5- Travail réalisé :

5.1 Etude Théorique

5.1.1 Les cellules photovoltaïques :

L'énergie solaire photovoltaïque provient de la transformation directe d'une partie du rayonnement solaire en énergie électrique. Cette conversion d'énergie s'effectue par le biais d'une cellule dite photovoltaïque basée sue un phénomène physique appelé effet photovoltaïque qui consiste à produire une force électromotrice lorsque la surface de cette cellule est exposée à la lumière.

La cellule photovoltaïque élémentaire constitue un générateur de très faible puissance vis-à-vis des besoins de la plupart des applications domestiques ou industrielles. Une cellule élémentaire de quelques dizaines de centimètres carrés délivre, au maximum, quelques watts sous une tension inférieure à un volt.

Pour produire plus de puissance, plusieurs cellules doivent être assemblées afin de créer un module ou un champ photovoltaïque. La connexion en série des cellules permet d'augmenter facilement la tension de l'ensemble, tandis que la mise en parallèle permet d'accroitre le courant.

Le câblage série/parallèle est donc utilisé pour obtenir globalement un générateur PV aux caractéristiques souhaitées.

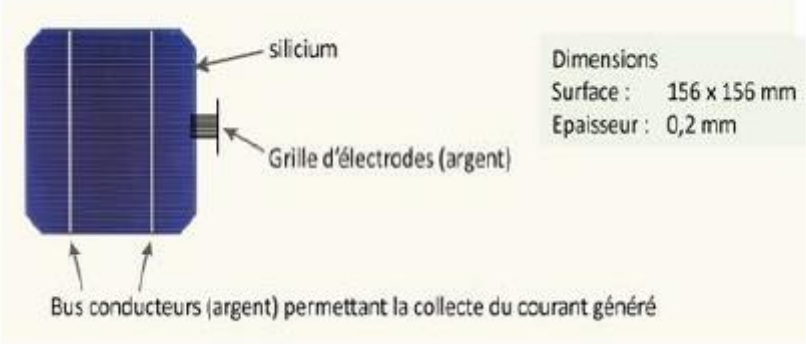

Figure 2 : Schéma d'une cellule photovoltaïque

Le composant le plus crucial de toute installation PV est le module photovoltaïque, qui se compose de cellules solaires interconnectées. Ces modules sont raccordés entre eux pour former des champs de manière à pouvoir satisfaire différents niveaux de besoins en énergie.

Figure 3 : Module photovoltaïque

5.1.2 Etude des capteurs :

Suite aux différentes études et recherches réalisés pour choisir le bon capteur permettant de mesurer la valeur de courant issu des panneaux, on a choisi le dispositif ACS712 Allegro. Ce capteur offre un moyen économique et précis de détection de courants AC et DC. Ce capteur de courant Allegro ACS712 est basée sur le principe de l'effet Hall, qui a été découvert par le Dr Edwin Hall en 1879 selon ce principe, quand un conducteur de courant est placé dans un champ magnétique déposé, une tension est générée sur ses bords perpendiculaires à la direction à la fois du courant et du champ magnétique.

Figure 4 : Capteur de courant ACS712

La sortie du dispositif a une pente positive lorsqu'un courant augmentant circule à travers le chemin de conduction de cuivre. Le ACS712-30B peut mesurer le courant jusqu'à ± 30A et fournit la sensibilité de sortie de 66 mV/A (à +5V), qui signifie que pour chaque augmentation de 1A dans le courant à travers les bornes de conduction dans le sens positif, la tension de sortie augmente aussi par 66 mV. Au zéro de courant, la tension de sortie est la moitié de la tension d'alimentation (Vcc / 2). Il est à noter que la sortie fournit ACS712 ratio métrique, ce qui signifie que le courant de sortie de zéro et la sensibilité de l'appareil sont à la fois

proportionnel à la tension d'alimentation VCC. Cette fonctionnalité est particulièrement utile pour l'utilisation de la ACS712 avec un convertisseur analogique-numérique [3].

Simulation du capteur du courant ACS712 sous Proteus :

La figure (3.9) représente le schéma du capteur du courant ACS712 sous Proteus. On a effectué une simulation sous Proteus pour mesurer un courant continu. Cette simulation montre la sensibilité de ce capteur pour mesurer le courant.

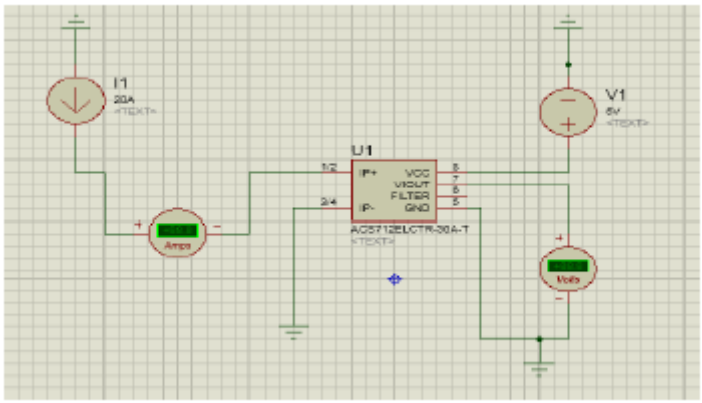

Figure 4 : Schéma du capteur du courant ACS712 sous Proteus

5.1.3 Etude de la carte Arduino UNO

Le système Arduino est un outil pour fabriquer de petits ordinateurs qui peuvent contrôler d'avantage les choses du monde matériel que votre ordinateur de bureau. C'est une plateforme open source d'électronique programmée qui est basée sur une carte à microcontrôleur et un logiciel, véritable environnement de développement intégré, pour écrire, compiler et transférer le programme vers la carte. Parmi toutes les cartes de la famille Arduino, nous avons choisi la version UNO vu ces caractéristiques et c'est à partir de cette base que nous avons développé toute la suite de ce projet

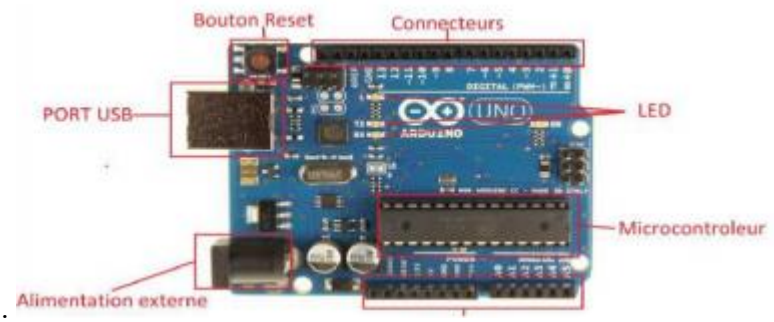

Figure 5 : Carte Arduino UNO

La carte Arduino Uno comme illustrée dans la figure ci-dessus est une carte à microcontrôleur basée sur l'ATmega328.

Elle dispose de :

- 14 broches numériques d'entrées/sorties (dont 6 peuvent être utilisées en sorties PWM (largeur d'impulsion modulée)
- 6 entrées analogiques (qui peuvent également être utilisées en broches entrées/sorties numériques),
- Un quartz 16Mhz,
- Une connexion USB,
- Un connecteur d'alimentation jack,
- Un connecteur ICSP (programmation "in-circuit"),
- Un bouton de réinitialisation (reset). Elle contient tout ce qui est nécessaire pour le fonctionnement du microcontrôleur ; pour pouvoir l'utiliser et se lancer, il suffit simplement de la connecter à un ordinateur à l'aide d'un câble.

La synthèse des caractéristiques :

Le tableau ci-dessous représente les caractéristiques de la carte Arduino utilisé :

Microcontrôleur	ATmega328
Tension de fonctionnement	5V
Tension d'alimentation (recommandée)	7-12V
Tension d'alimentation (limite)	6-20v
Broches E/S numériques	14 (dont 6 disposent d'une sortie PWM)
Broches d'entrée analogiques	6 (utilisables en broches E/S numériques)
Mémoire programme flash	32 KB (ATmega328)
Mémoire SRAM (mémoire volatile)	2 KB (ATmega328)
Mémoire EEPROM (non volatile)	1 KB (ATmega328)
Vitesse d'horloge	16 MHz

Tableau1 : Caractéristiques de la carte Arduino

L'alimentation de la carte Arduino :

La carte Arduino Uno peut être alimentée soit via la connexion USB (qui fournit 5V sous 500mA) ou à l'aide d'une alimentation externe. La source d'alimentation est sélectionnée automatiquement par la carte. L'alimentation externe (non-USB) peut être soit un adapteur secteur (pouvant fournir typiquement de 3V à 12V sous 500mA) ou des piles (ou des accus). L'adaptateur secteur peut être connecté en branchant une prise 2.1mm positive au centre dans le connecteur jack de la carte. Les fils en provenance d'un bloc de piles ou d'accus peuvent être insérés dans les connecteurs des broches de la carte appelées Gnd (masse ou 0V) et Vin (Tension positive en entrée) du connecteur d'alimentation. La carte peut fonctionner avec une alimentation externe de 6 à 20 volts. Cependant, si la carte est alimentée avec moins de 7V, la broche 5V pourrait fournir moins de 5V et la carte pourrait être instable. Si on utilise plus de 12V, le régulateur de tension de la carte pourrait chauffer et endommager la carte. Aussi, la plage idéale recommandée pour alimenter la carte Uno est entre 7V et 12V.

5.2 Réalisation :

Au cours de ce stage, j'étais chargé de faire l'étude et la réalisation d'un réseau de capteur sans fil dans une station photovoltaïque.

Un objet connecté :

Le système complet d'un objet connecté intègre quatre composants distincts :

- Des capteurs : ces capteurs nous permettrons de suivre l'énergie des panneaux ainsi que d'autres performances. On a choisi de commencer avec un capteur de courant pour indiquer les mesures en temps réel de la valeur de courant DC produit par les cellules photovoltaïques.
- Une connectivité : vu que les stations photovoltaïques sont localisées dans des lieux distants de l'utilisateur et isolés, on s'est trouvé obligé de se référer à une connexion 3g via un module GSM monté avec la carte Arduino. En effet, la communication 3g est une communication ayant une très longue portée et une forte puissance. Encore, les réseaux GSM utilisent des puissances d'émission de l'ordre de 1W à 2W selon les gammes de fréquence.Soit 10x plus que le Wifi et 40x plus que les LPWAN (Low-Power Wide-Area Network) et 300x plus que le BLE(Bluetooth Low Energy) .Aussi , la couverture en 3G est presque universelle. Mais La connexion demande beaucoup d'échanges avant de pouvoir émettre.

- Un traitement de l'information : Le traitement de l'information sera fait sur le site internet **https://thingspeak.com**
- Une interface utilisateur : Pour ce projet, l'interface utilisateur est le smartphone via une application Android et le pc via un programme LabVIEW.

La démarche de travail sera détaillée ci-dessous. En effet, dans le but d'accomplir cette mission, j'ai passé par les étapes suivantes :

1) Réception des données par la carte Arduino UNO :

Afin de lire les données issues d'un capteur d'humidité en temps réel via la carte Arduino Uno, on a réalisé le montage suivant :

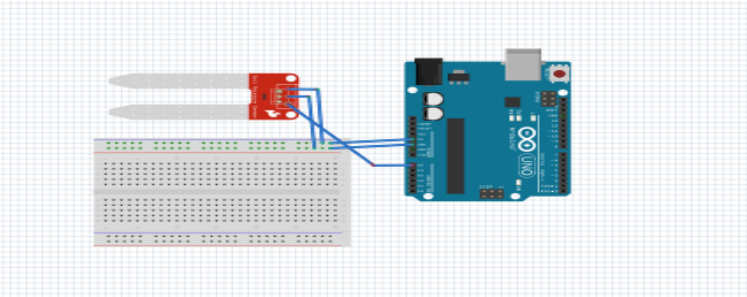

Figure 6 : Simulation de cicuit

Ensuite, on a développé un simple code sur Arduino IdE qui lit les données en temps réel.

```
#include <SoftwareSerial.h>


void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
pinMode (6,INPUT);


}

void loop() {
  // put your main code here, to run repeatedly:
Serial.print (analogRead (6));
Serial.print("\n");
delay(2000);
}
```

Figure 7 : Code sous Arduino

2) Création d'un compte Thingspeak :

Les appareils connectés enregistrent aujourd'hui des dizaines de données chaque minute. Ce sont des données qui concernent autant son usage que le suivi de son bon fonctionnement. Il faut donc un espace de stockage pour cette base de données. Là encore, il est plus simple et plus évident

de stocker les données récoltées à l'extérieure de l'objet, un endroit où elles résident en sécurité accessible à tout instant. Ainsi, la réception des données à distance entre un microcontrôleur et une interface utilisateur que ce soit un pc ou un smartphone nécessite bien évidemment un espace jouant le rôle d'intermédiaire permettant L'hébergement et le stockage des données. Il s'agit d'un espace Cloud : [1]

> Le Cloud computing ou informatique en nuage est une infrastructure dans laquelle la puissance de calcul et le stockage sont gérés par des serveurs distants auxquels les usagers se connectent via une liaison Internet sécurisée. L'ordinateur de bureau ou portable, le téléphone mobile, la tablette tactile et autres objets connectés deviennent des points d'accès pour exécuter des applications ou consulter des données qui sont hébergées sur les serveurs. Le cloud se caractérise également par sa souplesse qui permet aux fournisseurs d'adapter automatiquement la capacité de stockage et la puissance de calcul aux besoins des utilisateurs.

> On a choisi comme serveur cloud la plateforme gratuite pour l'Internet des objets fournit par le site Thingspeak.com

ThingSpeak [2] est une API et une application open source pour « L'Internet des objets », permettant de stocker et de collecter les données des objets connectés en passant par le protocole HTTP via Internet ou un réseau local. Avec **ThingSpeak**, l'utilisateur peut créer des applications d'enregistrement de données capteurs, des applications de suivi d'emplacements et un réseau social pour objets connectés, avec mises à jour de l'état.

Les fonctions assurées par **ThingSpeak** :

- API ouverte
- Collecte de données en temps réel
- Donnés de géolocalisation
- Traitement des données
- Visualisations de données
- Messages d'état des circuits : **ThingSpeak** peut être intégré aux plates-formes Arduino, Raspberry Pi, ioBridge / RealTime.io, Electric Imp, aux applications mobiles/Web, aux réseaux sociaux et aux analyses de données avec MATLAB.

Afin de pouvoir manipuler un projet avec **Thnigspeak** et pouvoir télécharger les données à des fins d'analyse et de traitement, il faut tout d'abord créer un compte sur ce site. Créer ensuite un nouveau canal dont les champs reflètent les données qu'on va télécharger

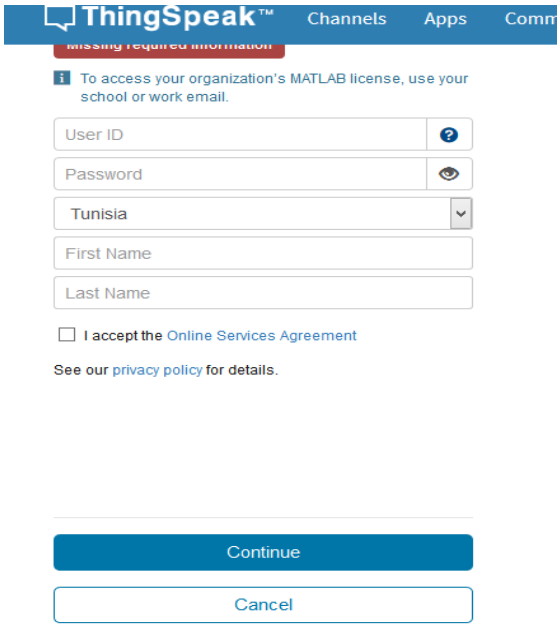

Figure 7 : Création d'un compte Thingspeak

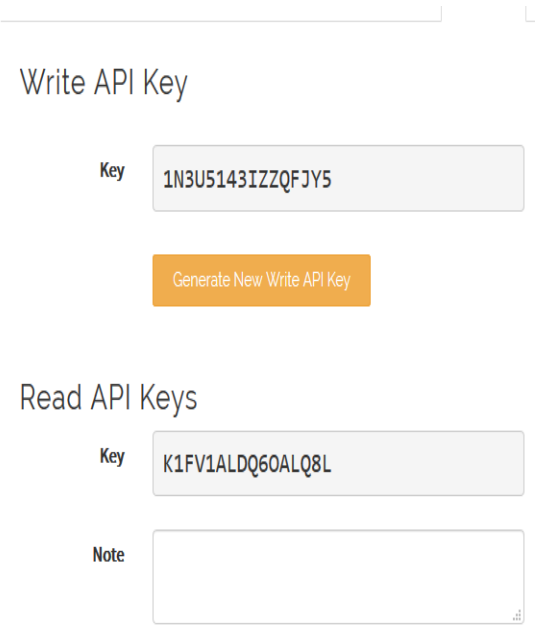

Figure 8 : Clé obtenue

Suite à la création du canal, on obtient une clé API unique correspondante à chaque canal, destinée à garantir le téléchargement des données au bon emplacement et à accéder aux données par l'utilisateur.

3) Envoie des données vers le site Thingspeak :

On a créé un code Arduino qui permet d'envoyer les données vers Cloud via le réseau GSM. Au niveau de la fonction **setup**, on a initialiser les variables nécessaires, tel que APN et le mot de passe du réseau à travers la fonction **gprs.attachGPRS(GPRS_APN, GPRS_LOGIN, GPRS_PASSWORD).** Ensuite, à l'aide de la fonction **client.print("GET ")** les valeurs seront envoyées vers le serveur .

```
Serial.println("Starting Arduino web client.");
// connection state
boolean notConnected = true;

while (notConnected)
{
  boolean k= gsmAccess.begin();
  boolean G= gprs.attachGPRS(GPRS_APN, GPRS_LOGIN, GPRS_PASSWORD);
  Serial.println(k);
   Serial.println(G);
  if ((gsmAccess.begin() == GSM_READY) &
      (gprs.attachGPRS(GPRS_APN, GPRS_LOGIN, GPRS_PASSWORD) == GPRS_READY))
   { notConnected = false;
    Serial.println("connected"); }
  else
  {
    Serial.println("Not connected");
    delay(1000);
  }
}

Serial.println("connecting...");
```

```
if (client.connect(server, port))
{
  Serial.println("connected");
  // Make a HTTP request:
 client.print("GET ");
  client.print(path);
  client.println(" HTTP/1.1");
  client.print("Host: ");
  client.println(server);
  client.println("Connection: close");
  client.println();
}
```

Figure 9 : Envoie des données vers Thingspeak via GSM

Avec ce code, on a rencontré un problème de synchronisation entre le module GSM associé à la carte Arduino et le réseau qu'on dispose. Afin de ne pas se bloquer sur cette étape et pouvoir valider l'étape suivante, on a développé un code MATLAB sur le site Thingspeak permettant de simuler l'envoie de données en temps réel.

MATLAB Code

```
ChannelID = [533096];

writeAPIKey = '1N3U5143IZZQFJY5';

 readAPIKey = 'K1FV1ALDQ6OALQ8L';

r =rand +100;
DCcurrent = round(r,2);
DC = thingSpeakRead(ChannelID,'Fields',1,'NumMinutes',60,'ReadKey',readAPIKey);
DcAvg= round(mean(DC),2) ;

thingSpeakWrite(ChannelID,[DCcurrent,DcAvg],'writekey',writeAPIKey);
```

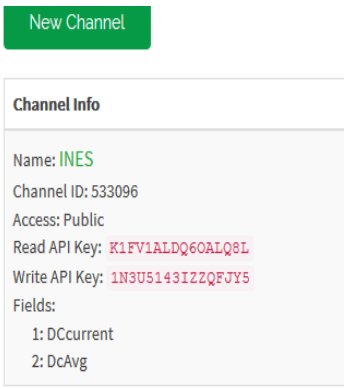

Figure 9 : Code MATLAB

4) Réception des données par LabVIEW et par Android :

- Application Androïde :

Pour le développement de l'application on s'est servi par l'outil **MIT App Inventor** [3].

En effet, App Inventor pour Android est une application développée par Google. Elle est actuellement entretenue par le Massachusetts Institute of Technology (MIT). Elle simplifie le développement des applications sous Android et le rend accessible même pour les novices et ceux qui ne sont pas familiers avec les langages de programmation. Elle est basée sur une interface graphique similaire à Scratch et à celle de StarLogo TNG (en). Grâce à son interface entièrement graphique et à l'absence totale de ligne de code, elle est facile à manipuler pour un débutant avec androïde.

Google publie l'application le 15 décembre 2010 et met fin à son activité le 31 décembre 2011. Dès l'été 2011, Google travaille sur un projet similaire Blockly2, développé cette fois en JavaScript. Depuis le retrait de Google, c'est le centre d'études mobiles au MIT qui gère le support technique de cette application sous le nouveau nom "MIT App Inventor".

Les blocs les plus importants au niveau de notre application sont les suivants :

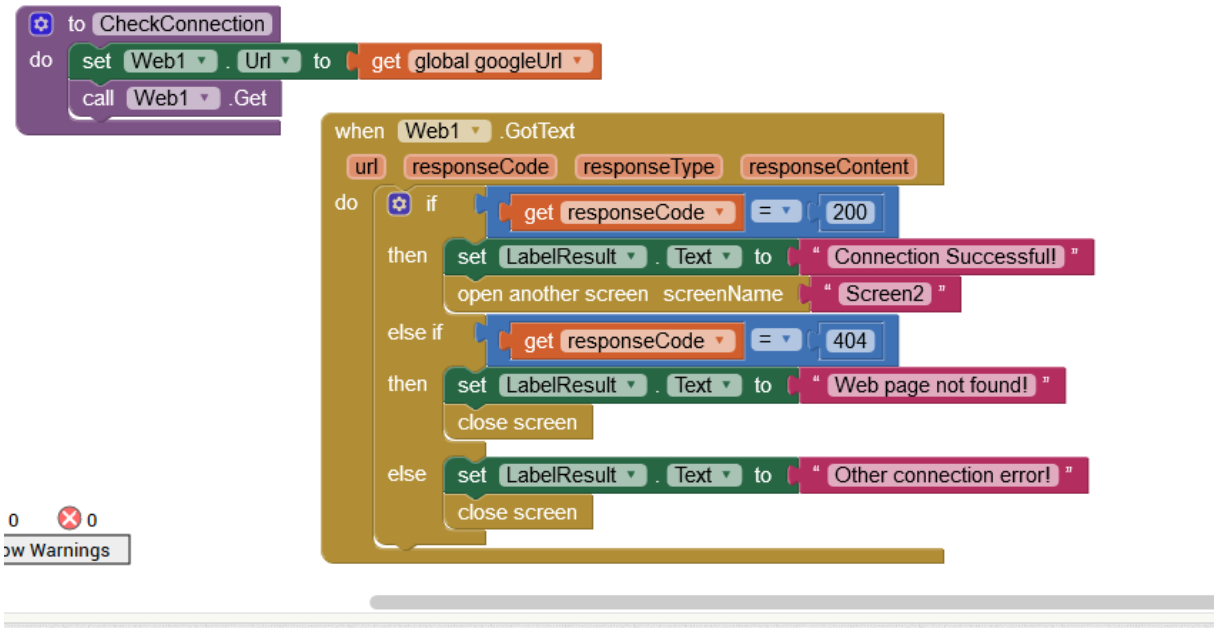

Figure 10 : Bloc de l'écran 1 sur MIT App Inventor

Le bloc ci-dessus permet de vérifier la connexion de notre smartphone à un réseau internet.

Ceci se fait à travers **get responseCode** qui renvoie des valeurs bien déterminé en essayant de se connecter à internet (par exemple la valeur 200 si la connexion est réussie).

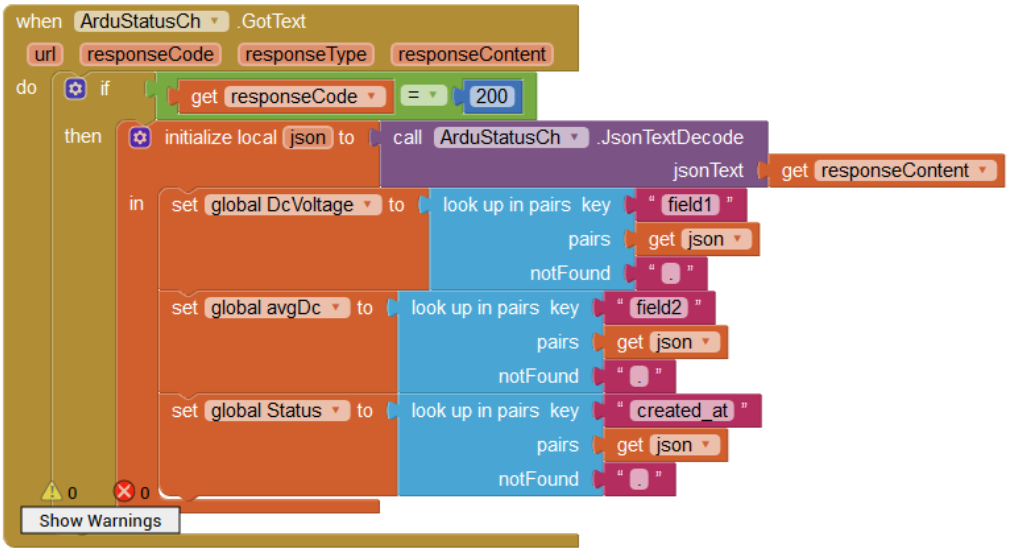

Figure 10 : Bloc de l'écran 2 sur MIT App Inventor

Le bloc ci-dessus permet de recevoir les données envoyées par la plateforme Cloud sous format **json**, puis extraire les données et les mettre dans leurs variables à travers la fonction **look up in pairs.**

Ci-dessous les captures d'écran présentant notre application :

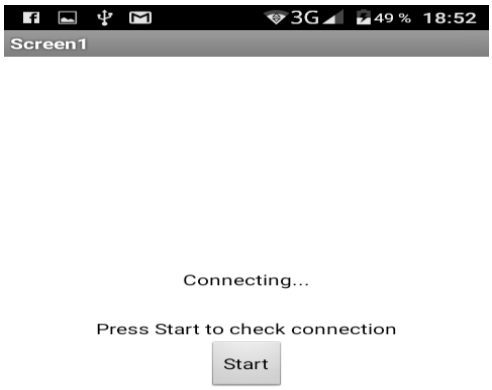

Figure 11 : Capture de l'application

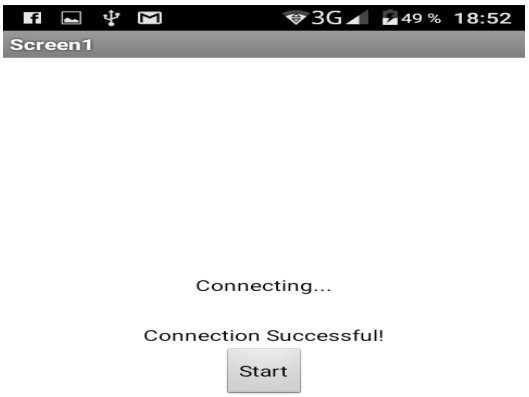

Figure 12 : Cas où la connexion réussi

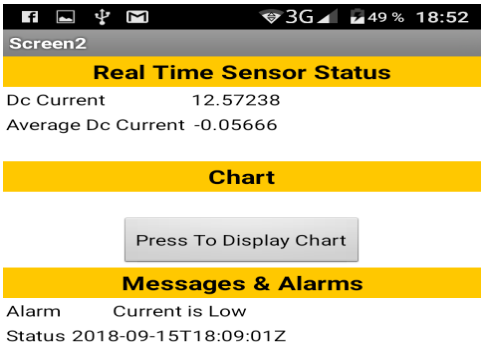

Figure 13 : Capture de la 2éme écran de l'application

Les valeurs de « Dc Current » et « Average Dc Current » Sont des valeurs reçues à partir du Cloud.

Le champ « Status » indique la date du dernier envoie des données.

Application LabVIEW :

LabVIEW [4] (Laboratory Virtual Instrument Engineering WorkBench) est un logiciel de développement de programmes d'application. LabVIEW utilise un langage de programmation essentiellement graphique dédié au contrôle, à l'acquisition, l'analyse et la présentation de données. En LabVIEW, on n'écrit pas de lignes de programme dans un langage textuel comme Pascal ou C, Basic ou Fortran. On manipule des objets graphiques. Ces objets graphiques représentent à la fois les variables du programme, ainsi que des fonctions qui vont réaliser des actions portant sur ces variables. La programmation en LabVIEW consiste simplement à concevoir le traitement de l'information, organiser et relier les variables avec les fonctions au moyen de fils. LabVIEW est dédié à la programmation conçue pour le pilotage d'instruments électronique. Avec LabVIEW on construit graphiquement des modules logiciels appelés des « VI » (« VI » sigle de « Visual Instruments ») au lieu d'écrire du code dans un langage informatique textuel. Son principe de programmation est basé sur l'assemblage graphique de modules logiciels appelés « Instruments Visuels (« VI »). Le rôle d'un VI est d'acquérir des données issues par exemple de fichiers, du clavier ou encore de cartes électroniques d'Entrée/Sorties », de les analyser, et de les présenter au travers d'interfaces hommes-machines graphiques (encore appelées « face avant » par analogie avec la face avant permettant de piloter un appareil électronique).

Au cours de ce stage, on a effectué un programme sous LabVIEW qui lit les données à partir de notre Channel sur Thingspeak et les enregistrer dans une base de données.

Tout d'abord, on a créé une base de données de type tableau sur Microsoft Access. Ensuite, on a créé un fichier **.UDL** pour accéder à la base de données via LabVIEW.

En effet, UDL (Universal Data Link, appelé aussi Microsoft Data Link) est un fichier universel qui établit un lien à une base de données.

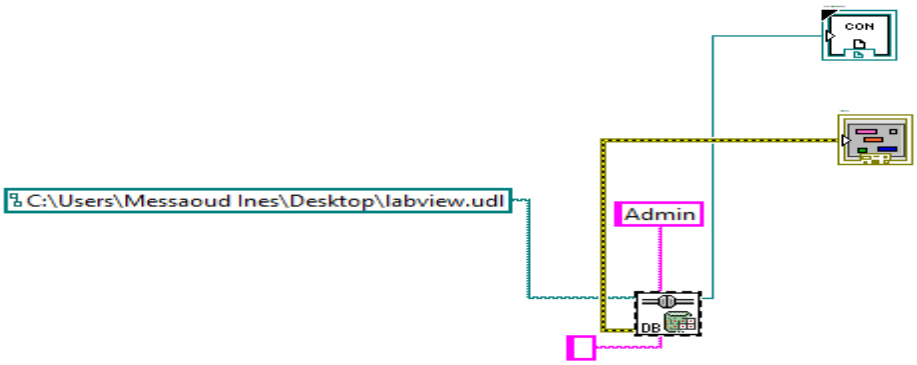

Figure 14 : SubVi OpenDatabase

Les différentes structures et blocs digramme LabVIEW utilisés sont :

- **Structure Séquence déroulée** : Composée d'un ou plusieurs sous-diagrammes ou étapes qui s'exécutent de façon séquentielle. Utilisez la structure Séquence déroulée pour garantir qu'un sous-diagramme s'exécute avant ou après un autre sous-diagramme.

On l'a utilisé dans notre programme pour réinitialiser les contrôles de la face avant.

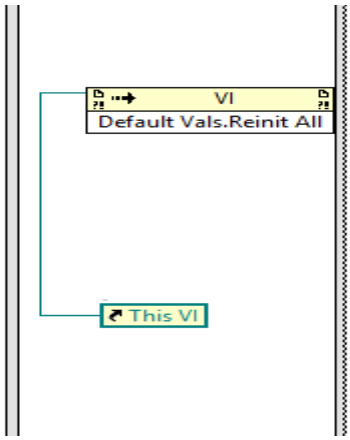

Figure 13 : Réinitialisation des valeurs

- **Event Structure** : Attend qu'un événement se produise puis exécute la condition appropriée pour gérer cet événement.

  A l'aide de cette structure, notre programme LabVIEW attend que l'utilisateur saisie les données de connexion et appuie sur le bouton ok.

- **Open Database** : permet d'établir une connexion à un serveur MySQL fonctionnant sur "host". Ce VI doit s'exécuter correctement avant de pouvoir utiliser une autre fonction du toolkit.

  Les parties essentielles dans notre code LabVIEW sont les suivants :

- Pour pouvoir visualiser les données reçues, on a exigé à l'utilisateur de s'identifier par son nom d'utilisateur et son mot de passe. Les données saisies sont par la suite vérifiées dans une liste des utilisateurs autorisés qui sont enregistrées dans une base de données.

  Ceci par le diagramme suivant :

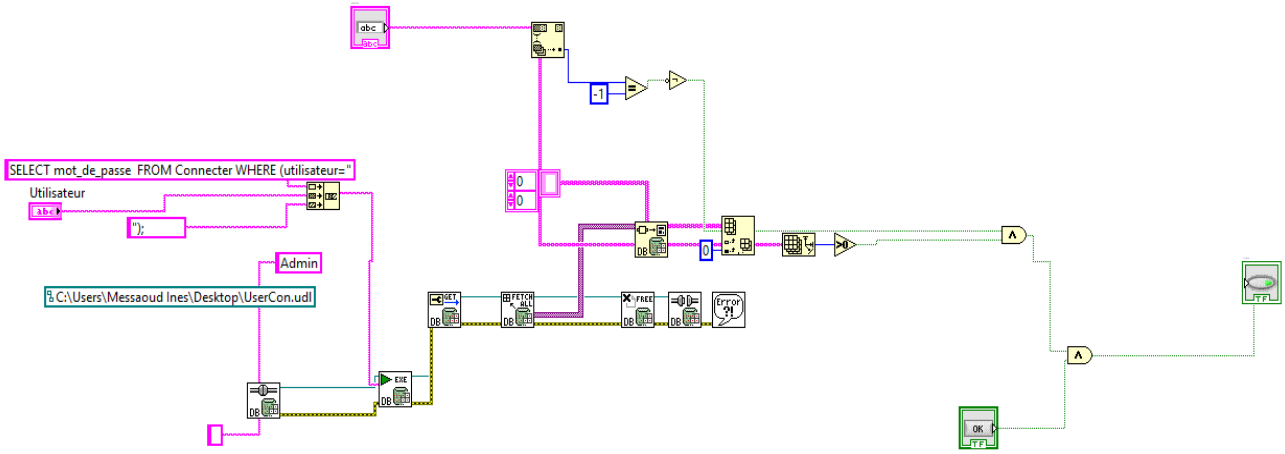

Figure 14 : Vérification des données saisies

❖ A l'aide du bloc **« http client get »** permet de recevoir les données avec le protocole http. Les données envoyées par le site Thingspeak sont sous format json. Donc afin d'extraire les données du corps reçu par protocole http on utilise un cluster associé au bloc « **http unflatten from json** »

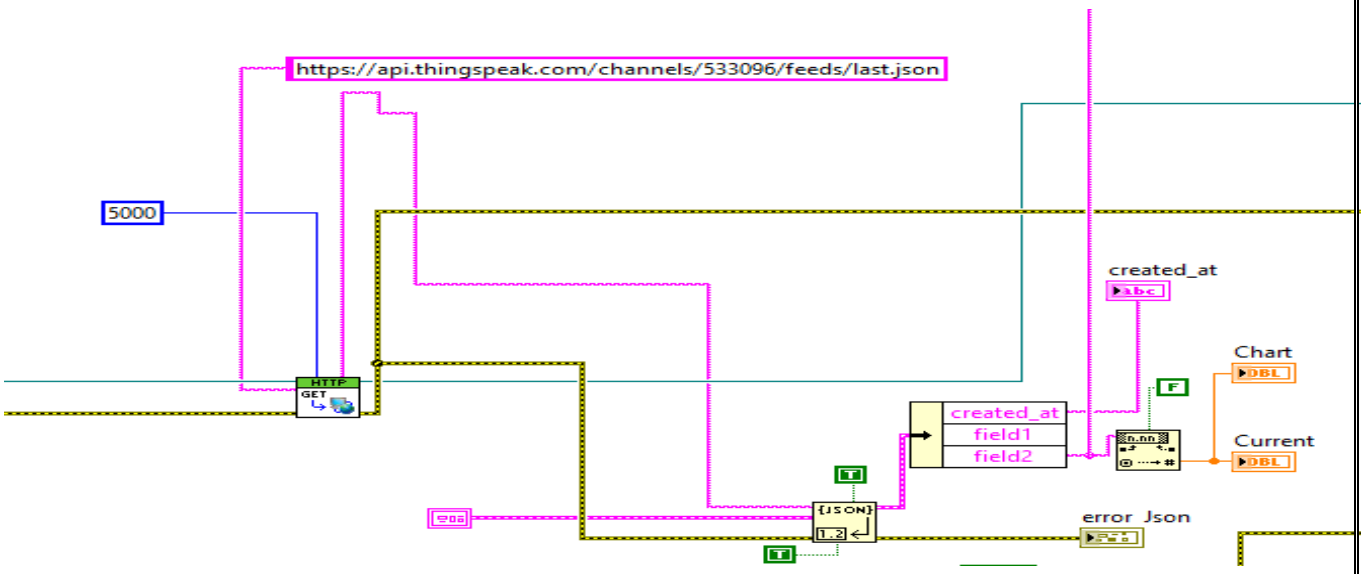

Figure 16 : Réception des données

Après avoir achevé le programme, les captures de la face avant sont les suivantes :

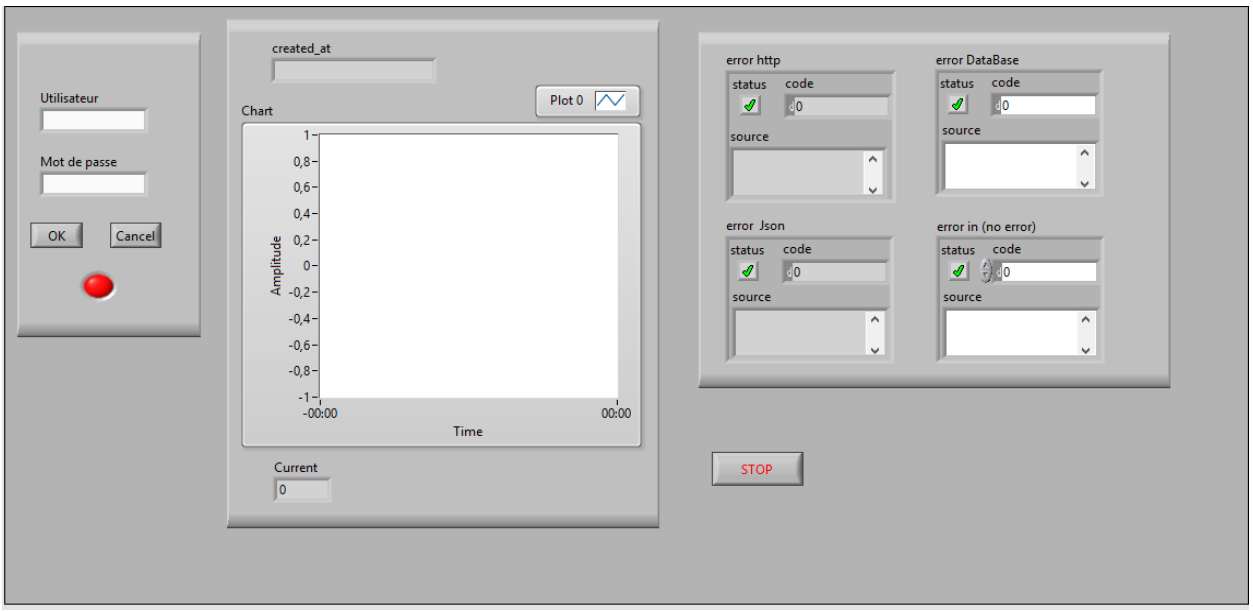

Figure 15 : Face avant initialement

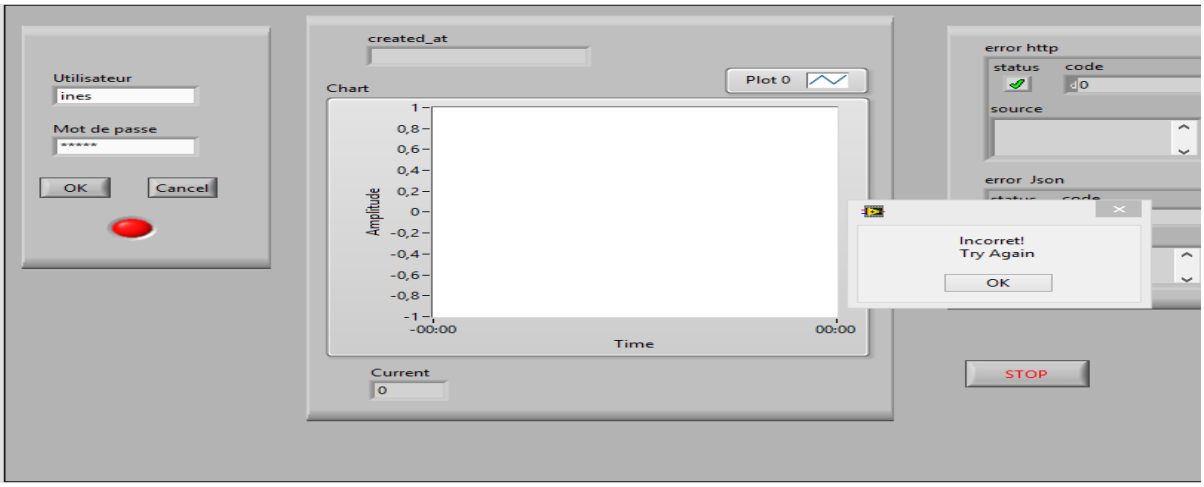

Figure 15 : Cas où les données saisies sont incorrectes