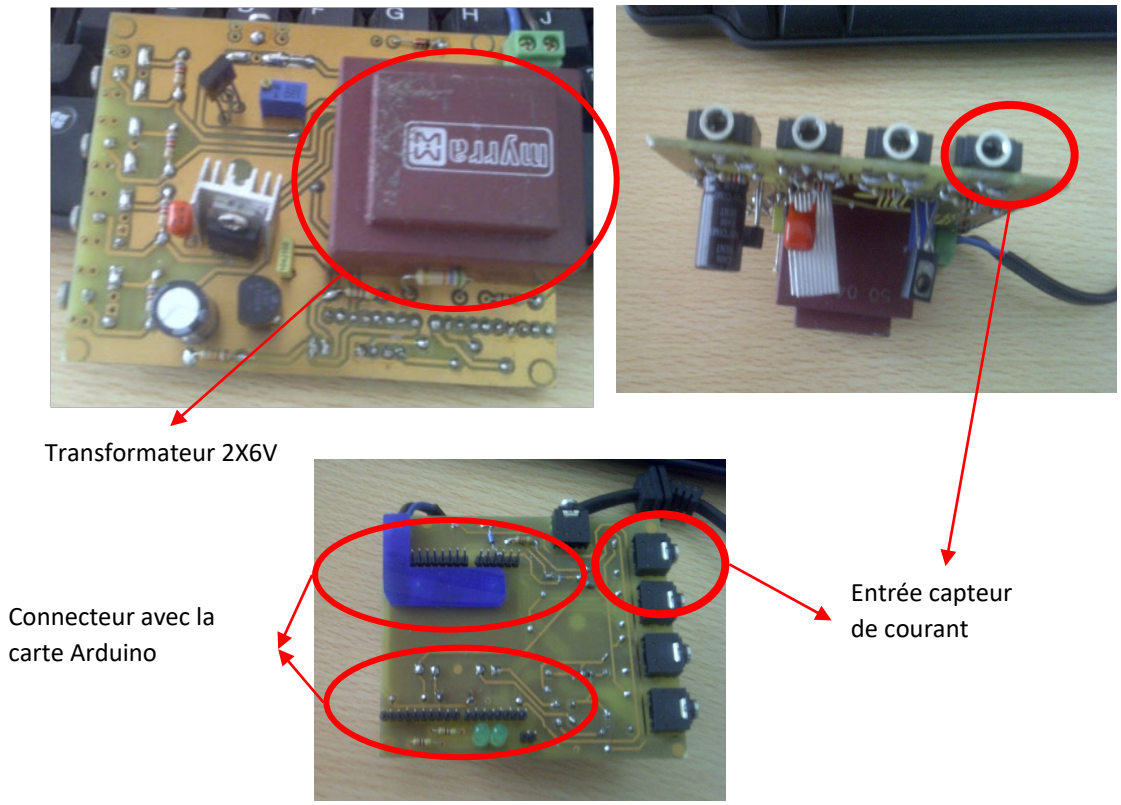

FIGURE 17: PROTOTYPE DE LA CARTE D'ACQUISITION

3.6 La matrice des coûts

L'une des conditions d'un éventuel succès de ce projet est le coût de fabrication doit rester faible. La matrice de coût de la solution propose est donnée en tableau suivant :

TABLE 1: MATRICE DE COUT DU MODEM

Composants	Coût (USD)
Carte Arduino Yun	70
Capteurs de courant SCT-013-000	20
Capteur de température DS18B20	6.7
Transformateur de tension 2X6	3
Régulateur de tension LM7805	0.5
Pont redresseur monophasé 1.5 A	0.5
Capacités, résistances, diodes, transistor	0.5
Connecteurs	0.2
Circuit imprimé	10
Total	**110.4**

Le contrôleur Arduino semble être l'élément qui pèse le plus dans le coût total de la solution. Il est recommandé de chercher une solution alternative moins cher. Aux coûts cités en haut, il faut ajouter le coût d'un éventuel boitier et un packaging.

3.7 Etude théorique de la partie métrologie du compteur électrique

Après la phase d'acquisition des courants et des tensions par la carte Arduino Yun, les échantillons sont conservés et utilisés pour calculer les valeurs efficaces ou (RMS) des courants et des tensions. La valeur efficace est la racine de la valeur moyenne du carré d'une grandeur physique.

Soit un signal discret X(n), on définit la valeur moyenne par la formule suivante:

$$< X^2 >= \frac{1}{N}\sum_{n=0}^{N} X^2(n) \qquad (1)$$

Donc la valeur RMS d'une grandeur physique discret se présente par :

$$X_{RMS} = \sqrt{\frac{1}{N}\sum_{n=0}^{N} X^2(n)} \qquad (2)$$

On peut ainsi calculer les valeurs efficaces de la tension et le courant à partir des formules suivantes :

$$U_{RMS} = \sqrt{\frac{1}{N}\sum_{n=0}^{N} U^2(n)} \qquad (3)$$

$$I_{RMS} = \sqrt{\frac{1}{N}\sum_{n=0}^{N} I^2(n)} \qquad (4)$$

Où N est égale au nombre d'échantillons effectués dans une seconde .

Une fois que ces deux grandeurs sont calculées, on peut passer aux calculs des puissances et des énergies.

La puissance active est la valeur moyenne de la puissance instantanée. Notée P elle s'exprime en Watts (W). Elle dépend des valeurs efficaces de u et de i et du déphasage φ entre les deux grandeurs.

Cette grandeur s'exprime par les deux formules suivantes :

$$P = U_{RMS} * I_{RMS} * \cos\varphi \qquad (5)$$

$$P = \sqrt{\frac{1}{N}\sum_{n=0}^{N} V(n) * I(n)} \qquad (6)$$

La puissance réactive est exprimée par analogie avec la puissance active. L'unité de cette puissance est le Voltampère réactive (VAR).

Elle est définit par :

$$Q = U_{RMS} * I_{RMS} * \sin\varphi \qquad (7)$$

$$Q = \sqrt{S^2 - P^2} \qquad (8)$$

Où S est la puissance apparente, elle est exprimé en Voltampère (VA). elle se calcule par :

$$S = U_{RMS} * I_{RMS} \quad (9)$$

Par définition, le facteur de puissance ($\cos \varphi$) est égale au rapport de la puissance active P sur la puissance apparente S et peut varier de 0 à 1.

$$\cos \varphi = \frac{P\ (KW)}{S\ (KVA)} \quad (10)$$

À la fin on définit l'énergie consommée par la formule suivante :

$$E = P * N \quad (11)$$

4 Back-end et base de données

Outre les terminaux mobile et web, un backend est nécessaire pour gérer le stockage des données ainsi que leurs traitements. Deux options sont disponibles ; self-hosted back end ou une solution cloud. Le choix a été porté sur cette dernière pour les raisons suivantes ; coût, simplicité et scalability.

Coût ; le coût des solutions cloud a nettement baissé au point qu'il est devenu beaucoup moins cher qu'une solution self-hosted. En fait, dans cette dernière en plus du coût du matériel (serveur et connection internet), il faut ajouter un coût de maintenance. Un calcul simple d'un serveur à 2000 euros (pour une durée de vie de 3 ans) et un coût de maintenance annuelle de 1000 euros. Le coût mensuel revient 83 euros par mois.

Le choix a été porté sur l'utilisation de Parse.com qui est une solution développée par Facebook et qui devenu récemment open-source. Parse.com offre jusqu'à 20 Go de données stockées gratuitement. Une autre alternative étudiée est Firebase développé par Google et qui se positionne comme étant une base de données temps réel pour l'internet des objets. Le choix s'est penché vers Parse car elle a une communauté riche de 600 000 applications développées. En plus, elle permet d'implémenter un backend logic que Parse lui donne le nom de Parse Cloud Code. Ce dernier, il est nécessaire pour pouvoir analyser les données et initier des alertes en case de problème. La solution finale est schématisée en Figure xx.

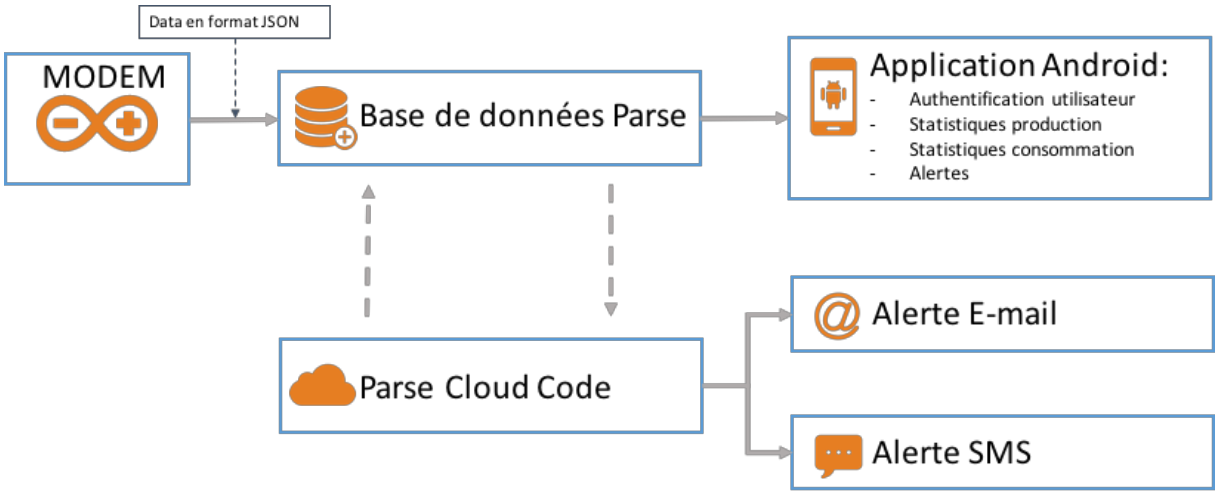

FIGURE 18: ARCHITECTURE DE LA PARTIE SOFTWARE DU SYSTEME DE MONITORING

Parse Dashbord

Il s'agit d'une interface graphique qui permet les opérations de base pour explorer les données enregistrées. Il permet également d'effectuer des opérations de maintenance en manuel. Figure xxx présente une capture d'écrans du tableau du bord. A gauche on a la liste des Class et dans chacune des class, chaque objet a un identifiant unique.

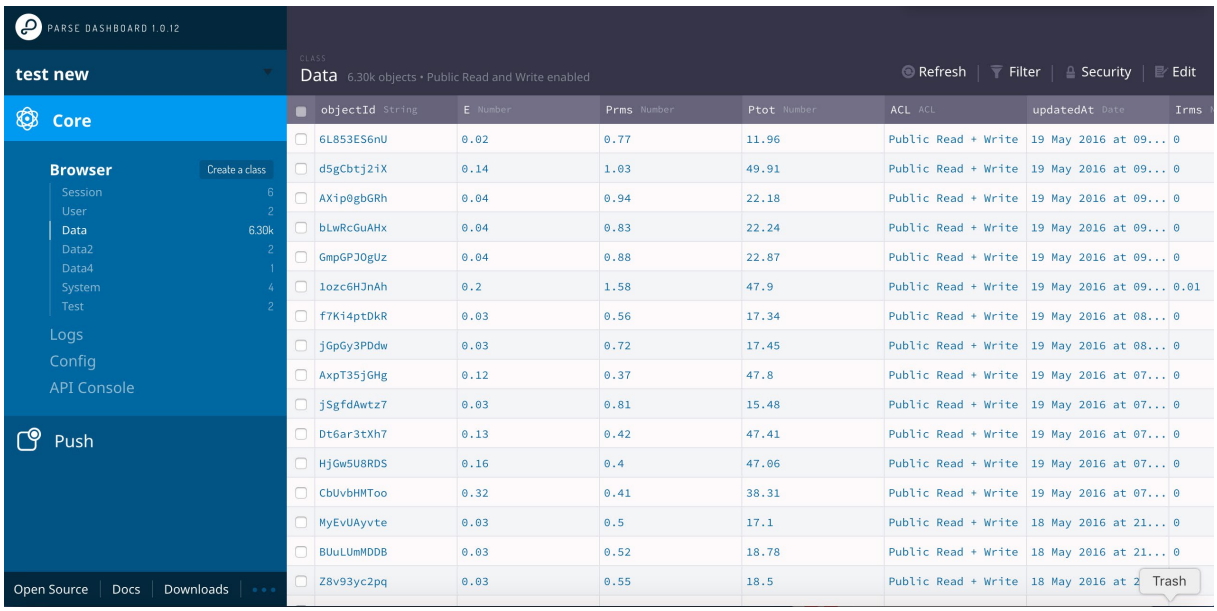

FIGURE 19: EXEMPLE DASHBORD DE PARSE

Parse Cloud Code

Le Cloud Code permet d'exécuter des opérations sur le serveur et éviter de les exécuter sur le terminal mobile. Il permet de réaliser des opérations complexes difficilement exécutable sur un terminal mobile. Il est construit sur le SDK JavaScript de Parse.

Il offre par défaut des fonctions qui s'exécutent automatiquement à chaque opération d'écriture et de suppression. Ces fonctions sont très utiles pour les opérations de sécurités (vérifier que l'opération n'est pas un spam) mais également pour réaliser des calculs

statistiques (ex. comptage des opérations). Ces fonctions ayant les noms suivants (beforeSave, afterSave, beforeDelete, afterDelete) seront d'une grande utilité pour ce projet (section xxx).

Parse Cloud Job

Il s'agit d'un nombre d'opérations qui s'exécute à un intervalle de temps bien précis. On peut imaginer que une fois par jour un nombre d'opérations s'exécute pour calculer les statistiques de production et de consommation de la journée. Le Job peut également lancer des alertes en cas où il détecte des anomalies.

Parse Push Server

Il s'agit de la solution Parse pour envoyer des alertes sur les smartphones android et iOS.

Parse Authentification

Il s'agit de la solution de gestion d'utilisateur offerte par Parse.

Une base de données relationnelle

La base de données de Parse permet de créer les liens entre les objets. Une telle fonction est très utile dans ce projet. En fait, il est possible de lier un utilisateur à son système. Le modèle de données adopté sera donné en section xxx.

5 Code de la solution embarquée

Le code embarque doit permettre la réalisation des tâches suivantes :

- Convertir les données mesurées à leurs réels (tension ou courant)
- Formater les données dans le bon format (JSON ?)
- Détecter la connection internet
- Maintenir une horloge
- Stocker les données dans une carte SD en cas où il n'y a pas une connection internet
- Envoyer les données vers la base de données
- Un voyant LED qui montre l'état de fonctionnement du modem

Le code suivant permet de réaliser les tâches citées est donné en annexe 1.

6 Design de l'application mobile

6.1 Le choix de la plateforme et méthodologie adoptée

Un autre élément important pour la réussite de ce projet est une diversité de terminaux pour que l'utilisateur puisse accéder aux données de son système. Le choix a été porte pour que le système soit développé autour de trois technologies :

- Une application mobile Android
- Une application mobile iPhone
- Une application web

Il a été décidé d'aller vers une stratégie « mobile first » qui consiste à développer les applications mobiles avant la mise en ligne d'une version web. Ce choix est motivé par deux raisons :

- Le temps passé sur les terminaux mobiles de types smartphone vient de dépasser l'accès via un PC
- Prendre avantage de la technologie des « pushs notifications » qui permettent d'envoyer des messages (alertes) courts à l'utilisateur pour les mettre au courant de l'état de fonctionnement de son système.

Il est bien connu que les campagnes de push s'ils ont utilisé adéquatement augmente l'engagement de l'utilisateur. La conservation d'énergie devient une partie intégrante du quotidien.

6.2 Modèle de données

Un modèle de donnée simple est considéré pour ce travail. Il est constitué de 8 tableaux relationnels. Chaque tableau enregistre un certain nombre d'information. Les relations entre les tableaux sont gérées par des pointeurs :

Tableau **User**: contient les paramètres de l'utilisateur avec des pointeurs vers ces systèmes

Tableau **System**: contient les paramètres de tous les systèmes.

Tableau **Data**: contient les données brutes et en temps réels venant de tous les modems

Statistic_hourly: contient les statistiques horaires de la production et de la consommation. Ces données sont générés par un cloud code.

Statistic_daily: contient les statistiques journalières de la production et de la consommation. Ces données sont générés par un cloud code.

Statistic_weekly: contient les statistiques hebdomadaires de la production et de la consommation. Ces données sont générés par un cloud code.

Statistic_monthly: contient les statistiques mensuelles de la production et de la consommation. Ces données sont générés par un cloud code.

Statistic_yearly: contient les statistiques annuelles de la production et de la consommation. Ces données sont générés par un cloud code.

User
name : String
system : System

System
size: String
panel_type: String
inverter_type: String
mac_adress: String

Data
mac_adress: String
E: Number
Prms: Number
Ptot: Number
Irms: Number

Statistic_hourly
system: System
Hour: Number
E_prod: Number
E_cons: Number
Temp: Number

Statistic_daily
system: System
Hour: Number
E_prod: Number
E_cons: Number
Temp: Number

Statistic_weekly
system: System
Hour: Number
E_prod: Number
E_cons: Number
Temp: Number

Statistic_monthly
system: System
Hour: Number
E_prod: Number
E_cons: Number
Temp: Number

Statistic_yearly
system: System
Hour: Number
E_prod: Number
E_cons: Number
Temp: Number

TABLE 2: MODELE DE DONNEES

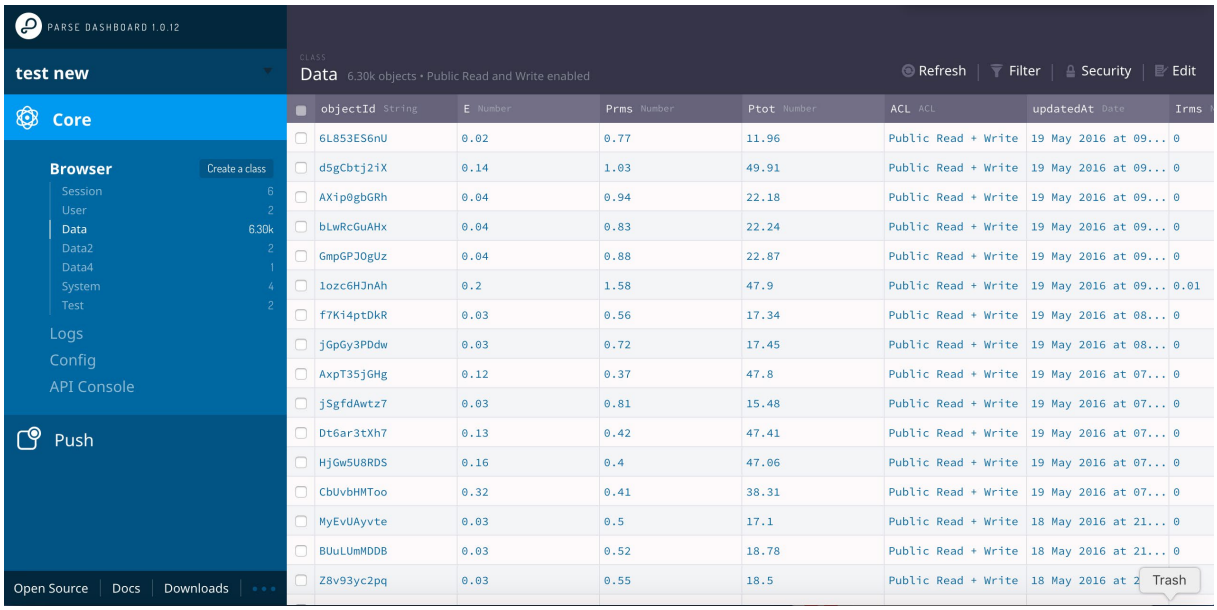

FIGURE 20: TABLEAU DE BORD DU MODELE DE DONNEE ; TABLEAU DATA

6.3 Analyse fonctionnel

L'application doit permettre d'exécuter les fonctionnalités suivantes :

Authentification ; devrait permettre à un utilisateur de créer un compte et accéder à ses données. Parse offre une solution d'authentification et gestion des utilisateurs au nom de

ParseUI. Cette dernière est une collection d'interfaces utilisateurs visant à fluidiser et à simplifier l'authentification de l'utilisateur, l'affichage de la liste de données, et d'autres éléments d'application commune. Par défaut, ParseUI offre la possibilité d'ajouter l'authentification avec les comptes sociaux les plus populaires (Facebook et Twitter). Une telle fonctionnalité évitera à l'utilisateur de créer un compte spécifique à ce système. L'API gère également la perte du mot de passe en envoyant un lien pour régénérer le mot de passe via l'adresse email de l'utilisateur. Elle offre également des fonctionnalités plus avancées de sécurité que nous n'avons ni explorées ni exploitées à ce stade du projet.

La bibliothèque est disponible en open-source sur GitHub sur ces deux liens pour les plateformes Android et iOS.

https://github.com/ParsePlatform/ParseUI-Android

https://github.com/ParsePlatform/ParseUI-iOS

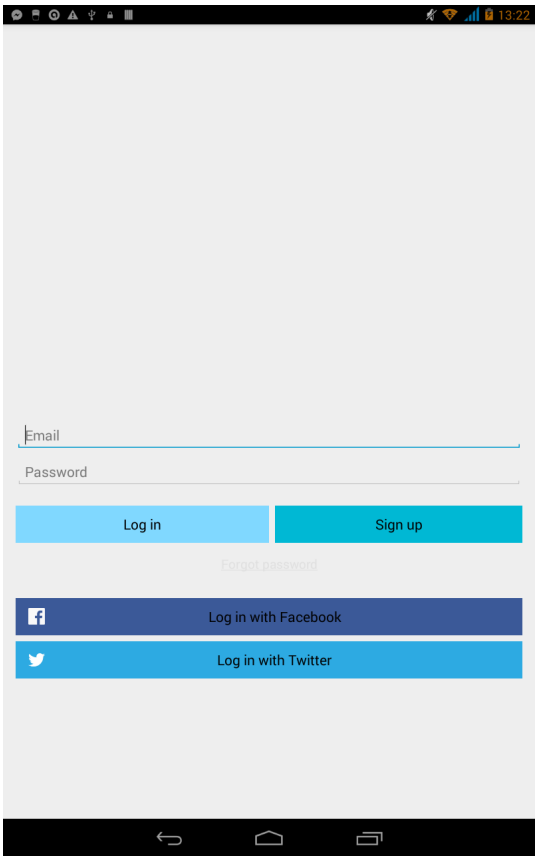

FIGURE 21 : CAPTURE D'ECRAN DE L'APPLICATION ANDROID; INTERFACE D'AUTHENTIFICATION

Connecter l'utilisateur aux données de son système ; une fois l'utilisateur crée un compte, il sera invité à ajouter un modem et le descriptif de son système. Le modem est identifié par son adresse mac. L'utilisateur retrouvera l'adresse mac du modem (de l'Arduino Yun) en se connectant via wifi et en local avec un PC au modem. Le modem dès qu'il est en marche et qu'il détecte une connection internet, il envoie les données vers le serveur Parse. Cette approche présente des risques de sécurités étant donné que n'importe qui ayant un compte ou pas peut écrire sur la base de données. Pour simplifier les choses, à ce stade du projet ce

choix est maintenu mais il est recommandé dans le future de développer des solutions plus sophistiquées pour éviter une telle prêche de sécurité.

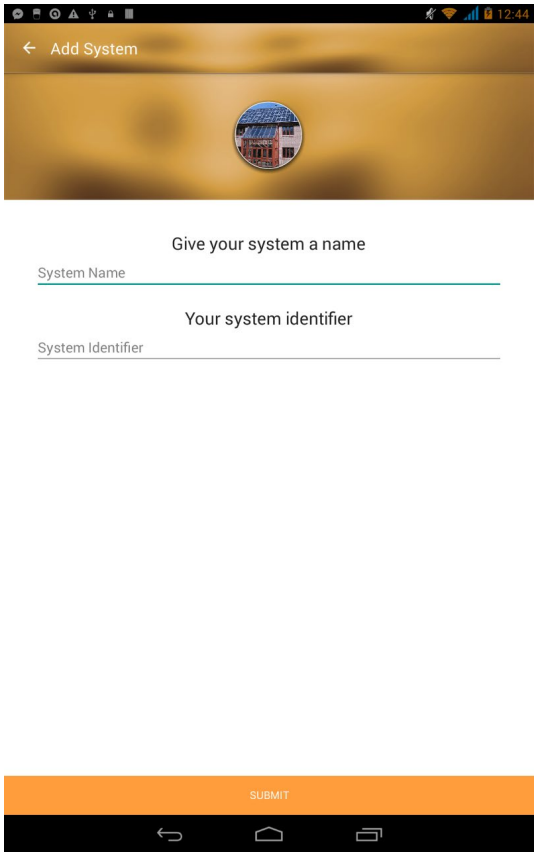

FIGURE 22 : CAPTURE D'ECRAN DE L'APPLICATION ANDROID; CONNECTER L'UTILISATEUR A SON SYSTEME

Afficher les données de l'utilisateur ;

Une fois la connection entre un utilisateur et un modem est crée. L'application cherchera les données générées par le modem en question et les présenter sous forme de graphe. Cette partie est le cœur même de l'application. Nous avons choisi de procéder à développer cette partie sur deux étapes :

Etape 1 : présenter uniquement l'historique en brute sans traitement.

Etape 2 : présenter des statistiques journalières, hebdomadaires, mensuels et annuelles.

Lors de ce travail, le travail s'est limité à l'Etape 1. Une fois le premier prototype est mis en marche, il sera facile d'étendre vers des éléments statistiques plus avancés avec Parse Cloud Code.

Pour afficher les graphes, nous avons fait appel à l'API MPAndroidChart developpe par Philipp Jahoda et disponible sous une license Apache sur ce lien :

https://github.com/PhilJay/MPAndroidChart

MPAndroidChart est une puissante bibliothèque de vue graphique pour Android, supportant des graphiques de type ligne, bar, camembére et radar. Elle permet également la gestion des

mises a l'échelle, le glissement et les animations. La combinaison des différents types de graphiques permettra de créer quasiment une infinité de possibilité.

En Figure xx est donnée un exemple de capture d'écran d'une courbe de puissance. Il est également possible de choisir une période de temps à afficher des données.

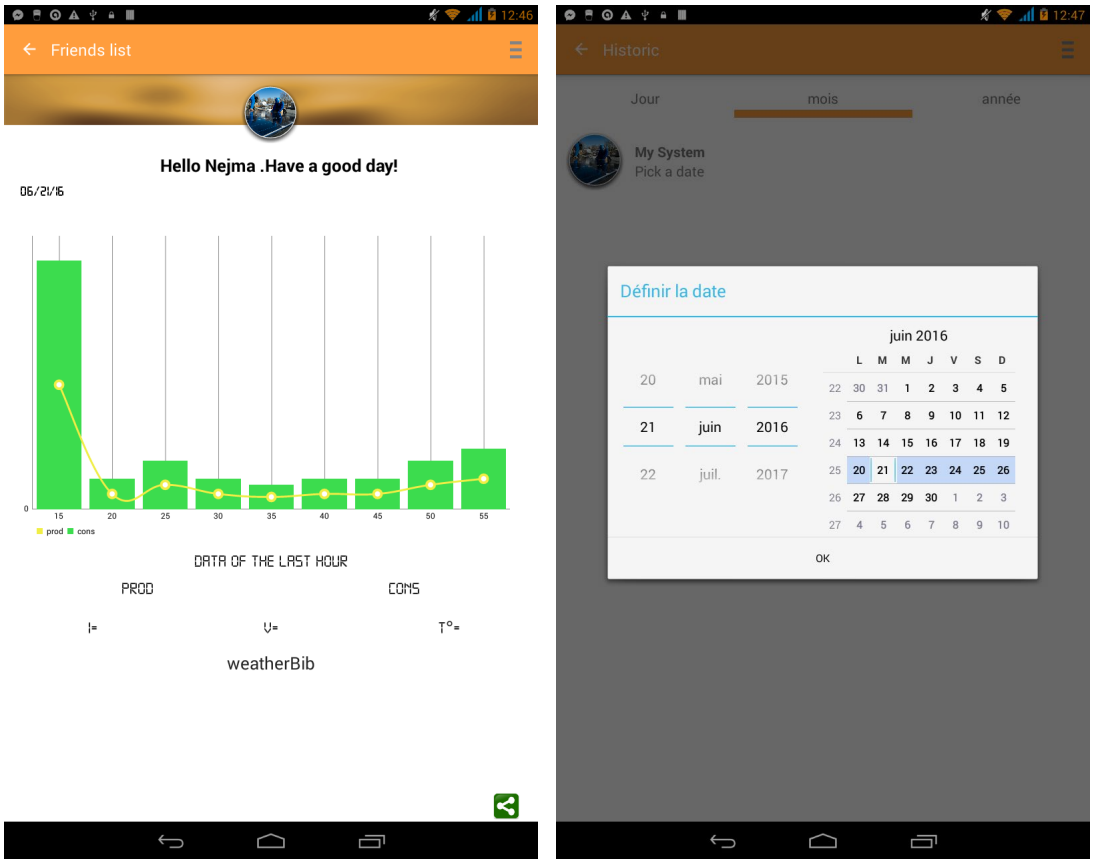

FIGURE 23 : CAPTURE D'ECRAN DE L'APPLICATION ANDROID; VISUALISATION DES DONNEES TEMPS REELS (GAUCHE), CHOIX DE LA PERIODE D'AFFICHAGE (DROITE)

Liste des systèmes de l'utilisateur

L'utilisateur peut avoir plusieurs systèmes si par exemple il a plusieurs foyers. Ainsi pour chacun des systèmes il est invité d'ajouter l'adresse mac du nouveau modem et automatiquement la connection va se créer. Via la vue Liste de systèmes, l'utilisateur peut gérer l'ajout mais également la suppression d'un système. Il sera également invité à spécifier les données du système ; taille, type d'onduleur, type de panneau.

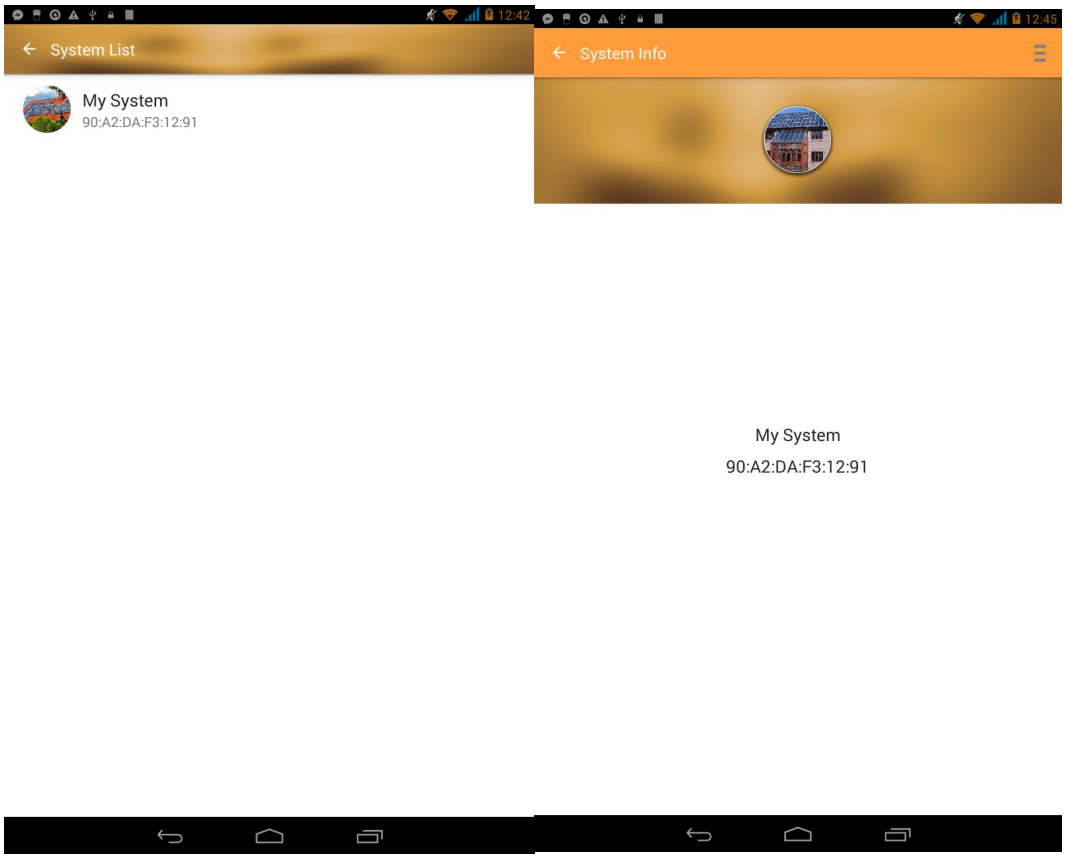

FIGURE 24 : CAPTURE D'ECRAN; VUE DES SYSTEMES

Préférences de l'utilisateur

Via cette vue, l'utilisateur peut gérer les fonctionnalités de base de son compte : nom, adresse, notifications, phone, adresse email, etc.

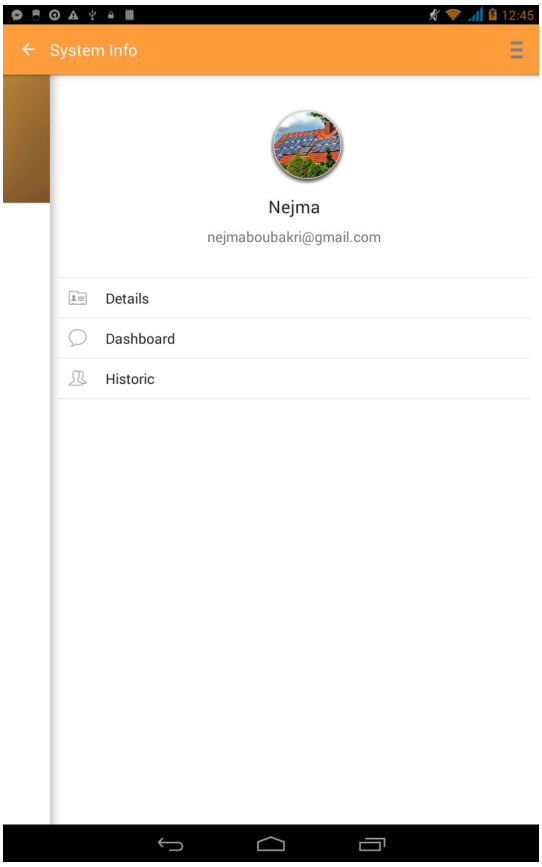

FIGURE 25 : CAPTURE D'ECRAN; VUE DES PARAMETRES UTILISATEURS

7 Code Cloud

Le modem envoi uniquement les données en temps réel et à l'état brute sans analyse. Cependant, l'utilisateur a besoin de plus d'informations. En particulier, il serait intéressé par des statistiques journalières, hebdomadaires et mensuelles de la production ainsi que la consommation. Le Parse Cloud Code permettra de manipuler les données enregistrées dans la base de données. Il est également possible d'initier des alertes quand par exemple le système est à l'arrêt ou quand la consommation sur un mois dépasse la production.

Plus d'infos

```
//After Save de donnée venue de l'Arduino Yun
Parse.Cloud.afterSave("Data", function(request) {
//Récupérer de donnée l'heure correspondante . Cette donnée est sous la
forme hh:mm
//ce n'est pas "created at", car elle peut etre une donnée de SD card
var str = request.object.get ("H");

//créer une donnée de l'heure correspondante HH
var res = str.substring(0, 2);

//Récupérer de donnée l'energie correspondante
var energy = request.object.get ("E");

//Récupérer les données la date correspondante jj/mm/aa
```

```
var stra = request.object.get ("D");

//Récupérer les données du mois correspondant mm/aa
var month = stra.substring(2);

//Récupérer les données de l'adresse MAC
var mac = request.object.get ("MAC");

// Créer query avec PARSE, la table de donnée/heure
 var query = new Parse.Query("DataH");
 query.equalTo("h", res);
 query.find({
//******************
    success: function(results) {
// S'il y a une donnée dont l'heure corresponde à celle de la donnée qui
vient d'être enregistrée
    if(results.length>0)
        {
//cumuler l'energie sur la donnée existante
      var object = results[0];
      var en = object.get("energy");
      object.set("energy",en+energy);
      object.save();
      console.log("Save ko datah");
    }
// S'il n'y a pas une donnée dont l'heure corresponde à celle de la
donnée qui vient d'être enregistrée
    else{
/* Créer une donnée dans Donnée/heure avec
energy,heure,date et adresse mac
*/
    var DataH = Parse.Object.extend("DataH");
    var currentPrice = new DataH();
     currentPrice.set("energy",energy);
     currentPrice.set("h",res);
     currentPrice.set("d",stra);
     currentPrice.set("mac",mac);
     currentPrice.save(null, {
    success: function(currentPrice) {
        console.log("update DataH succeed");
       },
       error: function (error) {
        response.error(error);
        console.log("Save ko");
    }
    });

    }
    },
    //**************************
    error: function(error) {
      console.error("Error finding related comments " + error.code + ": "
+ error.message);
    }


  }

  );
```

```
// Créer query avec PARSE, la table de donnée/jour
 var quer = new Parse.Query("DataJ");
 quer.equalTo("d", stra);
 quer.find({

// S'il y a une donnée dont la date corresponde à celle de la donnée qui
vient d'être enregistrée

    success: function(results) {

    if(results.length>0)
        {

////////////////////////// CUMULATION DE L'ENERGIE SUR LA DONNEE
EXISTANTE//////////////////

      var object = results[0];
      var en = object.get("energy");
      object.set("energy",en+energy);
      object.save();
      console.log("Save ko datah");
    }
    else{
    ////////////////////////// CREATION D'UNE DONNEE DATA/JOUR AVEC LES
DONNEEs RECUES
     var DataH = Parse.Object.extend("DataJ");
     var currentPrice = new DataH();
     currentPrice.set("energy",energy);
     currentPrice.set("mac",mac);
     currentPrice.set("d",stra);
     currentPrice.save(null, {
    success: function(currentPrice) {
        console.log("update DataH succeed");
       },
       error: function (error) {
        response.error(error);
        console.log("Save ko");
    }
    });


    }
    },
    //**************************
    error: function(error) {
      console.error("Error finding related comments " + error.code + ": "
+ error.message);
    }


  }

  );


  //**************************CETTE PARTIE ET CELLE DU DONNEE/MOIS
C'EST avec les mmes techniques


  /* var quer1 = new Parse.Query("DataM");
 quer1.equalTo("m", month);
```

```
 quer1.find({

 //******************
    success: function(results) {

    if(results.length>0)
        {
      var object = results[0];
      var en = object.get("energy");
      object.set("energy",en+energy);
      object.save();
      console.log("Save ko datah");
    }
    else{

     var DataM = Parse.Object.extend("DataM");
     var currentPrice = new DataM();
     currentPrice.set("energy",energy);
     currentPrice.set("mac",mac);
     currentPrice.set("m",month);
     currentPrice.save(null, {
     success: function(currentPrice) {
        console.log("update DataM succeed");
       },
       error: function (error) {
        response.error(error);
        console.log("error save DataM");
    }
    });

    }
    },
    //*************************
    error: function(error) {
      console.error("Error finding related comments " + error.code + ": "
+ error.message);
    }


  }

  );*/

});
```

8 Test du modem

CLASS
Data 6.30k objects • Public Read and Write enabled

Refresh | Filter | Security | Edit

objectId String	E Number	Prms Number	Ptot Number	ACL ACL	updatedAt Date	Irms N
6L853ES6nU	0.02	0.77	11.96	Public Read + Write	19 May 2016 at 09...	0
d5gCbtj2iX	0.14	1.03	49.91	Public Read + Write	19 May 2016 at 09...	0
AXip0gbGRh	0.04	0.94	22.18	Public Read + Write	19 May 2016 at 09...	0
bLwRcGuAHx	0.04	0.83	22.24	Public Read + Write	19 May 2016 at 09...	0
GmpGPJOgUz	0.04	0.88	22.87	Public Read + Write	19 May 2016 at 09...	0
1ozc6HJnAh	0.2	1.58	47.9	Public Read + Write	19 May 2016 at 09...	0.01
f7Ki4ptDkR	0.03	0.56	17.34	Public Read + Write	19 May 2016 at 08...	0
jGpGy3PDdw	0.03	0.72	17.45	Public Read + Write	19 May 2016 at 08...	0
AxpT35jGHg	0.12	0.37	47.8	Public Read + Write	19 May 2016 at 07...	0
jSgfdAwtz7	0.03	0.81	15.48	Public Read + Write	19 May 2016 at 07...	0
Dt6ar3tXh7	0.13	0.42	47.41	Public Read + Write	19 May 2016 at 07...	0
HjGw5U8RDS	0.16	0.4	47.06	Public Read + Write	19 May 2016 at 07...	0
CbUvbHMToo	0.32	0.41	38.31	Public Read + Write	19 May 2016 at 07...	0
MyEvUAyvte	0.03	0.5	17.1	Public Read + Write	18 May 2016 at 21...	0
BUuLUmMDDB	0.03	0.52	18.78	Public Read + Write	18 May 2016 at 21...	0
Z8v93yc2pq	0.03	0.55	18.5	Public Read + Write	18 May 2016 at 21...	0

CLASS
Data 6.30k objects • Public Read and Write enabled

Refresh | Filter | Security | Edit

Irms Number	MAC String	createdAt Date	D String	H String	Add a new column
0	90:A2:DA:F3:12:91	19 May 2016 at 09...	05/19/16	10:48	
0	90:A2:DA:F3:12:91	19 May 2016 at 09...	05/19/16	10:45	
0	F3:12:91	19 May 2016 at 09...	05/19/16	10:42	
0	F3:12:91	19 May 2016 at 09...	05/19/16	10:39	
0	F3:12:91	19 May 2016 at 09...	05/19/16	10:36	
0.01	F3:12:91	19 May 2016 at 09...	05/19/16	10:33	
0	F3:12:91	19 May 2016 at 08...	(undefined)	09:03	
0	F3:12:91	19 May 2016 at 08...	(undefined)	09:00	
0	F3:12:91	19 May 2016 at 07...	(undefined)	08:57	
0	F3:12:91	19 May 2016 at 07...	(undefined)	08:54	
0	F3:12:91	19 May 2016 at 07...	(undefined)	08:51	
0	F3:12:91	19 May 2016 at 07...	(undefined)	08:48	
0	F3:12:91	19 May 2016 at 07...	(undefined)	08:39	
0	F3:12:91	18 May 2016 at 21...	(undefined)	22:57	
0	F3:12:91	18 May 2016 at 21...	(undefined)	22:51	
0	F3:12:91	18 May 2016 at 21...	(undefined)	22:48	

Trash

9 Conclusions

L'adoption de l'énergie solaire photovoltaïque dans le secteur résidentiel est en pleine expansion. Chaque année des millions de nouveaux systèmes sont installés. La majorité de ces systèmes ne sont pas équipé par un système de surveillance. Le besoin d'un système

simple et pas cher est réel. Un tel système aura pour vocation de surveiller le bon fonctionnement du système et lancer des alertes en cas d'anomalies.

Lors de ce stage, j'ai travaillé avec l'équipe technique de la société Oroora Solar sur le développement d'une solution de monitoring en ligne des installations solaires. Un modem électronique a été conçu pour mesurer l'énergie produite et celle consommée par le foyer et l'envoyer en temps réel a un serveur dans le cloud. Le modem est conçu autour d'un microcontrôleur de type Arduino Yùn. Le prototype est fonctionnel et capable d'envoyer des données comme prévu.

J'ai également développé un code embarqué qui permet de faire un formatage des données brutes collectées avant l'envoie vers le serveur. Le code permet de gérer les coupures d'internet (offline mode) en stockant les données temporairement sur une carte SD en attendant le retour de la connection et l'envoie de ces mêmes données vers le serveur.

J'ai également participé au développement d'une application mobile Android qui sera l'interface qu'avec l'utilisateur peut accéder à ces données. La version actuelle ne permet l'affichage que des données temps réels.

L'ambition est de pouvoir appliquée des analyses poussées des données collectées et détectée automatiquement les anomalies. Le système doit être capable de lancer des alertes aux équipes de maintenance. C'est dans cette optique qu'un premier code cloud a été implémenté. Il permet de réaliser actuellement des opérations basiques statistiques. A terme, le code permettra l'envoie des alertes par sms, email ou push notifications à l'utilisateur et à l'installateur.

A l'état actuel, le système n'est encore qu'à l'état de prototype. Sur le moyen terme, il faut travailler sur les éléments suivants :

- Baisser le coût du modem ; avec un coût matériel estimé à xx, la solution est considérée encore cher et il faut travailler sur des solutions alternatives qui baisserons ce coût.
- Améliorer la robustesse et la sécurité du code embarqué.
- Améliorer le design de l'application Android et ajouter des nouvelles fonctionnalités
- Ajouter d'autres fonctions statistiques dans le Cloud Code
- Etendre le développement vers d'autres plateforme ; iOS et web.

10 Annexes

10.1 Code Arduino

```
// Vous allez trouver plusieurs Serial.print en commentaire, ils sont
utilisé pour visualiser le log sur PC avec le arduino IDE. Cependant, ils
```

```
utilisent suffisament de memoires au point qu'ils  empechent le bon
fonctionnement de la carte (difficulte d'envoie des données).

#include <math.h>
#include <DHT.h>
#include "EmonLib.h"
#include <Bridge.h>
#include <Process.h>
#include <ArduinoJson.h>
#include <FileIO.h>
#include <MemoryFree.h>
int last=1;


// Constants PARSE
const char *PARSE_API = "parseapi.back4app.com";
const char *PARSE_APP = "ici_parse_app_ref";
const char *PARSE_CLASS = "Data";
const char *PARSE_KEY = "ici_parse_app_key";



int i= 0;
float Irms=0,Ptot=0,Prms=0,interval=0,E=0;
String heur,date_d;
//initiation de la librairie emonlib pour le calcul des données
EnergyMonitor emon1;
// nbr de minute: interval d'envoie
int N=5;
// minute actuel
int minutes;
//addresse mac du Yun
String macAdr;

void setup()
{
  // une periode d'attente; delay pour permetre à la parti linux de
stabilisé
  delay(50000);
  /// ouvrir le bridge entre arduino et linux
  Bridge.begin();
  // initialiser le serial pour affichage de donnée
  Serial.begin(9600);
  Serial.println("begin");
  // calibrage de la librairie
  emon1.current(3,2);
  //obtention de l'adress MAC
  macAdr=macAdress();
 // Serial.print("Mac= ");
 // Serial.println(macAdr);

}

////////////////////////// LOOP //////////////////
void loop()
{

      int minutes = minut() ; // obtention de minute actuel
      Irms = emon1.calcIrms(1480);             //////////READ I USE THE
LIBRARY EMONLIB//////////
```

```
      Prms=Irms*230.0; // Calcul de P en se basant sur I (lorsque M.Lotfi
apporte la nouvelle carte nous allons changer cette ligne.)
      Ptot+=Prms;                          //////////CUMULATE P TO FINALLY
CALCULATE ENERGY //////////
      i++;           //nbre d'itération
     /* Serial.print("data = ");
      String s
=String(Irms)+","+String(Prms)+","+String(Ptot)+","+String(E)+","+macAdr;*/
//       Serial.println(s);
     // Serial.println(i);
      if (minutes!=last &&(minutes%N==0)) // si le temps n'est pas egale à
la dernière time step d'envoie et l'interval a passé
       {
       ////////////////      E =SUMME(DALTA T(h) * P(W))/i=DALTA T(h)
SUMME(P(W))/i      /////////
         interval=(N*60)/i;     //interval= delta t (s)
         E = (Ptot*interval)/3600;  ////// s= h/3600
         Process  isstreamer1;
         isstreamer1.runShellCommand("curl
http://www.arduino.cc/asciilogo.txt");  // HTTP REQUEST pour tester la
connection
         if(isstreamer1.available())
         {
            isstreamer1.close();
           // Serial.println("Internet Connection Available");
            ////////////////////// IF THERE IS CONNECTION SEND LIVE
DATA//////////////////////
            request(Irms, Prms, Ptot,E,macAdr); // envoi de données
(I/P/MAC/DATE/TEMPS/E/Ptot)
            ////////////////////////// OPEN THE FILE datalogtest.tx AND SEND
FROM IT ////////
            sdcardsend();   //Envoyer s'il y a des données dans SD card

          }
        else{
           sdappend(); // pas de connection alors append dans sd card
          }
        i=0;
        Ptot=0;
        E=0;
        last=minutes;
        interval=0;

      }
     delay( 5000);
}
////////////////////////////// EXTRACT THE MINUTS FROM THE TIME
/////////////////////////////////
int minut()
{
     Process date;
     if (!date.running()){
        date.begin("date");
        date.addParameter("+%T,%D");
        date.run();
      }
      ////////////////////////////EXTRACT THE MINUTES///////////////
       String timeString = date.readString();
       int place = timeString.indexOf(",");
       date_d=timeString.substring(place+1,timeString.indexOf("\n"));
       timeString=timeString.substring(0, place);
```

```
      // Serial.println( date_d);
      // Serial.println( timeString);
       place = timeString.indexOf(":");
       int secondColon= timeString.lastIndexOf(":");
       String minString = timeString.substring(place+1, secondColon);
       heur =timeString.substring(0, secondColon);
       return minString.toInt();
  }
//////////////////////////// APPEND THE DATA TO THE SD CARD
///////////////////////////////


void sdappend(){
    // Serial.println("No internet connection");
     File dataFile = FileSystem.open("/mnt/sd/datalogtest.txt",
FILE_APPEND);
     if (dataFile) {
        String pp =generate_ine(Irms,Prms,Ptot,E);
        dataFile.println(pp);
        dataFile.close();
    // print to the serial port too:
       // Serial.print("Data Added to the file=");
       // Serial.println(pp);
       }

  }


////////////////////// SEND THE DATA TO BACK4APP
///////////////////////////////////
void request(float i,float j,float k, float l,String mac )
{
    Process process;
    char buf[100];
    process.begin( "curl" );
    process.addParameter( "-k" );
    process.addParameter( "-v" );
    process.addParameter( "-X" );
    process.addParameter( "POST" );
    process.addParameter( "-H" );
    process.addParameter(
    "Content-Type:application/json"
    );

  // Parse.com application
    process.addParameter( "-H" );
    sprintf(
    buf,
    "X-Parse-Application-Id: %s",
    PARSE_APP
    );
    process.addParameter( buf );

  // Parse.com key
    process.addParameter( "-H" );
    sprintf(
    buf,
    "X-Parse-REST-API-Key: %s",
    PARSE_KEY
    );
```

```
    process.addParameter( buf );

  // JSON body
    process.addParameter( "-d" );

   // char*  pp= generat(i,j,k,l,mac);
   // Serial.println(pp);

    //Serial.println(String(pp).length());
    process.addParameter( generat(i,j,k,l,mac));
    Serial.print("freeMemory()=");
    Serial.println(freeMemory());
  // URI
    sprintf(
    buf,
    "https://%s/classes/%s",
    PARSE_API,
    PARSE_CLASS
    );
    process.addParameter( buf );

  // Run the command
  // Synchronous
    process.run();

     char c;

  // While there is data to read
  while( process.available() )
  {
    // Get character
    c = process.read();
    Serial.print(c);
  }
process.close();
  Serial.println("  ");
}

////////////////////////////// READ THE DATA FROM THE SD CARD AND SEND IT TO
THE SERVER ///////////////////////
void sdcardsend(){
  File myFile = FileSystem.open("/mnt/sd/datalogtest.txt", FILE_READ);
                    if (myFile ) {
                           while (myFile.available()) {
                            /////////// READ UNTIL THE END OF LINE
//////////////////////
                           String line = myFile.readStringUntil('\n');
                          // Serial.print("Line= ");
                          /// Serial.println(line);
                            ////////////////////// CREATE THE REQUEST TO
Backendless AND RUN IT USING LINE COMMAND ////////////////////

                            ////////////////////// CREATE THE REQUEST TO
Backendless AND RUN IT USING LINE COMMAND ////////////////////
                          int firstColon = line.indexOf(",");
                          int secondColon= line.indexOf(",",firstColon+1);
                          int theerdColon=line.lastIndexOf(",");

                          Irms=line.substring(0,firstColon).toFloat();

Prms=line.substring(firstColon+1,secondColon).toFloat();
```

```
Ptot=line.substring(secondColon+1,theerdColon).toFloat();
                          E=line.substring(theerdColon+1).toFloat();

                          request(Irms, Prms, Ptot,E,macAdr);

                            }
                    myFile.close();
                    FileSystem.remove("/mnt/sd/datalogtest.txt");
                    }


  }
/////////////////GENERATE THE DATA ON JSON FORMAT TO BE SEND TO THE SERVER
BY REST ///////////////////
char* generat( float Irms,float realPower,float supplyVoltage, float
energykwh,String macAdr) {
  StaticJsonBuffer<80> jsonBuffer;
  JsonObject& root = jsonBuffer.createObject();
  root["Irms"]     = Irms;
  root["Prms"]     = realPower;
  root["Ptot"]     = supplyVoltage;
  root["E"]    = energykwh;

  int i=heur.length()+1;
  char c[i];
  heur.toCharArray(c,i);
  root["H"]    = c;

  int ii=macAdr.length()+1;
  char cc[ii];
  macAdr.toCharArray(cc,ii);
  root["MAC"]    = cc;


  int iii=date_d.length()+1;
  char ccc[iii];
  date_d.toCharArray(ccc,iii);
  root["D"]    = ccc;


  char buffer[160];
  root.printTo(buffer, sizeof(buffer));

  Serial.println(buffer);

  return buffer;
  }
////////////////////// GENERATE A LINE OF DATA TO BE STORED TO THE FILE
WHILE INTERNET CONNECTION INTERREPTION ///////////////////
String generate_ine(float Irms,float realPower,float supplyVoltage, float
energykwh){
  String line;

line=String(Irms)+","+String(realPower)+","+String(supplyVoltage)+","+Strin
g(energykwh);
  return line;
  }
////////////////////////// FIND THE YUN MAC ADRESS
//////////////////////////////
  String macAdress(){
```

```
    Process wifiCheck;  // initialize a new process
    String Mac="";
    wifiCheck.runShellCommand("/usr/bin/pretty-wifi-info.lua");
    while (wifiCheck.available() > 0) {
    String c  = wifiCheck.readStringUntil('\n');
         if (c.startsWith("MAC address")){
              int ind = c.lastIndexOf(" ");
              Serial.println(c);
              Mac = c.substring(ind+1);

             /* ind = Mac.indexOf(":");
              Mac = Mac.substring(ind+1);
              ind = Mac.indexOf(":");
              Mac = Mac.substring(ind+1);
              ind = Mac.indexOf(":");
              Mac = Mac.substring(ind+1);*/

              return Mac;
           }

    }
  return Mac;
}
```