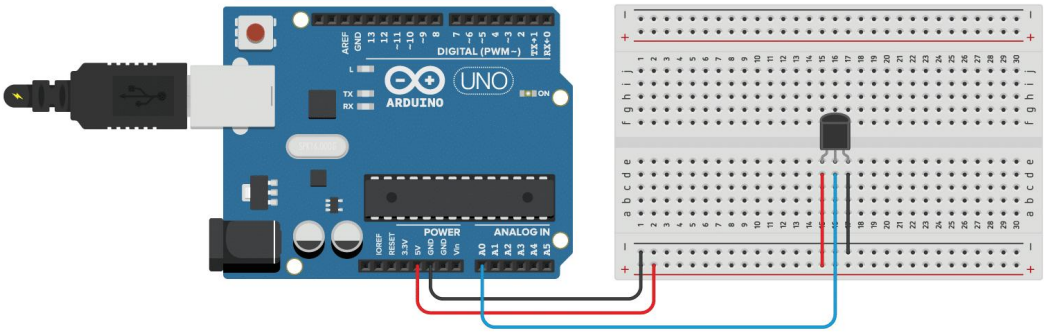
Projet :
Capteur de Température LM35, module ESP-01 et Carte Arduino Uno



Master Informatique : Big Data et Fouille de Données

Enseignants: Monsieur Ali Chérif

Ulrick BLAVIN

UNIVERSITÉ PARIS 8
VINCENNES-SAINT-DENIS

SOMMAIRE

INTRODUCTION :

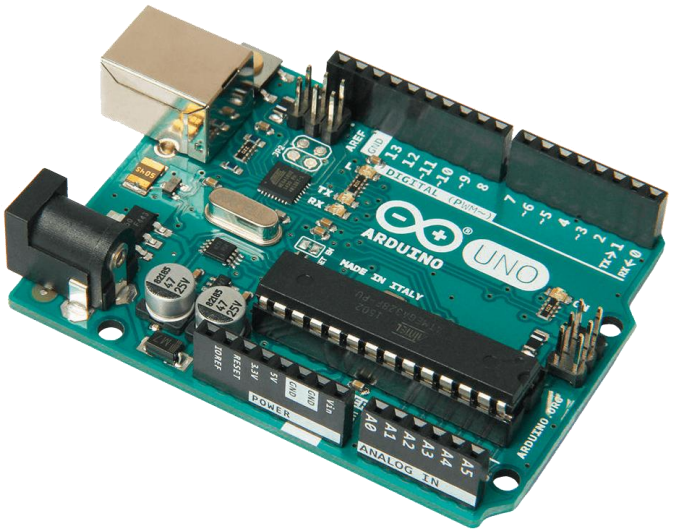
Le LM35 est un capteur de température à haute précision qui peut mesurer la température ambiante avec une précision de ± 0,5 degré Celsius. Il est facile à utiliser avec un microcontrôleur comme l'Arduino Uno, qui est une carte de développement populaire basée sur un microcontrôleur AVR. L'ESP-01 est un module Wi-Fi qui peut être utilisé avec l'Arduino Uno pour permettre à l'appareil de se connecter à un réseau Wi-Fi et de transmettre les données de température à un serveur ou à un autre appareil connecté. En combinant ces trois éléments, vous pouvez créer un système de surveillance de la température qui peut être surveillé à distance.

Le but de ce projet étant de récupérer des données du capteur de températures et de l'afficher sur une application mobile. J'ai exploité deux visions, une avec mon API que j'ai réalisé donc je peux manipuler les données comme je le souhaite et afficher les données sur le web et une autre ou j'utilise une API externe.

lien du projet :

https://drive.google.com/file/d/1cPwti6ijXOR8_C09Dfg0IeU76moTb1A9/view?usp=share_link

ARDUINO UNO :



L'Arduino Uno est une carte de développement électronique basée sur

- un microcontrôleur ATmega328.
- Il a 14 broches d'entrée/sortie numériques,
- 6 entrées analogiques,
- une horloge à quartz de 16 MHz
- une connectivité USB.
- une alimentation externe (6-20V)
- une prise ICSP
- un bouton de réinitialisation

Il est compatible avec une large gamme de capteurs et de périphériques externes, et est utilisé couramment pour les projets de robotique, l'Internet des objets (IoT) et l'électronique de loisir. Il est également compatible avec un grand nombre de logiciels de développement, tels que l'environnement de développement intégré (IDE) Arduino. Le logiciel open-source appelé l'environnement de développement intégré (IDE) Arduino. Il est utilisé pour créer des projets interactifs tels que des capteurs, des afficheurs, des robots et des systèmes automatisés. Il est souvent utilisé par les débutants en électronique et en programmation car il est facile à utiliser et abordable.

Microcontrôleur ATmega328P :

Le microcontrôleur ATmega328P est :

- un microcontrôleur AVR de 8 bits fabriqué par Microchip.
- Il possède 32 Ko de mémoire flash programmable,
- 2 Ko de mémoire RAM,
- 1 Ko de mémoire EEPROM,
- 23 entrées/sorties numériques,
- 6 entrées analogiques,
- 32 registres de données,
- une horloge interne à 1 MHz,
- un module de communication série (UART),
- un module I2C et un module SPI.

Il possède également un mode d'alimentation économiseur d'énergie pour les applications à faible consommation d'énergie. Il est utilisé couramment dans les projets de robotique, l'Internet des objets (IoT) et l'électronique de loisir. Il est compatible avec le logiciel de développement Arduino, ce qui en fait un choix populaire pour les projets de bricolage et de démarrage.

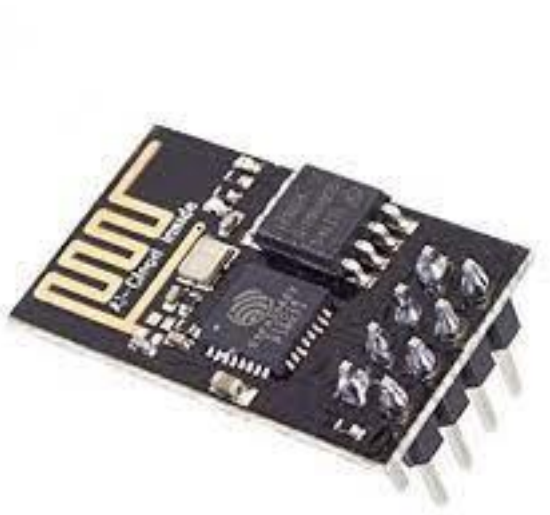
Capteur LM35DZ :

Le capteur LM35 est un capteur de température de précision fabriqué par Texas Instruments. Il est basé sur une diode à jonction à effet de champ qui produit un courant proportionnel à la température.
Il a une précision de ± 0,5 °C à l'intérieur de la plage de température de -55 à 150 °C et une sensibilité de 10 mV/°C. Il peut être utilisé pour mesurer la température dans une plage de -55 à 150°C avec une précision de 0.5°C près. Il est facile à utiliser et ne nécessite pas de compensation de température ou de calibrage. Il est alimenté par une tension de fonctionnement de 4 à 30V et a une sortie linéaire de 10mV/°C. Il est souvent utilisé dans les systèmes de contrôle de température, les systèmes de climatisation, les réfrigérateurs, les congélateurs et les fours.

Il possède des broches :

- GND = Masse
- VCC = Tension d’alimentation de 5V
- OUT = envoie les information

Module WIFI ESP-01 :



Le module WIFI ESP-01 est un module WiFi basé sur le microcontrôleur ESP8266 de Espressif Systems. Il permet de connecter des périphériques électroniques à un réseau WiFi existant. Il a un seul port UART pour la communication série et dispose de 8 broches d'E/S numériques. Il prend en charge la connectivité WiFi 802.11b/g/n et utilise le protocole TCP/IP pour communiquer avec les appareils connectés. Il a une faible consommation d'énergie et peut être alimenté par une tension de 3.3V. Il peut être utilisé pour connecter des projets de domotique, des projets IoT, des projets de capteurs sans fil, et tout autre projet nécessitant une connectivité WiFi. Il est souvent utilisé en conjonction avec un microcontrôleur comme l'Arduino pour permettre la communication réseau.

Matériels :

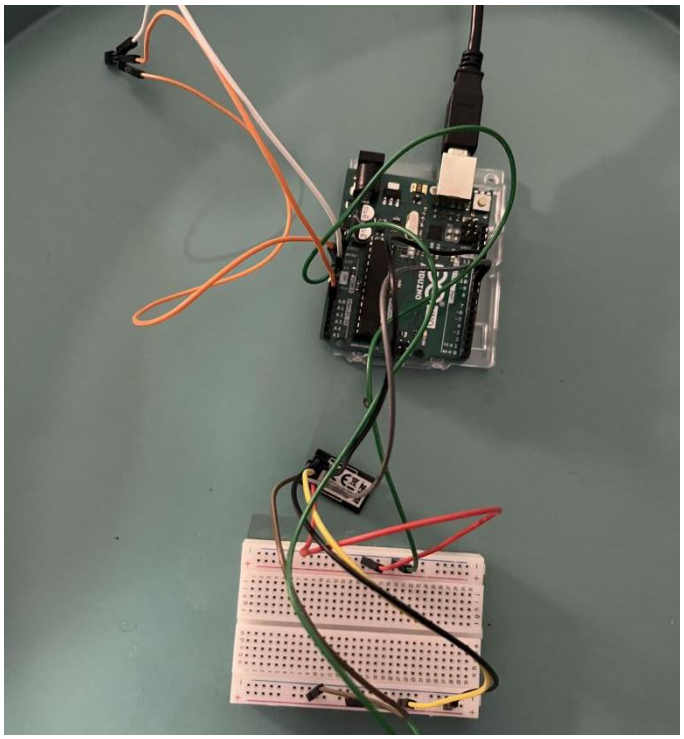
Matériels fournit par Paris 8 :

- Module ESP-01
- Capteur de température LM35DZ
- Carte Arduino Uno
- Quelque fil de liaison.

Matériels supplémentaires :

- Breadboard
- Clé Flash
- D’autre fil de liaison
- D’autre matériels (transitor, condensateur , diode, bouton, résistance, etc...)

Branchement :



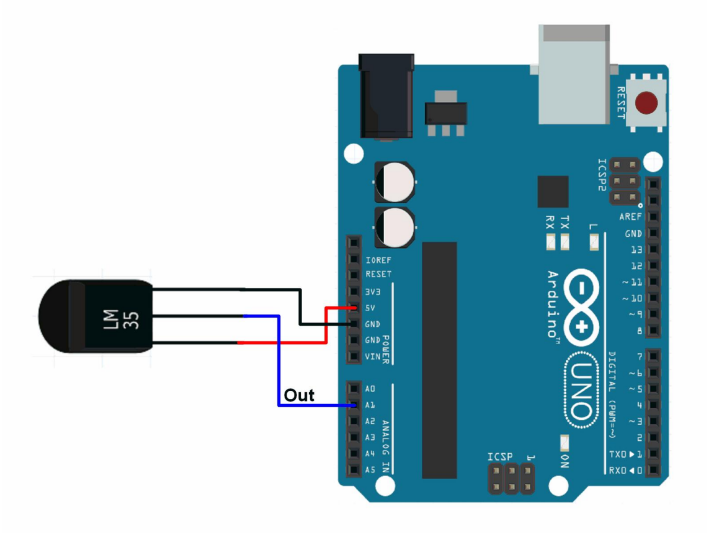
De l’arduino Uno au module ESP-01, nous avons réalisé ces branchements :

- TX de L’ESP vers le RX de l’arduino Uno.
- RX de l’ESP vers le TX de l’arduino Uno.
- GND de l’ESP vers le GND de l’arduino Uno.
- VCC de l’ESP vers le 3.3V de l’arduino Uno.
- RST de l’ESP vers le 3.3V de l’arduino Uno.
- CH-PD de l’ESP vers le 3.V de l’arduino Uno.

Nous avons besoin d’un breadboard parce que nous devons branché plusieurs fils sur le 3.3V hors nous avons qu’une seule broche de 3.3V sur l’arduino.

De l’arduino au capteur de température LM35DZ, nous avons réalisé ces branchements :

- GND du capteur LM35 vers le GND de l’arduino Uno.
- VCC du capteur LM35 vers le 5V de l’arduino Uno.
- OUT du capteur LM35 vers le A0 de l’arduino Uno.



Code arduino avec mon API :

```
#include <SoftwareSerial.h>

SoftwareSerial esp01(10, 11);
const int sensorPin = A0;

// Déclaration du Nom du wifi du point d’accès
String NomduReseauWifi = "Freebox-572C58";

// Déclaration du Mot de passe du point d’accès
String MotDePasse = "adgnasci!5-chald7%-tractabant5.-
litoreas";

// Déclaration de l'API KEY de notre API
String apiKeyValue = "tPmAT5Ab3j7F9";

// Déclaration de la requete
String requete;

//Déclaration de la température
float temp;
float voltage;

void echo(unsigned long attente = 2000ul) {
unsigned long startTime = millis() ;
while (millis() - startTime < attente) {
while (esp01.available()) {
int ch = esp01.read();
Serial.write((byte) ch);
}
}
}

void setup() {

// Initialise la communication avec le PC
```

```
Serial.begin(115200);

Serial.println();

// Initialise la communication avec l'ESP
esp01.begin(9600);

//définir l'ESP en mode AT
esp01.println("AT+CWMODE=1");
echo();

//Test de démarrage
esp01.println("AT");
echo();

//Connexion au point d'accès
esp01.println("AT+CWJAP=\""+ NomduReseauWifi + "\",\"" +
MotDePasse +"\"\r\n");
echo(10000);

//Adresse IP local
/*esp01.println("AT+CIFSR");
echo(10000);*/

//Fermeture de la connexion TCP ou UDP
/*esp01.println("AT+CIPCLOSE");
echo();*/
}

void loop() {

// Mesure la tension sur la broche A0
int valeur_brute = analogRead(sensorPin);

// Transforme la mesure (nombre entier) en température
via un produit en croix
voltage = (valeur_brute / 1024.0) * 5.0;
temp = voltage * 100;


// Conversion de la temperature en string
String data = String(temp);
```

```
// Création de la requete pour envoyer
requete = "GET /Projet_Tempo/insertTemp.php?";

// Ajout de l'API Key dans l'URL
requete += "apiKeyValue=";
requete += apiKeyValue;

// Ajout de la température dans l'URL
requete += "&tempo=";
requete += data;

// Imprime les données sur le port série
Serial.println(requete);

// Attente de 2 secondes
delay(2000);

// La fonction update() est lancé
update();
}

void update(){

// Établir une connexion TCP ou inscrire port UDP,
démarrer la connexion
esp01.println("AT+CIPSTART=\"TCP\",\"192.168.1.54\",80");
echo(5000);
// Envoyer des données
esp01.print("AT+CIPSEND=");
esp01.println(String(requete.length()+2));

//echo();

//Ecrire la requete dans l'ESP
esp01.println(requete);

//echo(5000);
echo(2000);
}
```

Code arduino API ThingSpeak :

À noter qu'il faudra créer un compte sur ThingSpeak au préalable et récupéer la l'APIkey et mettre le nombre de valeurs qu'on veux récupérer et les définirs.

```
#include <SoftwareSerial.h>


const int sensorPin = A0;

// affectation des broches
const byte rxPin = 10;
const byte txPin = 11;

// Configurer un nouvel objet SoftwareSerial
SoftwareSerial esp01(rxPin, txPin);

// Déclaration du Nom du wifi du point d'accès
String NomduReseauWifi = "box";

// Déclaration du Mot de passe du point d'accès
String MotDePasse = "Mdp";

// Déclaration de l'API KEY de notre API
//String apiKeyValue = "tPmAT5Ab3j7F9";
String key= "Q72W93E2L9N58I0H";

// Déclaration de la requete
String requete;

//Déclaration de la température
float temp;
float voltage;

// Fonction qui écris les réponses de l'ESP
void echo(unsigned long attente = 2000ul) {
unsigned long startTime = millis() ;
while (millis() - startTime < attente) {
while (esp01.available()) {
```

```
int ch = esp01.read();
Serial.write((byte) ch);
}
}
}

void setup() {

// Initialise la communication avec le PC
Serial.begin(115200);

Serial.println();

// Initialise la communication avec l'ESP
esp01.begin(9600);

//définir l'ESP en mode AT
esp01.println("AT+CWMODE=1");
echo();

//Test de démarrage
esp01.println("AT");
echo();

//Connexion au point d’accès
esp01.println("AT+CWJAP=\""+ NomduReseauWifi + "\",\"" +
MotDePasse +"\"\r\n");

echo(2000);

}

void loop() {

// Mesure la tension sur la broche A0
int valeur_brute = analogRead(sensorPin);

// Transforme la mesure (nombre entier) en température
via un produit en croix
voltage = (valeur_brute / 1024.0) * 5.0;
temp = voltage * 100;

// Conversion de la temperature en string
```

```
String data = String(temp);

// Création de la requete pour envoyer
//requete = "GET /Projet_Tempo/insertTemp.php?";
requete = "GET /update?";

// Ajout de l'API Key dans l'URL
requete += "api_key=";
requete += key;

// Ajout de la température dans l'URL
//requete += "&tempo=";
requete += "&field1=";
requete += data;

// Imprime les données sur le port série
Serial.println(requete);

// Attente de 2 secondes

delay(1000);

// La fonction update() est lancé
update();
}

void update(){

// Établir une connexion TCP ou inscrire port UDP,
démarrer la connexion (184.106.153.149 = adresse du
serveur)
esp01.println("AT+CIPSTART=\"TCP\",\"184.106.153.149\",80
");
echo(2000);

// Envoyer des données
esp01.print("AT+CIPSEND=");
esp01.println(String(requete.length()+2));

echo();

//Ecrire la requete dans l'ESP
```

```
esp01.println(requete);

echo(2000);
}
```

Code page insertTemp.php :

```php
<?php
ini_set('display_errors', 'on');
$servername = "127.0.0.1";
$username = "root";
$password = "";
$dbname = "sensor_data";
try{
$pdo = new PDO("mysql:host=$servername;dbname=sensor_data", $username, $password);
// set the PDO error mode to exception
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}
catch(PDOEception $e){
    echo $e->getMessage();

}

$api_key_value = "tPmAT5Ab3j7F9";

if ($_SERVER["REQUEST_METHOD"] == "GET") {
$api_key = $_GET["apiKeyValue"];
if($api_key == $api_key_value) {
     $temperature = $_GET["tempo"];
     $today = date("H:i:s");
     $today2 = date("Y-m-d H:i:s");
     /*$arrayD[] = $dist;
     $arrayT[] = $today;*/


       $sql = "INSERT INTO `SensorData`(`tempo`, `DateN`) VALUES (:tempo,:DateN)";
        $res = $pdo->prepare($sql);
        $exec = $res->execute(array(":tempo"=>$temperature,":DateN"=>$today2));

  }
  else {
        echo "Wrong API Key provided.";
    }
    }
  else {
    echo "No data posted with HTTP POST.";
  }
```

C’est sur cette page que les données sont envoyées par l’ESP et insérées dans la base de données. Il y a une clé qui permets de sécuriser un peu plus le transfert des données.

Code index.php :

```
<!DOCTYPE html>
<html>
<head>

       <link rel="stylesheet" href="style.css" />
      <link rel="shortcut icon" type="image/x-icon" href="images/microchip.ico" />
        <title>Temperature</title>
        <h1>CAPTEUR DE TEMPERATURE</h1>
</head>
<body>
 <div>
 <?php include ('grapheTempo.php'); ?>
    </div>
<script src="https://code.jquery.com/jquery-2.1.1.min.js" type="text/javascript"></script>
<script>
$(document).ready(function(){
setInterval(function(){
$("#content").load('BaseTempo.php #box1')
}, 1000);
});
</script>
<div id="content"></div>
</body>
<html>
```

C’est sur cette page qu’on va afficher les deux différentes pages c’est-à-dire les pages qui vont afficher le graphique et l’autre le tableau.

Code grapheTempo.php :

```php
<?php
ini_set('display_errors', 'on');
// Établir une connexion à la base de données en utilisant PDO
try {
    $conn = new PDO("mysql:host=localhost;dbname=sensor_data", "root", "");
    // Définir le mode d'erreur pour PDO sur exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    }
catch(PDOException $e)
    {
    echo "Connection failed: " . $e->getMessage();
    }

// Préparer et exécuter une requête pour récupérer les températures de la base de données
$stmt = $conn->prepare("SELECT * FROM SensorData ORDER BY DateN DESC LIMIT 20");
$stmt->execute();
$data = $stmt->fetchAll();
foreach($data as $row){
$temperatures[] = $row['tempo'];
$temps[] = $row['DateN'];

}

// Récupérer les résultats de la requête dans un tableau

?>
<!DOCTYPE html>
<html>
<head>
    <title>Graphique de températures</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js@2.9.3/dist/Chart.min.js"></script>
</head>
<body>
    <canvas id="tempChart"></canvas>
    <script>
        var ctx = document.getElementById('tempChart').getContext('2d');
        var chart = new Chart(ctx, {
            type: 'line',
            data: {
                labels: <?php echo json_encode($temps) ?>,
                datasets: [{
                    label: 'Températures',
                    data: <?php echo json_encode($temperatures) ?>,
                    backgroundColor: 'rgba(255, 99, 132, 0.2)',
                    borderColor: 'rgba(255, 99, 132, 1)',
                    borderWidth: 1
                }]
            },
            options: {
                scales: {
                    y: {
                        beginAtZero: true
                    }
                }
            }
        });
    </script>
</body>
</html>
<?php
header("refresh: 5");
?>
```

Cette partie du code, gère le graphique, elle permets également d'actualiser le graphique toute les 5 secondes.

Code BaseTempo.php :

```php
<html>
<head>
<?php
ini_set('display_errors', 'on');
?>
      <link rel="stylesheet" href="style.css" />

</head>
<div id="box1" >

<?php
$now = date('Y-m-d H:i:s');
echo "<p>$now</p>";

$servername = "127.0.0.1";
$username = "root";
$password = "";
$dbname = "sensor_data";

try{

    $db = new PDO("mysql:host=$servername;dbname=sensor_data", $username, $password);
}
catch(PDOEception $e){
    echo $e->getMessage();

}
$stmt = $db->prepare('SELECT * FROM SensorData ORDER BY DateN DESC LIMIT 0,10');
$stmt->execute();
$mes2 = $stmt->fetchAll();

$stmt4 = $db->prepare('SELECT Min(tempo) as donnees FROM SensorData');
$stmt4->execute();
$msg5 = $stmt4->fetch();

$stmt2 = $db->prepare('SELECT Max(tempo) as donnees FROM SensorData');
$stmt2->execute();
$msg3 = $stmt2->fetch();
//echo $msg3['donnees'];

$stmt3 = $db->prepare('SELECT AVG(tempo) as donnees FROM SensorData');
$stmt3->execute();
$msg4 = $stmt3->fetch();
 ?>
```

Cette partie du code se connecte à la base de données et effectue cinq requêtes différentes qui vont les dix dernières températures, le minimum et le maximum des températures mesurées. Pour finir, elle affichera également la moyenne de la température.

```
 <div id="tableau">
<table id="tableau-style">

                <thead>
                <tr>
                        <th colspan="4">DONNÉES</th>
                </tr>
                <tr>
                        <th>DATE</th>
                        <th>VALEUR</th>

                </tr>
                 </thead>
              <tbody>
              <?php
              foreach ($mes2 as $select) {
                  ?>
                  <tr>
                   <td><?= $select['DateN'];?></td>
                   <td><?= $select['tempo'];?></td><br>
                  </tr>
             <?php }?>
               </tbody>
               </table>

               <table id="statistique">
                 <tr>
                          <th>MIN</th>
                          <th>MOYENNE</th>
                         <th>MAX</th>


                 </tr>
                 <tr>
                        <td><div id="col_min"><?php echo $msg5['donnees']; ?></div></td>
                        <td><div id="col_moyenne"><?php echo $msg4['donnees']; ?></div></td>
                        <td><div id="col_max"><?php echo $msg3['donnees'] ?></div></td>

                 </tr>
               </table>
</div>

</div>
</html>
```

Dans cette partie de code, nous affichons en tableau les valeurs récupérées grâce aux requêtes réalisées précédemment. La partie de gauche du tableau va afficher les dernières valeurs récupérés et la partie de droite va afficher la valeur minimum, maximum et la moyenne.

La Base de données :

tempo	DateN
23.95	2023-01-14 22:36:41
23.46	2023-01-14 22:37:20
24.44	2023-01-14 22:37:34
24.44	2023-01-14 22:37:48
24.44	2023-01-14 22:38:04
24.44	2023-01-14 22:38:17
23.95	2023-01-14 22:38:31
24.44	2023-01-14 22:38:45
24.44	2023-01-14 22:39:01
24.44	2023-01-14 22:39:14
23.95	2023-01-14 22:39:28
23.95	2023-01-14 22:39:42
24.93	2023-01-14 22:39:58
23.95	2023-01-14 22:40:11
23.95	2023-01-14 22:40:29
23.46	2023-01-14 22:40:39
22.48	2023-01-15 01:30:05

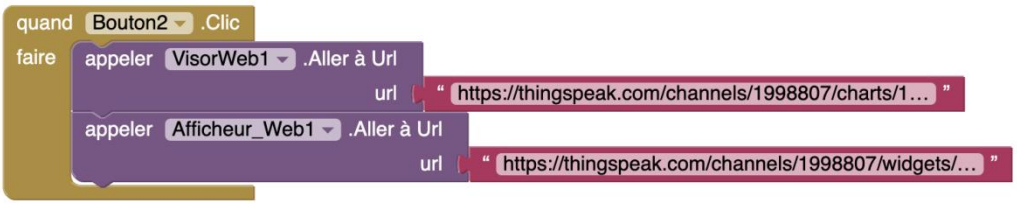
Cette capture représente la base de données, où les données sont stockées.

App Inventor, c’est quoi ?

C’est un logiciel en ligne qui permet de créer des applications pour appareils Android à travers une interface purement visuelle (WYSIWYG pour « What You See Is What You Get ») et de configurer les actions de l’application par un système de blocs logiques. . L’outil est gratuit et permet de développer sa créativité et ses compétences en programmation. Ainsi, vous pourrez réaliser vos applications et les installer sur votre smartphone. - Il vous faudra obligatoirement un compte google ! Et un smartphone ou tablette Android pour tester votre application avec l’application MIT AI2 Companion installée. La conception des applications se fera essentiellement en ligne donc pour se faire l'utilisateur doit être muni :

- D'un ordinateur.
- D'un navigateur
- Une connexion fiable à Internet.
- Un compte Google
- un terminal (smartphone ou tablette) avec un système d’exploitation Android version 2.3 ou supérieur

Code Application mobile avec l’API ThingSpeak:



05:43 95

Ulrick_Température

Capteur Température

Afficher Graphique

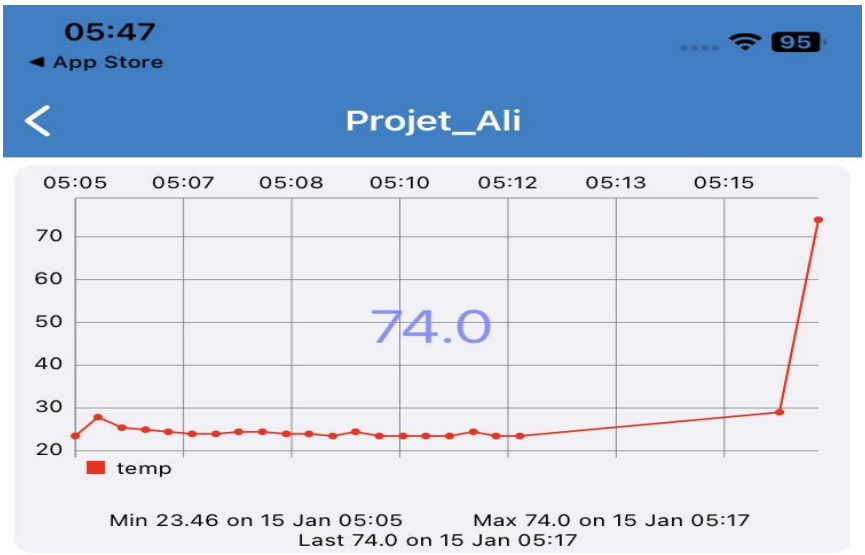
Dernière Température

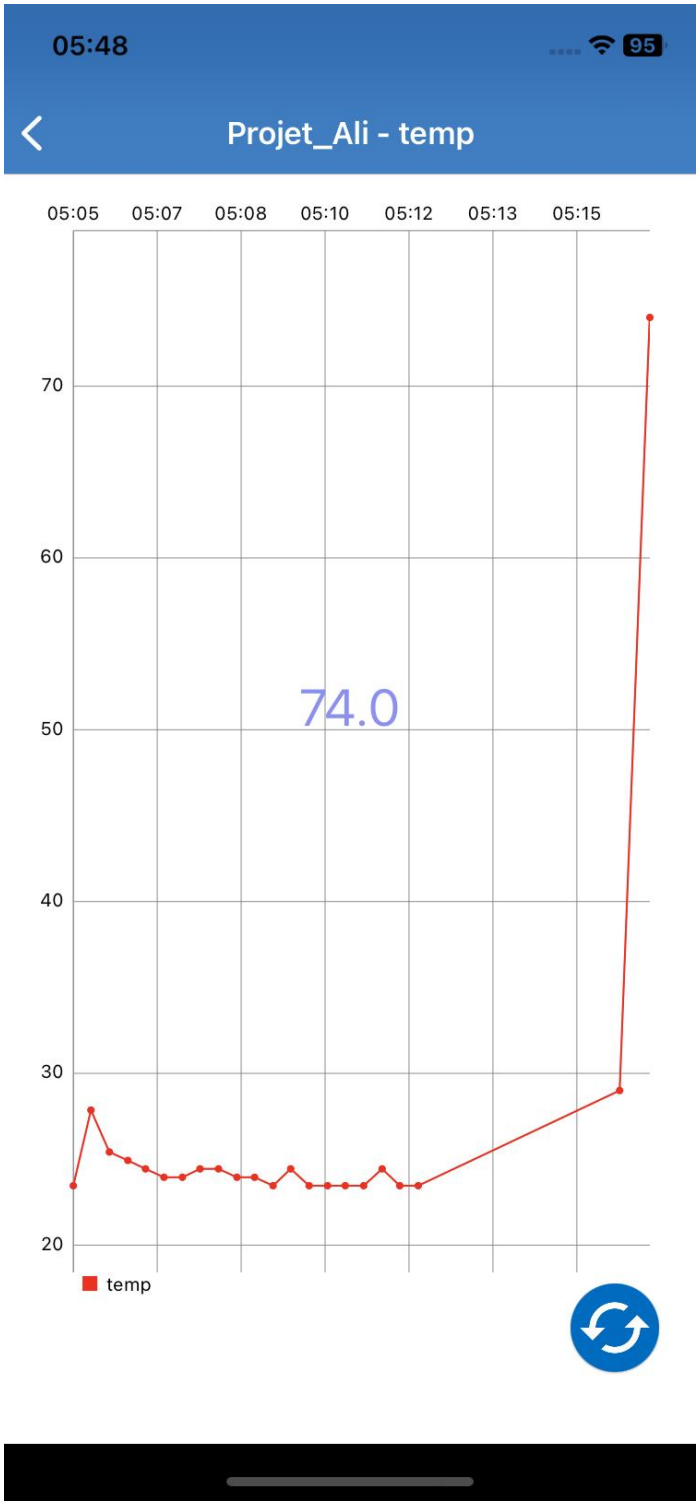
74.0

26 minutes ago

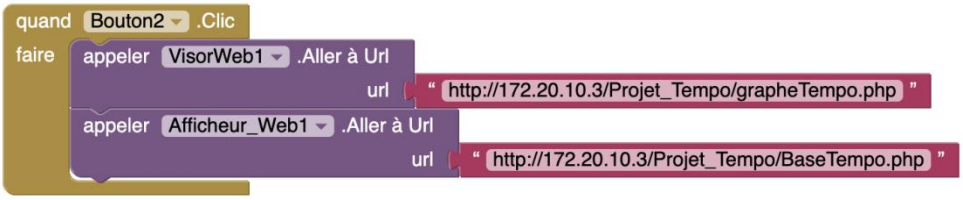
ThingViewFree (application de Thingspeak) :

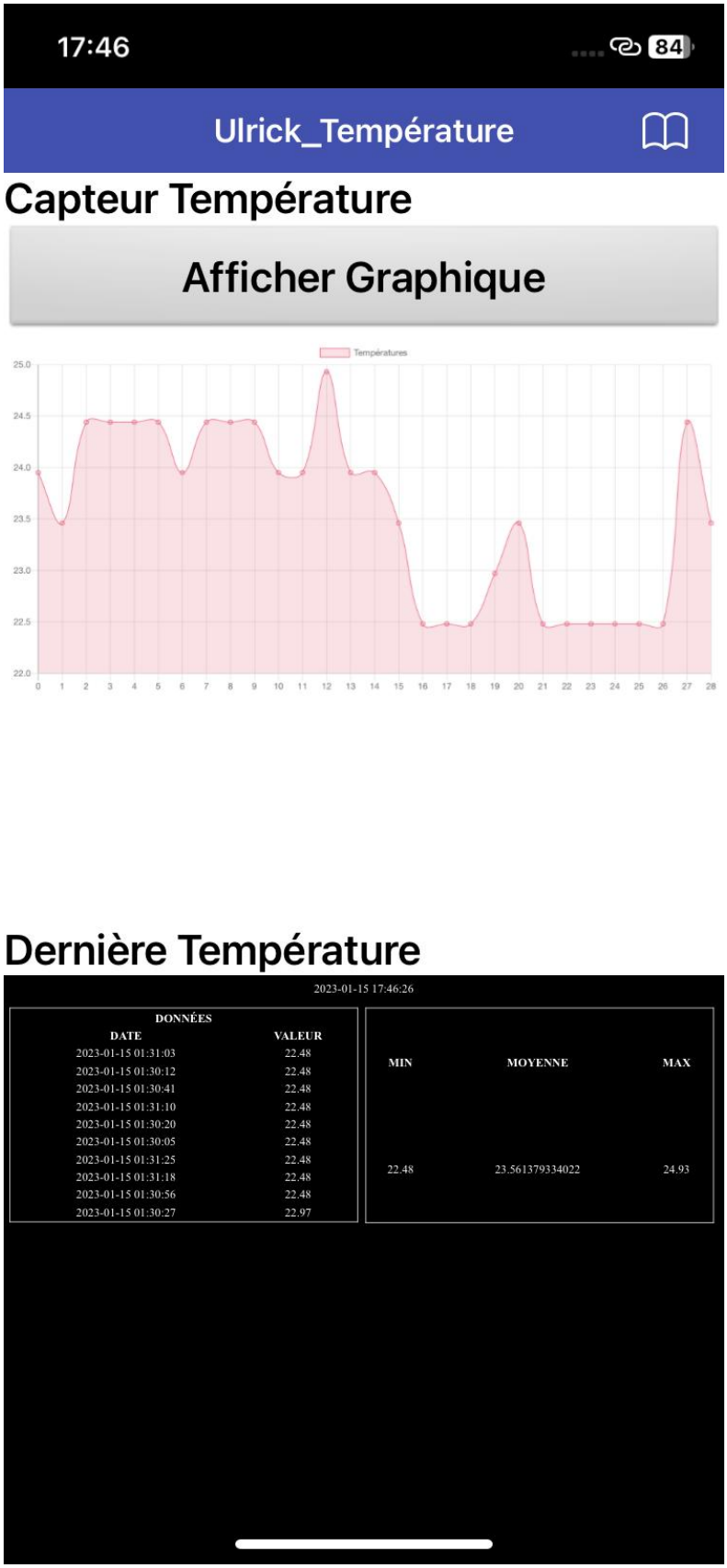
Il y a également une application dédié directement qui va afficher les graphiques avec plus de details . Ce sont ces deux photos ci-dessous :




05:48
95
Projet_Ali - temp
05:05
05:07
05:08
05:10
05:12
05:13
05:15
70
60
74.0
50
40
30
20
temp

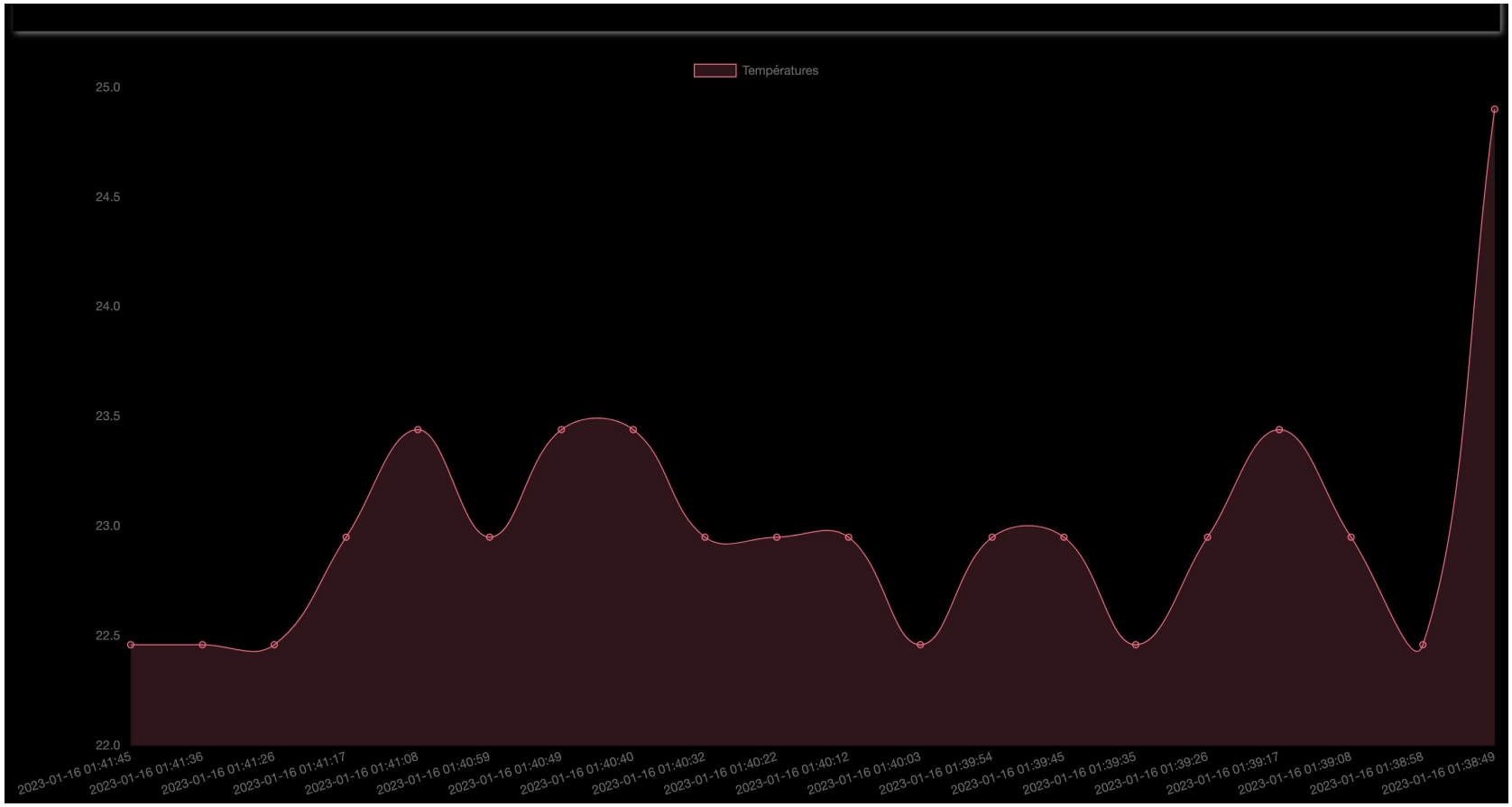
Code Application mobile avec mon API :

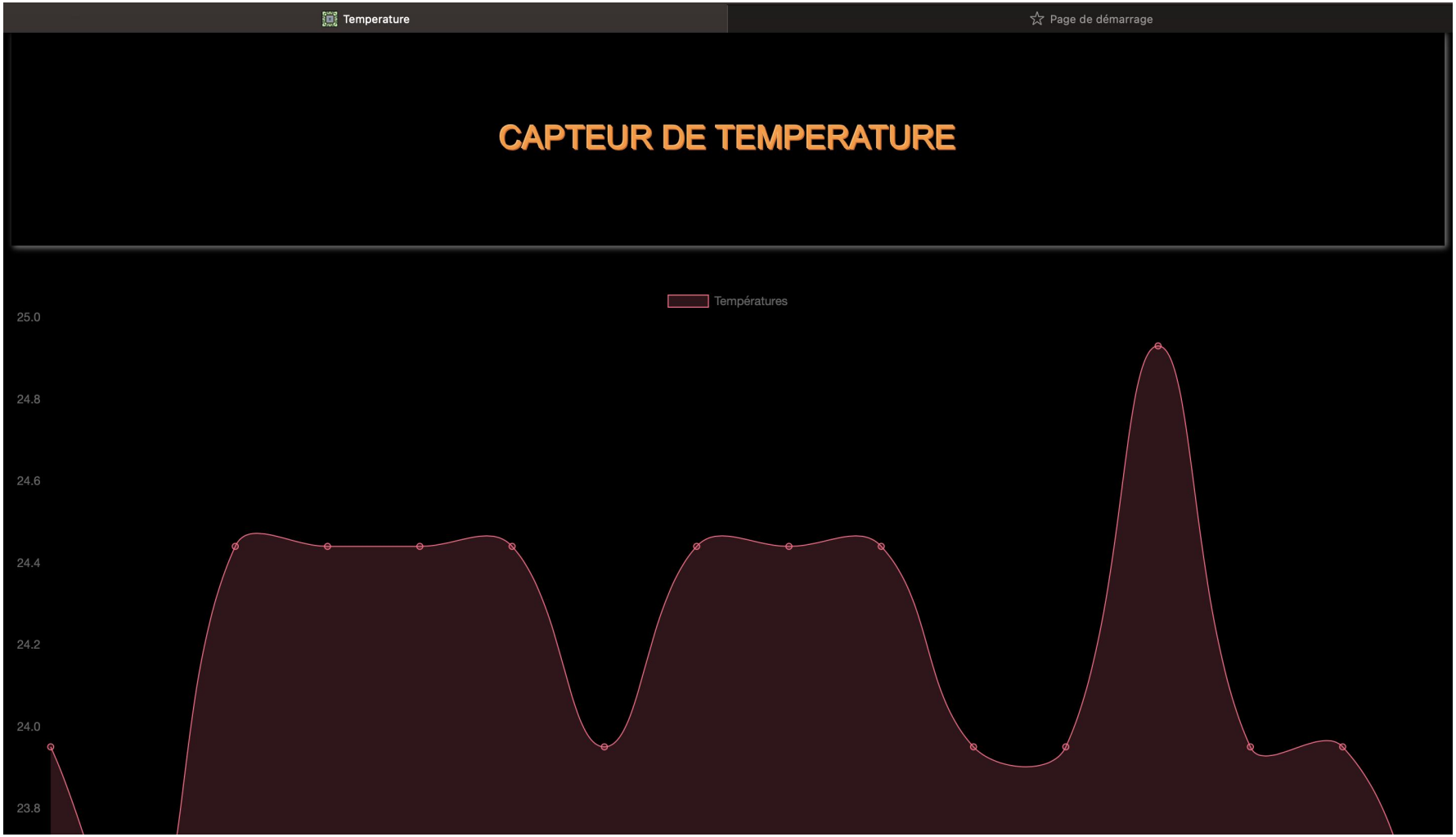




Cette image est une capture d’écran de l’application mobile de mon API.

Application Web (mon API):

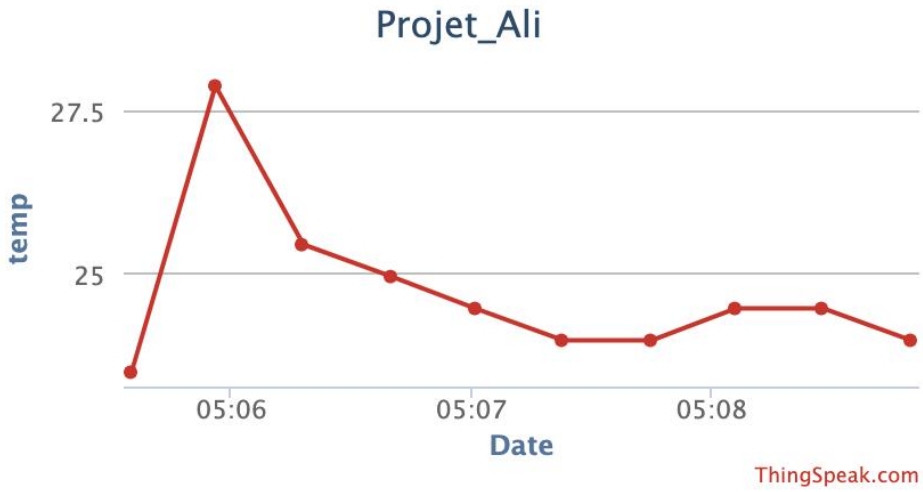



Temperature
Page de démarrage
CAPTEUR DE TEMPERATURE
Températures
25.0
24.8
24.6
24.4
24.2
24.0
23.8

DONNÉES	
DATE	**VALEUR**
2023-01-14 22:39:58	24.93
2023-01-14 22:38:17	24.44
2023-01-14 22:37:34	24.44
2023-01-14 22:37:48	24.44
2023-01-14 22:38:04	24.44
2023-01-14 22:39:14	24.44
2023-01-14 22:39:01	24.44
2023-01-14 22:38:45	24.44
2023-01-14 22:38:31	23.95
2023-01-14 22:39:28	23.95

MIN	MOYENNE	MAX
23.46	24.164375424385	24.93

Application Web (API Thingspeak):



Récaputilatif :

Le projet suit le cheminement suivant :

- Le capteur de température LM35 alimenté par l'arduino récupère la température.
  - Mesure la tension sur la broche A0.
  - Transforme la mesure (nombre entier) en température via un produit en croix.
- Le module ESP se connecte au réseau Wifi.
- Le module ESP envoie une requete à l'IP qu'on aura déclaré.(dans mon cas l'IP de mon serveur).
  - Initialise la communication avec le PC.
  - Initialise la communication avec l'ESP.
  - Définir l'ESP en mode AT.
  - Connexion au point d'accès.
  - Établir une connexion TCP ou inscrire port UDP, démarrer la connexion.
  - Conversion de la temperature en string.
  - Création de la requete pour envoyer.
  - Ajout de l'API Key et de la température dans l'URL.
  - Envoyer des données
- La page insertTemp.php reçois les données et l'insère dans la base de données.
- On peut visualiser les données sur notre page PHP, mais également sur l'application mobile.
- Une visualisation en local sur une page web.
- Une visualisation des données dans une base MySQL.

Conclusion :

Ce projet m'a permis d'approfondir mes connaissances sur la programmation, notamment en me documentant sur chaque élément du projet ainsi que leur fonctionnement individuel. J'ai pu me rendre compte qu'avant de programmer, il fallait assimiler de nombreux prérequis qui me permettrait de comprendre le cheminement pour coder correctement.

Étant donné que nous sommes dans une société où tous les objets sont connectés et que la plupart sont dotés de capteurs ainsi que de microcontrôleur. Ce projet est un mélange de ces grandes avancées technologiques qui ne cessent d'accroître son optimisation.

J'ai pu remarquer que de nombreux projets existants utilisait déjà ce capteur de température. Ce qui permet de voir que cette technologie touche tous les secteurs et laisse un large choix d'invention. Ce projet m'a permis de réaliser qu'il était possible d'afficher des valeurs sur une carte en même temps envoyer les valeurs sur un serveur ce qui va permettre un affichage sur une application Web. J'ai pu également connecter un capteur Wifi et envoyer les valeurs sur une application mobile.

J'ai mis deux manières de gérer les données, une avec une API externe et une avec une API que j'ai créé du début jusqu'à la fin.

Le forum Arduino.com m'a été d'une grande aide pour toute question, ayant un compte que j'ai pu m'informer.

lien du projet :

https://drive.google.com/file/d/1cPwti6ijXOR8_C09Dfg0IeU76moTb1A9/view?usp=share_link

- 1 code arduino (sketch_nov20a.ino est le code de mon API)
- 1 code arduino (sketch_nov20a_Thing.ino est mon code qui utilise ThingSpeak)
- 4 code de pages en PHP
  - BaseTempo.php
  - grapheTempo.php
  - insertTemp.php
  - index.php)
- code CSS

Pour mon API fichier requis : sketch_nov20a.ino , les 4 codes PHP et Css.

Pour l'API ThingSpeak : sketch_nov20a_Thing.ino et créer un compte sur ThingSpeak.