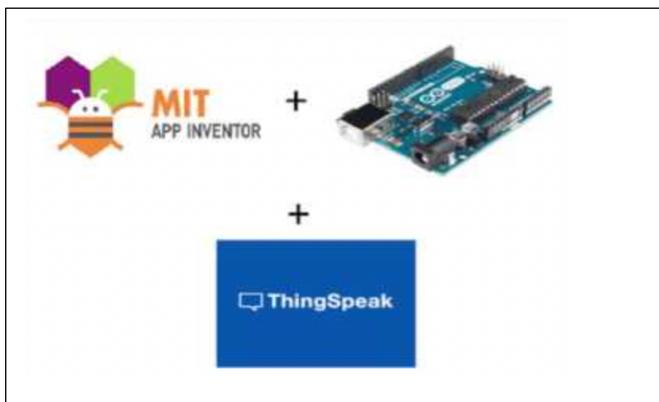


Système de détection de Température (Application Android)



Réaliser par :

Nadia OUACIF

Responsable de cours :

Pr. Ben Ali Cherif

Formation :

M2 Big Data

Université Paris8 Vincennes-Saint-Denis
Objets connectés et données massives

Introduction :

Les objets connectés, veut dire l'ajout d'une connexion internet à un objet physique. Les éléments les plus importants, d'abord les capteurs qui permettent de collecter l'information et de l'analyser (capteurs de mouvement, de sons, de température, d'humidité ...). Ensuite il faut assurer la **connectivité** qui permet d'utiliser internet pour analyser les informations. Pour la connectivité on peut utiliser soit wifi, du réseau 3G et le Bluetooth.

Donc grâce aux objets connectés on peut contrôler nos objets à distance il suffit juste qu'ils seront connectés.

Description du projet :

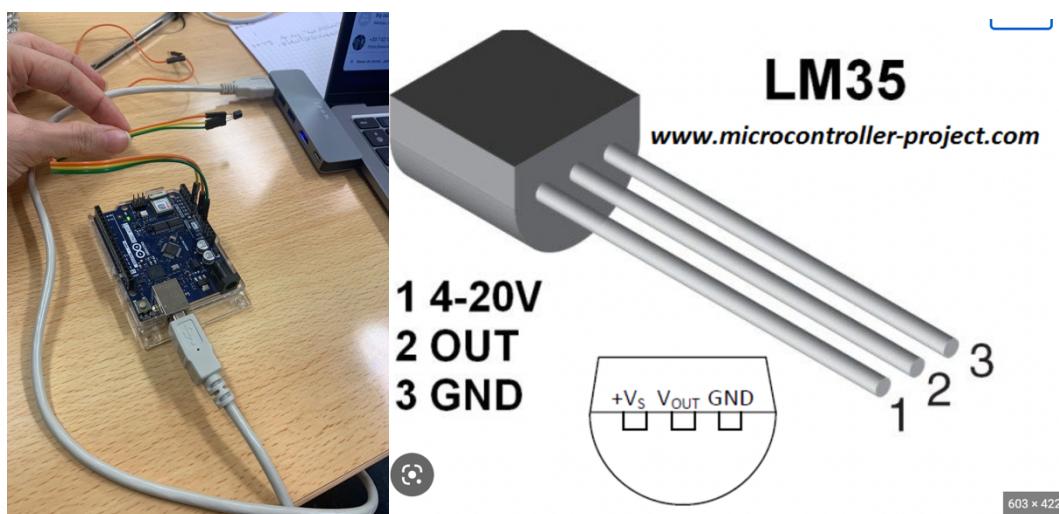
Dans ce projet je vise à créer une application qui mesure la température d'une pièce, et la contrôler à distance. Ce projet est constitué de trois grandes partie :

- La partie 1 : est la partie qui assure la configuration de la partie électronique de l'objet connecté, et je branche les composants électroniques (capteur, la carte Arduino et le PC)
- La partie 2 : est la partie de la récupération de l'information captée par le capteur en utilisant la plateforme **ThingSpeak**
- La partie 3 : est la partie consacrée pour la création de l'application avec App Inventor

Partie 1 : Arduino Wifi Rev2

Outils :

- Arduino IDE 2.00
- Carte Arduino UNO Wifi REV2
- Capteur de température Temperatursensor LM35DZ
- Cable USB Shielded Highspeed Cable 2.0, pour brancher la carte Ardouin à l'ordinateur
- Files de connexion pour connecter le capteur et la carte Arduino



Description de la partie :

Est la partie qui assure la configuration de la partie électronique de l'objet connecté donc c'est la partie de Hardware de ce projet, et je branche les composants électroniques (capteur, la carte Arduino et le PC).

Configuration de Arduino IDE 2.00

→ Connection de la carte Ardouin UNO REV2 à mon PC

→ Installation de la Bibliothèque WiFiNINA à l'aide de la

bibliothèque Manager dans votre IDE Arduino

→ Installez la bibliothèque Arduino "thingspeak" à l'aide de la bibliothèque Manager dans votre IDE Arduino

Le code sur IDE 2.00

- J'appelle les bibliothèques
`#include "ThingSpeak.h"`; `#include <WiFiNINA.h>`
- Je configure le Nom de Wifi et le mot de passe, pour connecter au Wifi
- Je configure la connexion entre Arduino Wifi REV2 et la plateforme ThingSpeak, afin de récupérer les données en temps réel

```
#include "ThingSpeak.h"
#include <WiFiNINA.h>
//#include "secrets.h"
char ssid[] = "AndroidAP041f" ; // your network SSID (name)
char pass[] = "lckt0603" ; // your network password
char keyIndex = 0; // your network key Index number (needed only for WEP)
WiFiClient client;
unsigned long myChannelNumber = 1955494;
const char * myWriteAPIKey = "ONB4QEWWHHNB2VHM";
int channelField = 1;
```

Les résultats obtenus :

```
22.46
Channel update successful.
21.97
Channel update successful.
22.46
Channel update successful.
```

Partie 2 : ThingSpeak

Outils :

- Accéder à La plateforme **ThingSpeak**
<https://thingspeak.com/login?skipSSOCHECK=true>
- Création du compte

La plateforme ThingSpeak

La plateforme **ThingSpeak**, est un service maintenu par la société MathWorks (Matlab), permet la création de canaux d'enregistrement pour sauvegarder ces données collectées dans le Cloud. Ces canaux peuvent être privés, ou publics si l'on souhaite partager ces données avec d'autres utilisateurs. Il contient Des outils de visualisation permettent d'afficher les données collectées. Il propose des outils d'analyse des données collectées, basés sur le savoir-faire de MathWorks déjà développé dans leur outil MATLAB.

Dans cette partie y'a deux étapes primordiales à suivre, la première étape est la création du compte pour récupérer les clés de lecture et d'écriture, la deuxième étape est celle de visualisation des données récupérer en temps réel ;

nadia_ProjetTemperature

Channel ID: 1955494
 Author: mwa0000028447466
 Access: Private

arduinoWIFI MIT

[Private View](#) [Public View](#) [Channel Settings](#) [Sharing](#) [API Keys](#) [Data Import / Export](#)

Write API Key

Key

[Generate New Write API Key](#)

Read API Keys

Key

Note

[Save Note](#) [Delete API Key](#)

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- **Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click [Generate New Write API Key](#).
- **Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click [Generate New Read API Key](#) to generate an additional read key for the channel.
- **Note:** Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

Write a Channel Feed

GET https://api.thingspeak.com/update?api_key=ONB4QEWWHHNB2VHM

Read a Channel Feed

GET https://api.thingspeak.com/channels/1955494/feeds.json?api_key=ONB4QEWWHHNB2VHM

nadia_ProjetTemperature

Channel ID: 1955494
 Author: mwa0000028447466
 Access: Private

arduinoWIFI MIT

[Private View](#) [Public View](#) [Channel Settings](#) [Sharing](#) [API Keys](#) [Data Import / Export](#)

[+ Add Visualizations](#)

[+ Add Widgets](#)

[Export recent data](#)

[MATLAB Analysis](#)

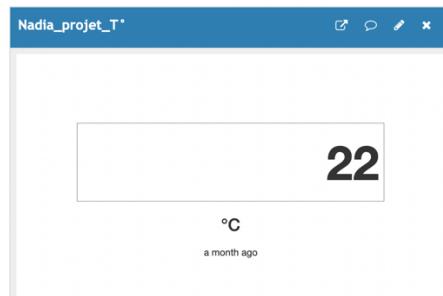
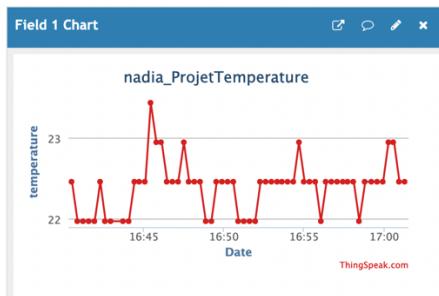
[MATLAB Visualization](#)

Channel Stats

Created: 28 days ago

Last entry: 27 days ago

Entries: 210



Partie 3 : Création d'une application avec App Inventor :

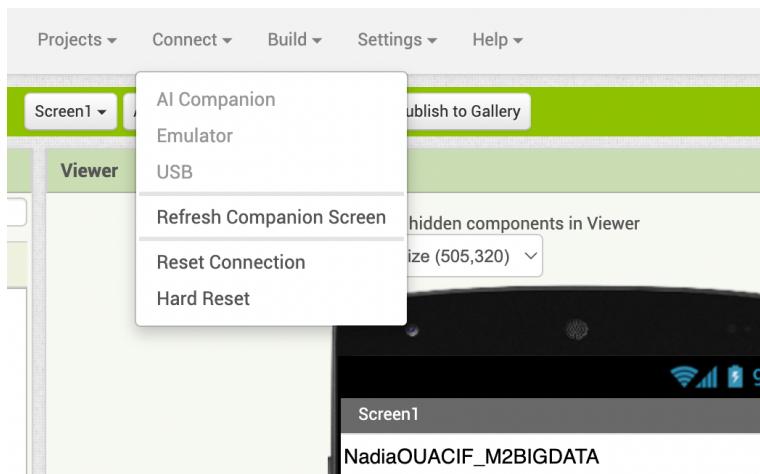
Outils :

L'application est réalisée en ligne sur le site AppInventor et testée grâce au **MIT AI2 Companion** pour Android.

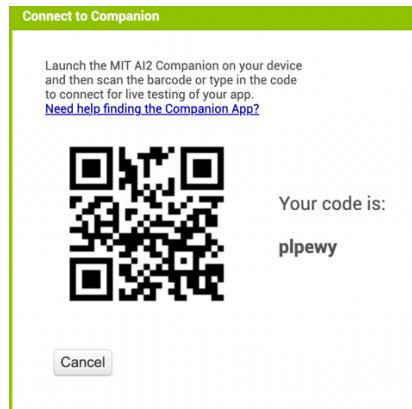
- Accéder à AppInventor : <http://ai2.appinventor.mit.edu/>
- Créer un compte sans passer par une authentification google : <http://code.appinventor.mit.edu>
- Télécharger le MIT AI2 Companion sur Google Play pour tester votre application: <https://play.google.com/store/apps/details?id=edu.mit.appinvantor.ai companion3&hl=fr> (url courte : <https://cutt.ly/ze6tD9G>)
- Scanner le Code QR avec le SmartPhone .

Lancer le compagnon pour tester l'application :

1. Cliquer sur : connecte > compagnon AI :



2. Lancer le Compagnon AI sur le téléphone ou tablette et scanner le qrcode



3. Attendre que le programme soit transféré pour visualiser l'application

Description de l'application :

Dans cette application Android je veux afficher la température de ma pièce ou ma salle capté par un capteur de température en utilisant Arduino (Voir la partie 1).

Pour cela j'ai créé deux boutons à mon application un pour afficher la température, et le deuxième pour effacer l'espace de l'affichage de la température ; le tutoriel suivi et qui m'étais très utile pour comprendre l'utilisation de MIT inventor

https://www.youtube.com/watch?v=gA_3cvfqHpY

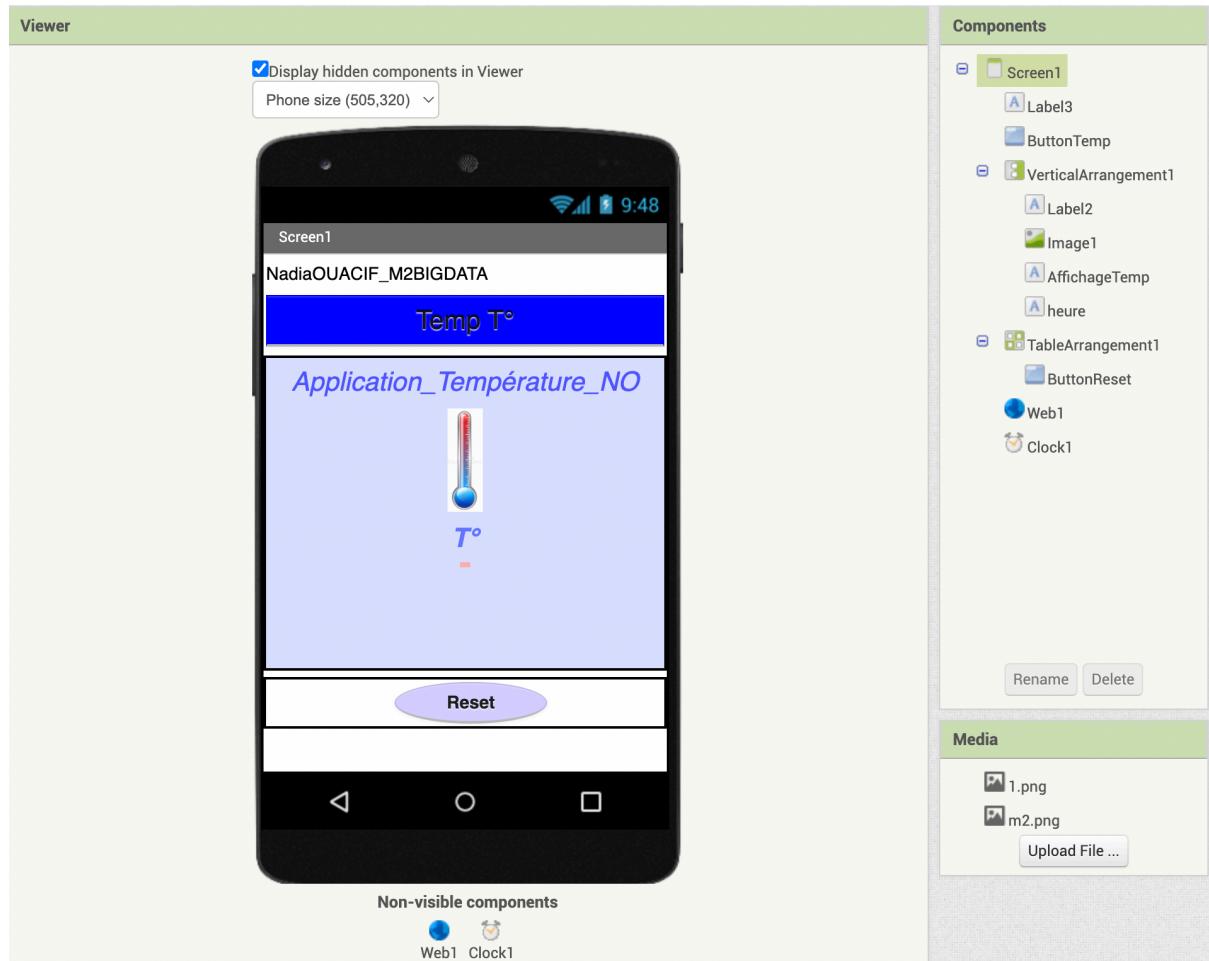
Stratégie :

- Je récupère les coordonnées des températures captées par Arduino (voir partie 1) depuis ThingSpeak (Voir partie 2) ;
- J'affiche la température détecter sur mon portable ;
- J'affiche l'heure actuelle pour que je vérifie le temp.

Affichage :

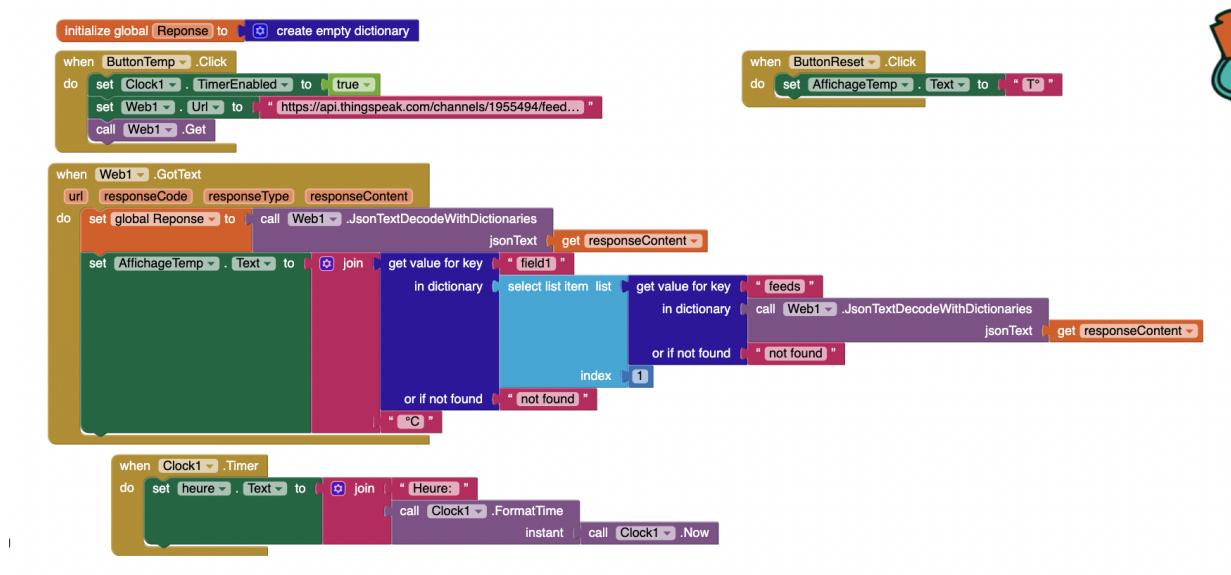
Étape 1 : Designer

Sur l'interface Designer d'appInventor je positionne les éléments que je veux afficher. En utilisant les éléments **Label** pour afficher un texte et **button** pour lancer une action sur le click. **Web** pour récupérer l'information depuis ThingSpeak et **Clock** pour afficher l'heure,



- Label pour afficher mon nom&prénom, formation : renommer le Label3
- Label pour afficher « application_Température » : renommer le Label2
- Label pour afficher la température : renommer AffichageTemp
- Label pour l'Heure (à laquelle a été effectué la mesure) : renommer heure
- Bouton pour lancer la récupération des données : renommer TempT°
- **Module connectivité : web** pour se connecter à l'API et récupérer les données (Température)
- **Module Clock1** : pour afficher l'heure

Étape 2 : Blocks

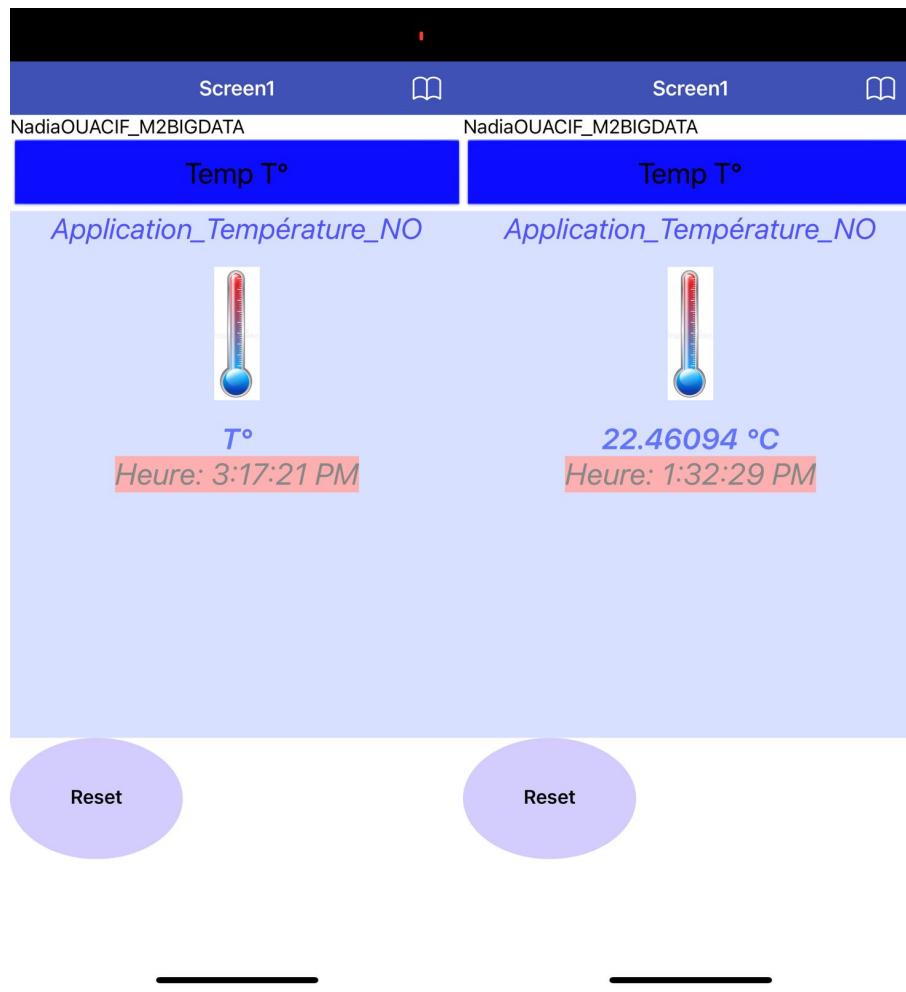


- **Button Temp** : pour qu'il récupère l'heure et la température depuis ThingSpeak :
https://api.thingspeak.com/channels/1955494/feed.json?api_key=VNAQIW0L56ED2GKI&results=2
- **ButtonReset** : permet de nettoyer la zone de texte qui affiche la température ;
- **Module connectivité web** : est le module qui permet de récupérer les informations depuis thingSpeak
- **Module Clock** : il récupère l'heure actuelle qui sera afficher dès l'entrer dans l'application ;

Étape 3 : L'application sur le Smartphone

Pour accéder a l'application MIT inventor depuis mon téléphone c'était un petit peu compliqué au début car j'ai un **Iphone** qui n'est pas

compatible avec les applications Android, et je ne peux pas utiliser un émulateur Android sur mon ordinateur puisque j'ai un **MacBook Pro M1**, donc après plusieurs recherches j'ai trouvé que pour gérer ce genre de conflit et avoir un projet complet il suffit d'utiliser l'application **ApkDownloader** et de télécharger **App Inventor**, en utilisant **ApkDownloader** ;



Annexe :

Dans cette partie je vais mettre le code qui m'a permis de détecter la température d'une pièce, qui seront récupérer via ThingSpeak afin de les afficher sur le SmartPhone ;

sketch_nov25b.ino

```

1  #include "ThingSpeak.h"
2  #include <WiFiNINA.h>
3  //##include "secrets.h"
4  char ssid[] = "Lydia" ; // your network SSID (name)
5  char pass[] = "lydiaaaa" ; // your network password
6  char keyIndex = 0; // your network key Index number (needed only for WEP)
7  WiFiClient client;
8  unsigned long myChannelNumber = 1955494;
9  const char * myWriteAPIKey = "ONB4QEWWHHNB2VHM";
10 //const char * channelField = "VNAQIW0L56ED2GKI";
11 int channelField = 1;
12 int SensorPin = 0;
13 float adcValue;
14 float voltageValue;
15 float temperatureValue = 0;
16 int samplingTime = 15000; // Wait 20 seconds between each channel update
17 void setup() {
18   Serial.begin(9600); // Initialize serial
19   if (WiFi.status() == WL_NO_MODULE) {
20     Serial.println("Communication with WiFi module failed!");
21     // don't continue
22     while (true);
23   }
24   String fv = WiFi.firmwareVersion();
25   if (fv != "1.0.0") {
26     Serial.println("Please upgrade the firmware");
27   }
28   ThingSpeak.begin(client); //Initialize ThingSpeak
29 }

30 void loop() {
31   // Connect or reconnect to WiFi
32   if(WiFi.status() != WL_CONNECTED){
33     Serial.print("Attempting to connect to SSID: ");
34     Serial.println(ssid);
35     while(WiFi.status() != WL_CONNECTED){
36       WiFi.begin(ssid, pass); // Connect to WPA/WPA2 network. Change this line if using open or WEP network
37       Serial.print(".");
38       delay(500);
39     }
40     Serial.println("\nConnected.");
41   }
42   adcValue = analogRead(A0); // Get Data from Temperature Sensor
43   float mv = ( adcValue/1024.0)*5000;
44   float cel = mv/10;
45   Serial.println(cel);
46   // Write to ThingSpeak
47   int x = ThingSpeak.writeField(myChannelNumber, channelField, cel, myWriteAPIKey);
48   if(x == 200){
49     Serial.println("Channel update successful.");
50   }
51   else{
52     Serial.println("Problem updating channel. HTTP error code " + String(x));
53   }
54   delay(15000); // Wait 20 seconds to update the channel again
55 }
```

Conclusion :

A la fin de ce projet, j'ai appris comment utiliser la carte Arduino UNO wifi et la configurer, et aussi j'ai réussi à récupérer ou récolter les informations détectées par la plateforme ThingSpeak.

A la fin, j'ai appris comment créer une application Android en ligne via le site AppInventor et la testée grâce au **MIT AI2 Companion** pour Android.