Collège de Maisonneuve

RASPBERRY PI
PROJET DE CAPTEUR DE TEMPÉRATURE ET D'HUMIDITÉ

Applications pour des objets connectés 00SX

Février 2026

Andrea Silva

Table des matières

I - Introduction 3
II - Résumé du projet 3
III - Analyse 5
IV – Conception 6
Architecture technique (vue d'ensemble) 6
Architecture logique 6
Flux de communication 7
Diagramme de séquence 7
V - Technologies utilisées 8
VI - Environnement technique 8
VII - Implémentation 9
Installation du système d'exploitation sur le Raspberry Pi 9
Connexion des broches du capteur DHT22 au Raspberry Pi 20
Installation de Python 3 22
Installation de la bibliothèque Adafruit pour le DHT22 23
Test de lecture du capteur DHT22 24
Configuration de Firebase 26
Test d'envoi des données vers Firebase 27
L'étape suivante consiste à créer un script Python pour le capteur DHT22 destiné à la production, qui enverra les données à Firebase 28
Exécutez le script dht_firebase.py au démarrage à l'aide des services 30
Configuration SSH 31
VIII - Développement 32
Développement backend 32
Développement du frontend 34
IX – Planification 35
X - Plan de déploiement 36
Déploiement du module IoT (Raspberry Pi) 36
Déploiement de l'application React sur Firebase Hosting 36

I - Introduction

Dans un contexte où la surveillance environnementale devient de plus en plus importante, notamment pour les espaces intérieurs, les laboratoires ou les environnements techniques, j'ai conçu et développé un système de monitoring basé sur les technologies IoT et le développement web moderne.

L'objectif principal de ce projet est de mesurer la température et l'humidité en temps réel à l'aide d'un capteur DHT22 connecté à un Raspberry Pi, puis de stocker ces données dans le cloud via Firebase.

Les informations collectées sont ensuite exploitées à travers une application web développée avec React et TypeScript, offrant un tableau de bord interactif et responsive grâce à Bootstrap. Cette interface permet de visualiser les mesures en temps réel et de suivre leur évolution de manière claire et structurée.

Ce projet me permet ainsi de combiner administration système sous Linux, programmation Python pour l'acquisition des données, et développement frontend moderne pour la visualisation, afin de créer une solution complète et fonctionnelle de surveillance environnementale.

II - Résumé du projet

Dans ce projet, j'ai développé un système de surveillance de la température et de l'humidité en temps réel en utilisant un Raspberry Pi et un capteur DHT22.

Le capteur DHT22 est connecté au Raspberry Pi et me permet de mesurer régulièrement la température et le taux d'humidité d'un environnement. Le Raspberry Pi récupère ces données, les traite, puis les envoie vers Firebase, que j'utilise comme base de données pour stocker les informations de manière sécurisée et structurée.

Ensuite, j'ai développé une application web avec React Js afin d'afficher ces données sous forme de tableau de bord interactif. Le dashboard permet de visualiser les valeurs en temps réel et de suivre l'évolution des mesures au fil du temps.

À travers ce projet, j'ai voulu combiner l'IoT et le développement web moderne pour créer une solution simple, fonctionnelle et accessible pour le monitoring environnemental.

Technologies Principales

- **Raspberry Pi:** Le Raspberry Pi me sert de mini-ordinateur et de passerelle entre le capteur et le reste du système. Il lit les données envoyées par le capteur DHT22 et exécute le script chargé de transmettre ces informations vers la base de données.
- **Capteur DHT22:** Le capteur DHT22 me permet de mesurer la température et l'humidité. Il est reconnu pour sa précision et sa stabilité, ce qui le rend adapté à un projet de surveillance environnementale.
- **Python:** J'utilise Python sur le Raspberry Pi pour lire les données du capteur et les envoyer automatiquement vers Firebase. Le script s'exécute à intervalles réguliers afin d'assurer une mise à jour continue des mesures.

- **Firebase :** Firebase me permet de stocker les données dans le cloud en temps réel. Il simplifie la gestion de la base de données et facilite la connexion avec l'application web.
- **React :** J'ai développé le tableau de bord avec React. Cette bibliothèque me permet de créer une interface dynamique et réactive qui affiche les données en temps réel et met à jour les informations automatiquement.

Points Forts Techniques

- Architecture MVC garantissant une structure claire et maintenable du code.
- Utilisation de **Linux (Raspberry Pi OS)** comme environnement système stable pour l'exécution du projet.
- Configuration et administration du Raspberry Pi à distance via **connexion SSH**.
- Développement d'un **script Python** pour la lecture automatique des données du capteur DHT22.
- Transmission sécurisée des mesures vers **Firebase** pour le stockage en temps réel.
- Développement du dashboard avec **React et TypeScript**, garantissant un code plus structuré et typé.
- Interface responsive et moderne grâce à **Bootstrap**, adaptée aux différents appareils.
- Architecture modulaire séparant acquisition des données, stockage cloud et visualisation.

III - Analyse

Introduction

Avant de commencer la conception technique, j'ai pris le temps d'analyser le besoin et le fonctionnement global du système. L'idée était simple : récupérer automatiquement la température et l'humidité d'un environnement et pouvoir consulter ces données facilement à travers une interface web.

L'analyse m'a permis de définir clairement :

- Ce que le système doit faire
- Comment les différentes parties vont communiquer
- Quelles technologies utiliser
- Comment assurer la fiabilité des données

L'objectif principal est d'avoir un système automatisé, stable et simple à utiliser.

Besoins fonctionnels

Le système doit permettre :

- De lire automatiquement les données du capteur DHT22
- D'envoyer ces données vers Firebase
- De stocker les mesures dans le cloud
- D'afficher les données en temps réel sur un tableau de bord

Le tout doit fonctionner sans intervention manuelle une fois configuré.

Cas d'utilisation

Dans ce projet, je suis à la fois l'administrateur et l'utilisateur du système.

Les principaux cas d'utilisation sont :

- Lire les données du capteur
- Le Raspberry Pi exécute un script Python qui lit la température et l'humidité toutes les 60 minutes.
- Envoyer les données vers Firebase
- Les mesures sont envoyées automatiquement via l'API REST vers Firebase Realtime Database.
- Consulter les données
- Je peux ouvrir l'application React pour visualiser les données en temps réel et observer leur évolution.

IV – Conception

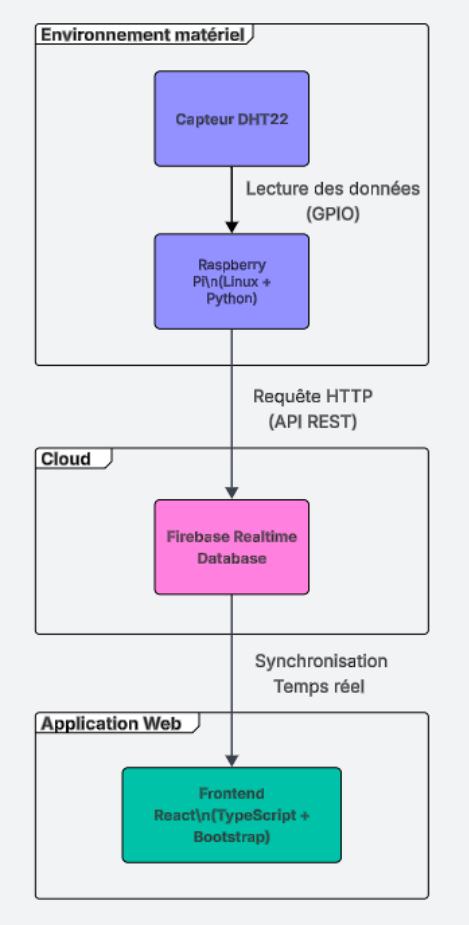
Architecture technique (vue d'ensemble)

J'ai choisi une architecture simple et claire, composée de quatre éléments principaux :

1. Le capteur DHT22
2. Le Raspberry Pi (Linux + Python)
3. Firebase Realtime Database
4. L'application web React (TypeScript + Bootstrap)

Le Raspberry Pi joue le rôle d'intermédiaire : il récupère les données du capteur et les envoie vers le cloud. Ensuite, l'application React récupère ces données pour les afficher sur le tableau de bord.

Architecture logique



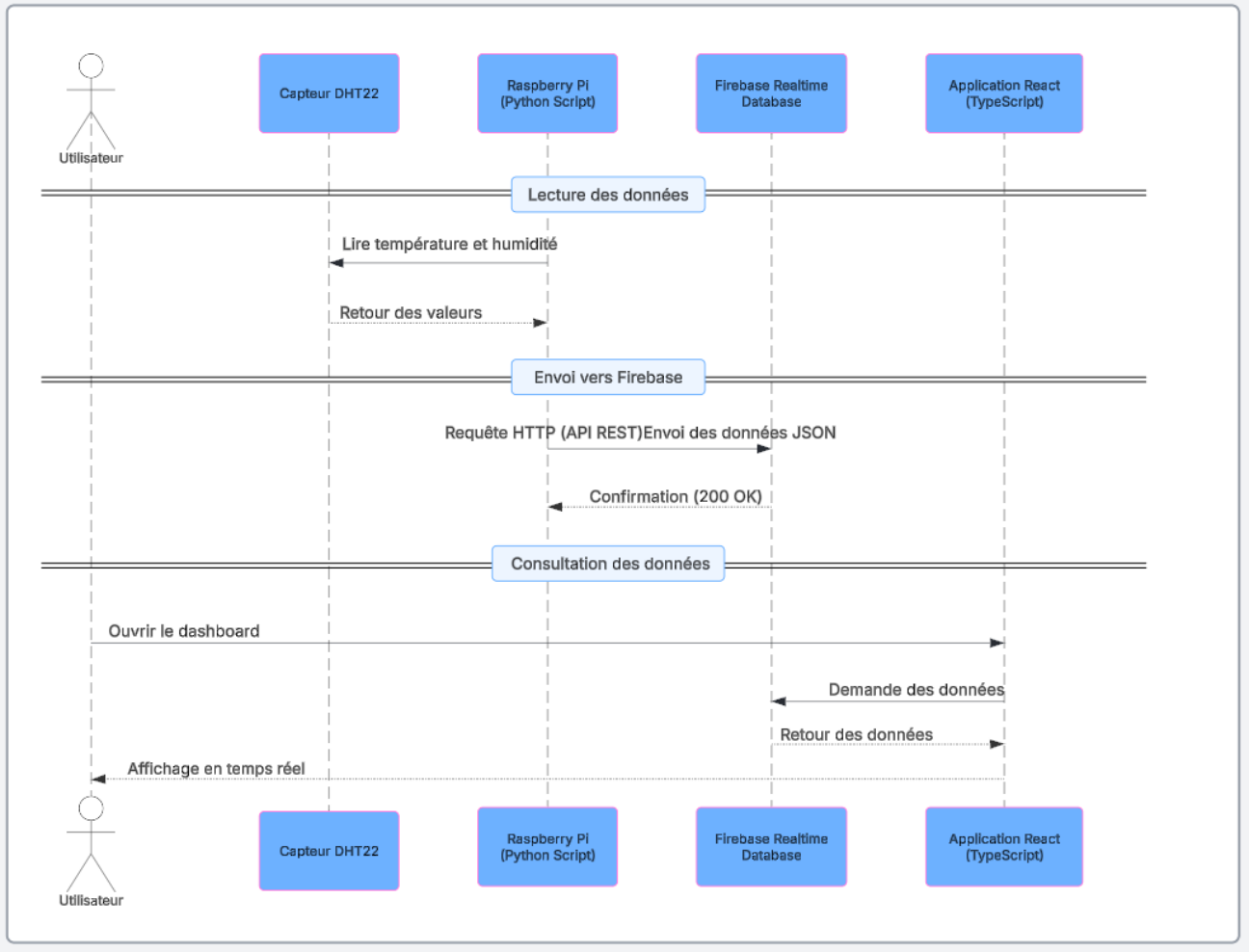
Flux de communication

Le fonctionnement du système suit une séquence logique :

1. Le capteur mesure la température et l'humidité.
2. Le script Python lit ces valeurs.
3. Les données sont envoyées vers Firebase via une requête HTTP.
4. Firebase enregistre les informations.
5. L'application React récupère les données en temps réel et met à jour l'affichage.

Ce processus se répète automatiquement toutes les 60 minutes.

Diagramme de séquence



V - Technologies utilisées

- **Raspberry Pi :** plateforme matérielle pour la collecte des données.
- **Capteur DHT22 :** mesure de la température et de l'humidité.
- **Python 3 :** lecture du capteur et envoi des données via API REST.
- **Firebase Realtime Database :** stockage cloud en temps réel.
- **React & TypeScript :** développement du tableau de bord.
- **Bootstrap :** interface responsive et moderne.

VI - Environnement technique

- **Système d'exploitation : Raspberry Pi OS (Linux).**
- **Accès distant : SSH.**
- **Gestion des services : systemd.**
- **Communication cloud : API REST Firebase.**
- **Hébergement Web : Firebase Hosting.**

VII - Implémentation

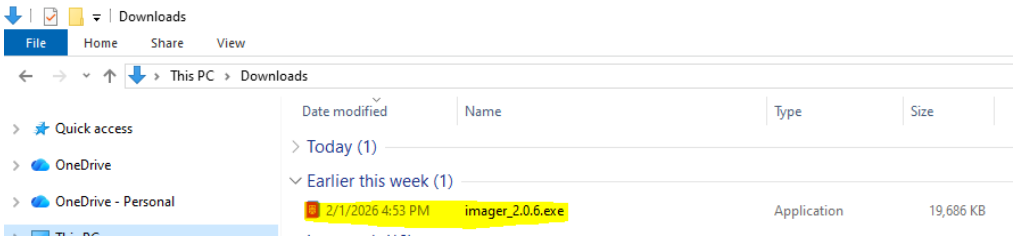
Le projet repose sur plusieurs environnements, allant de l'installation du système sur le Raspberry Pi jusqu'au développement de l'application web.

Installation du système d'exploitation sur le Raspberry Pi

Avant de commencer le développement, j'ai installé le système d'exploitation sur le Raspberry Pi.

J'ai téléchargé l'image officielle Raspberry Pi OS, puis je l'ai installée sur une carte microSD à l'aide de l'outil Raspberry Pi Imager.

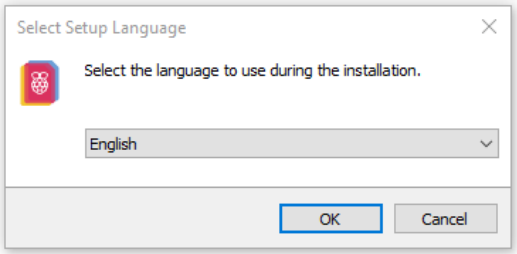
Télécharger: https://www.raspberrypi.com/software/

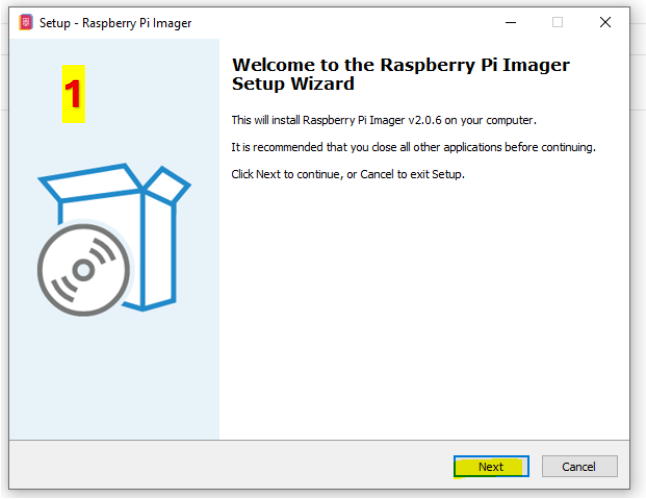


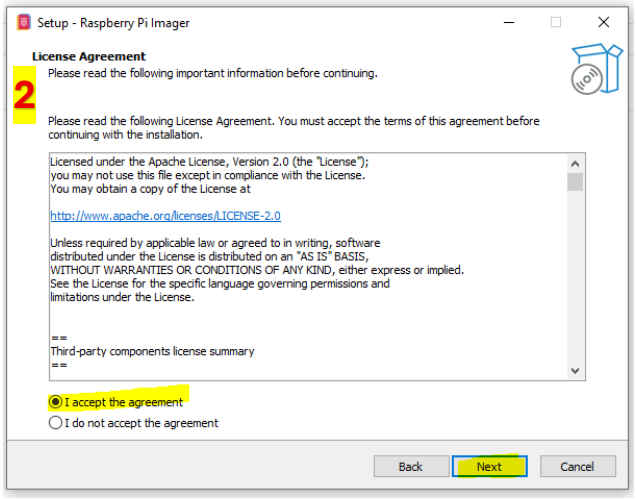
➔ Double-cliquez sur le fichier **imager_2.0.6.exe** et suivez les étapes suivantes

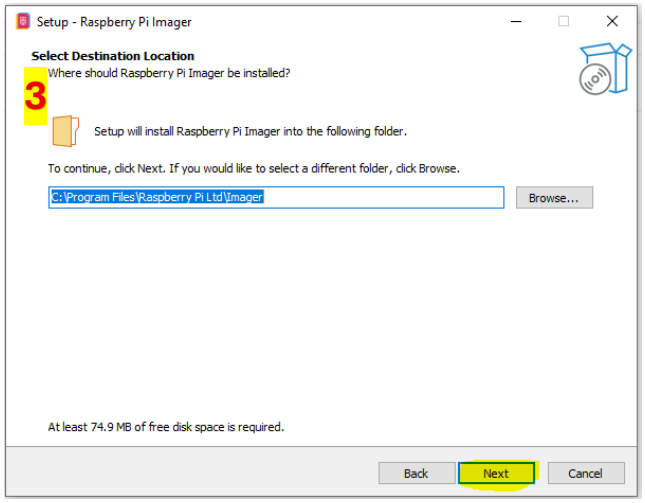
♦ Commencer l'installation avec **Raspberry Pi Imager**, puis suivre les étapes du Setup Wizard.

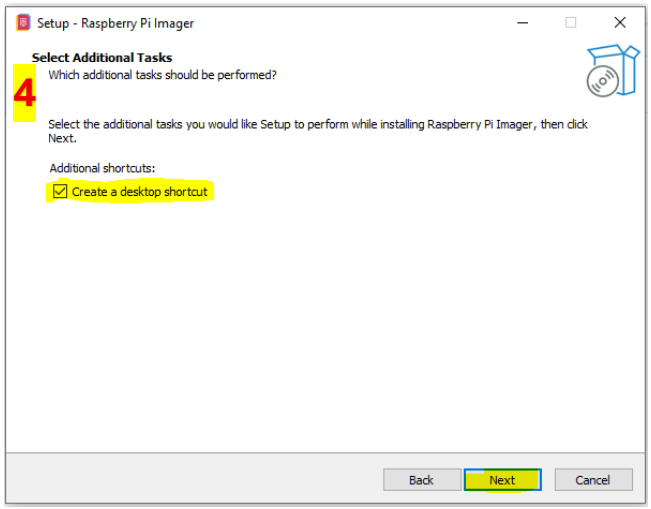
1. Choisissez la langue :

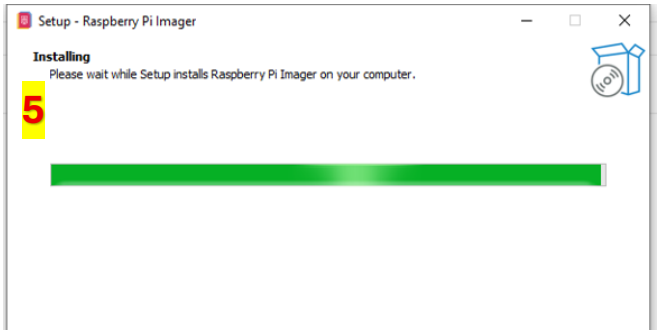


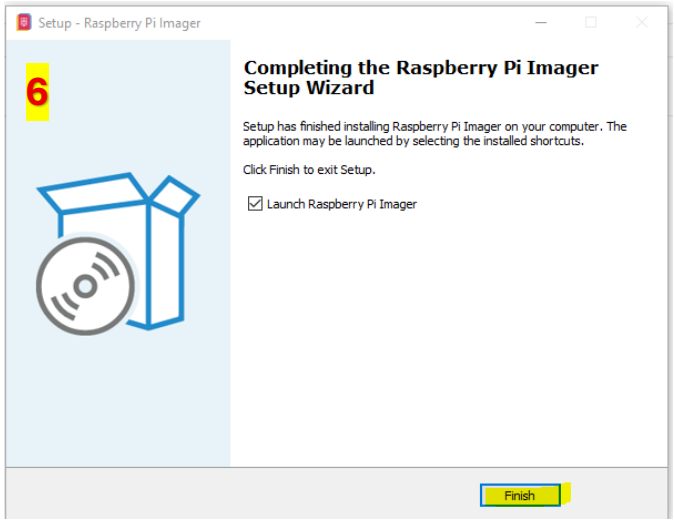

Setup - Raspberry Pi Imager
1
Welcome to the Raspberry Pi Imager Setup Wizard
This will install Raspberry Pi Imager v2.0.6 on your computer.
It is recommended that you close all other applications before continuing.
Click Next to continue, or Cancel to exit Setup.
Next
Cancel


Setup - Raspberry Pi Imager
License Agreement
Please read the following important information before continuing.
2
Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
==
Third-party components license summary
==
I accept the agreement
I do not accept the agreement
Back
Next
Cancel


Setup - Raspberry Pi Imager
Select Destination Location
Where should Raspberry Pi Imager be installed?
3
Setup will install Raspberry Pi Imager into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Program Files\Raspberry Pi Ltd\Imager
Browse...
At least 74.9 MB of free disk space is required.
Back
Next
Cancel

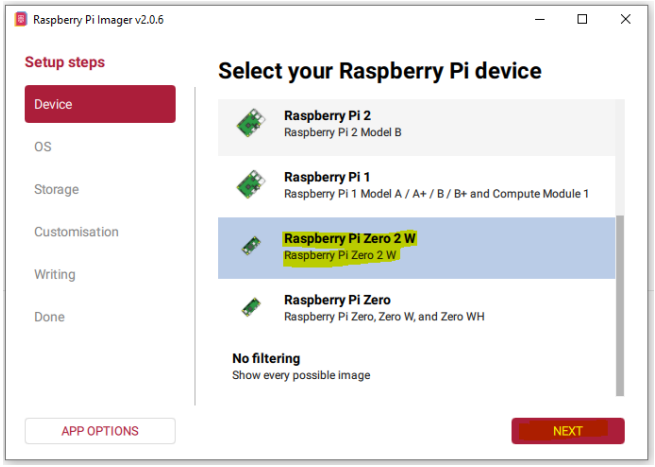

Setup - Raspberry Pi Imager
Select Additional Tasks
Which additional tasks should be performed?
4
Select the additional tasks you would like Setup to perform while installing Raspberry Pi Imager, then click Next.
Additional shortcuts:
Create a desktop shortcut
Back
Next
Cancel


Setup - Raspberry Pi Imager
Installing
Please wait while Setup installs Raspberry Pi Imager on your computer.
5

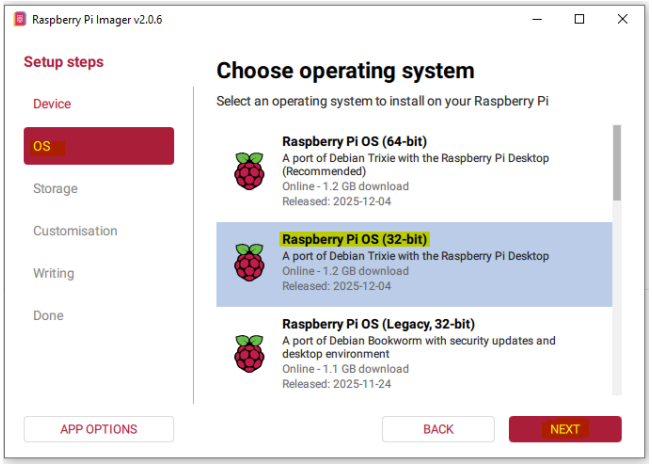

Setup - Raspberry Pi Imager
6
Completing the Raspberry Pi Imager Setup Wizard
Setup has finished installing Raspberry Pi Imager on your computer. The application may be launched by selecting the installed shortcuts.
Click Finish to exit Setup.
Launch Raspberry Pi Imager
Finish

- Une fois l'assistant terminé, la page de configuration de **Raspberry Pi Imager** s'ouvrira automatiquement.

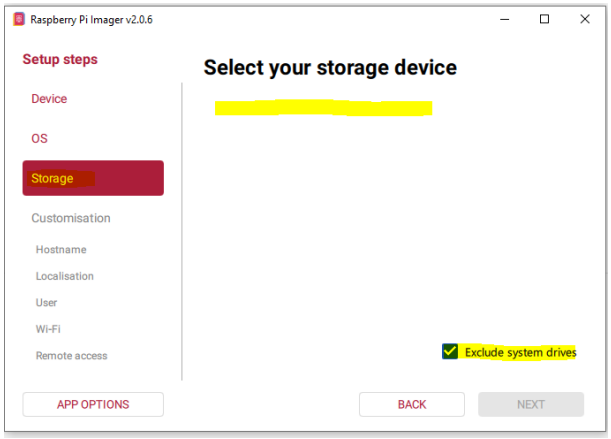
➔ **Choisissez votre modèle d'appareil Raspberry Pi**

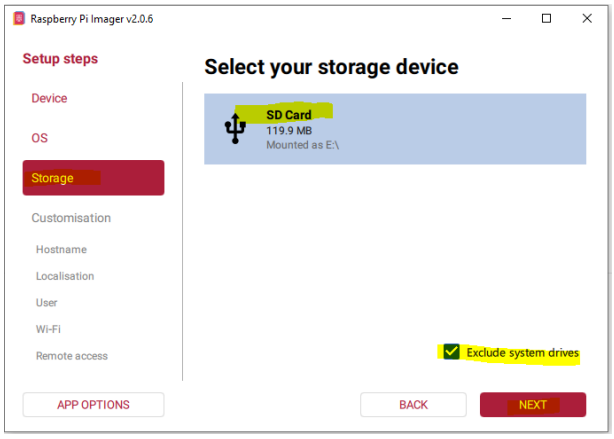


➔ **Choisissez le système d'exploitation**

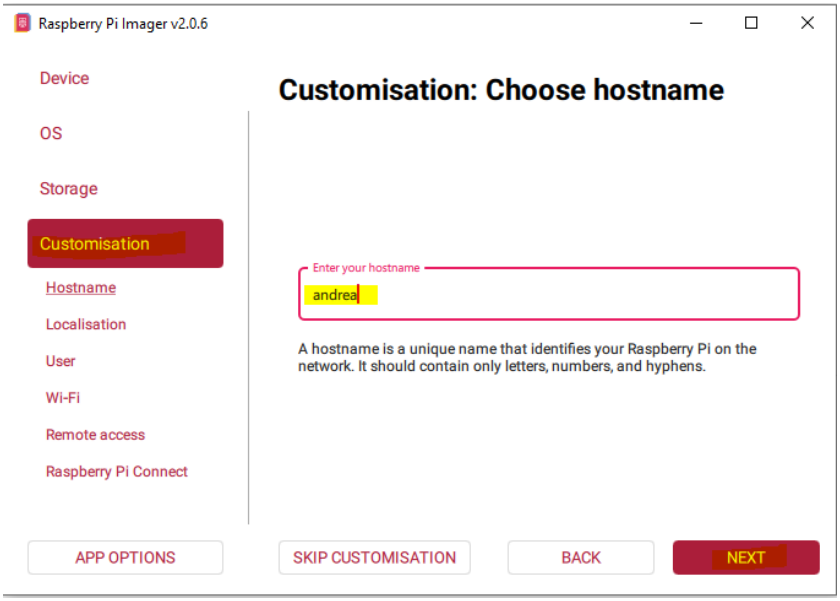


➔ **Sélectionnez votre périphérique de stockage :** à cette étape, insérez la carte SD dans votre ordinateur portable et sélectionnez-la.

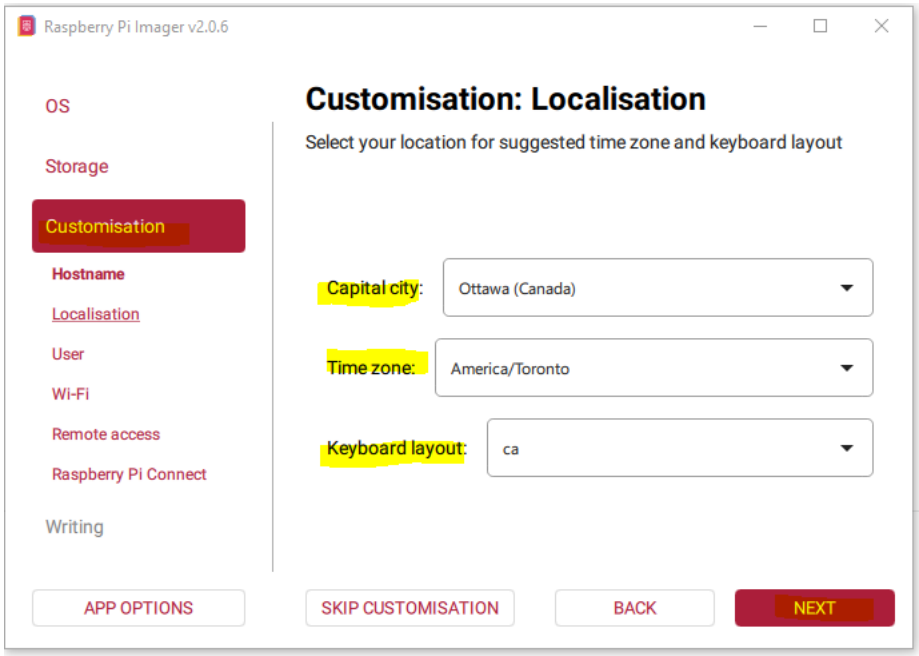




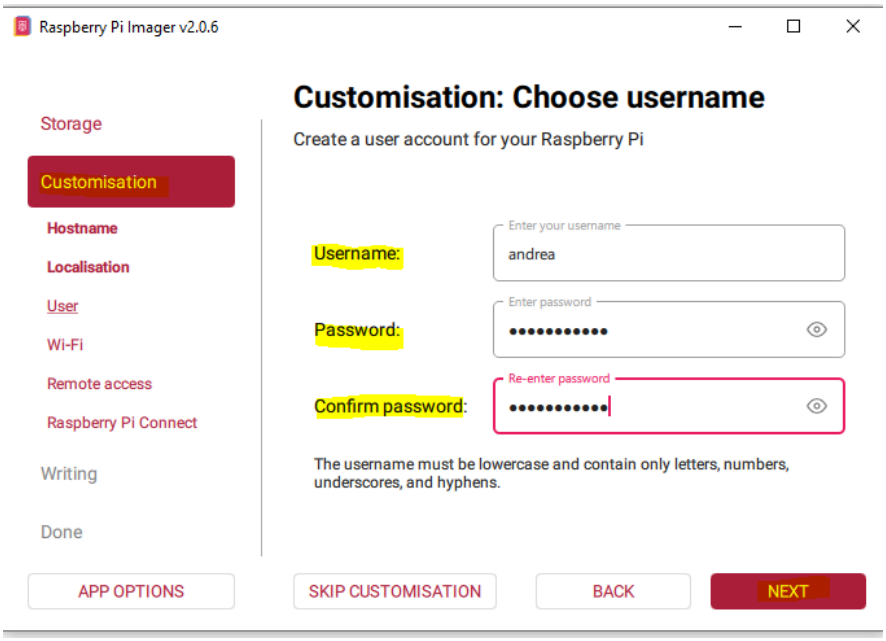
➔ **Choisissez le nom d'hôte (hostname):** notez-le pour pouvoir l'utiliser ultérieurement. Il s'agira du nom d'hôte dans le système d'exploitation Linux, sous la forme utilisateur@**hostname.**



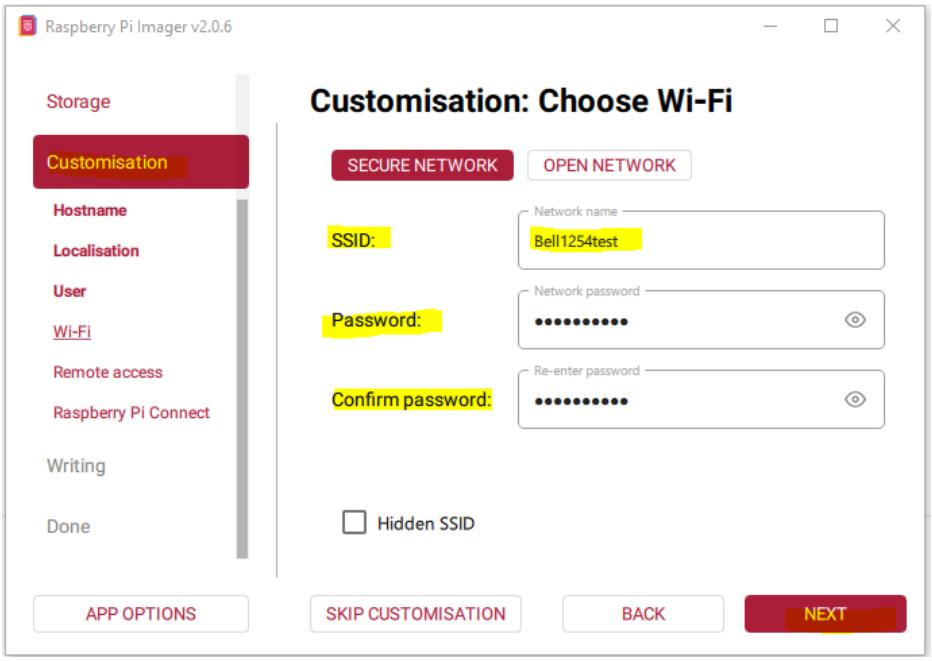
➔ **Sélectionnez l'emplacement :** cela peut être mis à jour ultérieurement dans la version bureau.



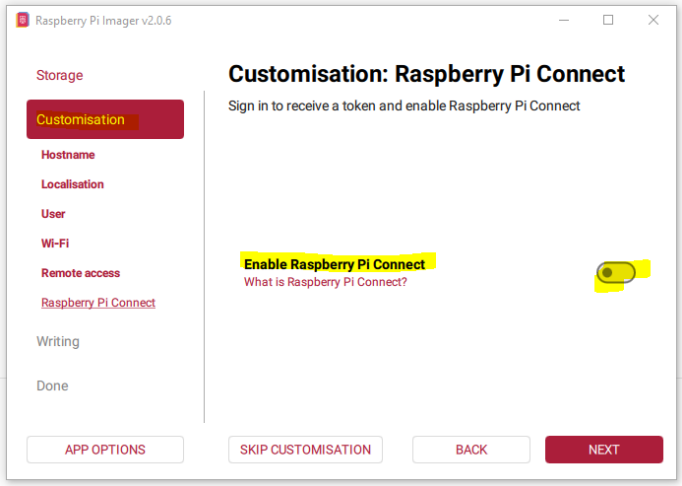
➔ **Choisissez un nom d'utilisateur et conservez-le, car vous l'utiliserez pour vous connecter au Raspberry Pi, en particulier via SSH.**

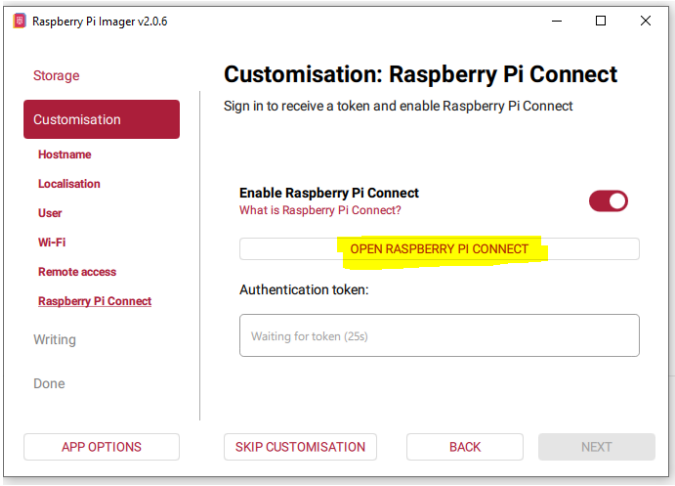


➔ **Entrez les informations de votre connexion Wi-Fi.**

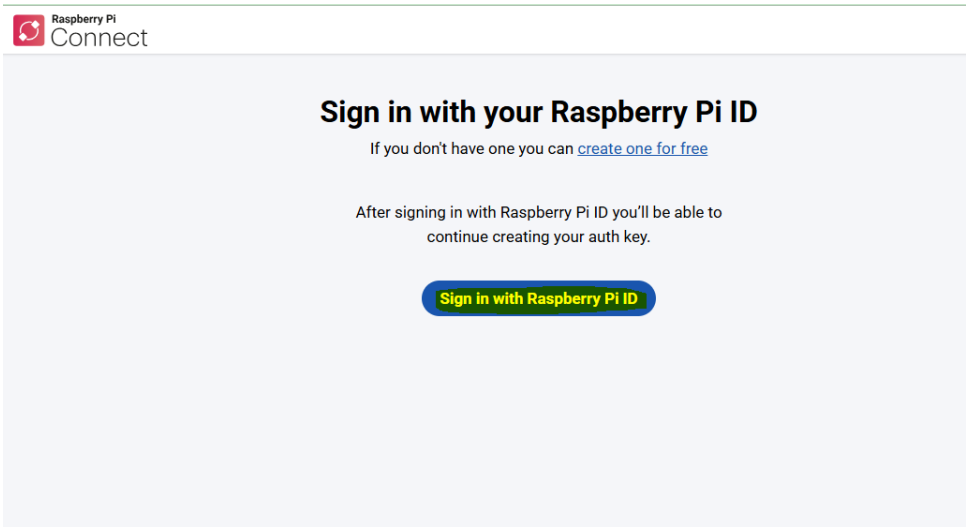


➔ **Raspberry Pi Connect permet d'accéder à votre Raspberry Pi à distance, où que vous soyez.**

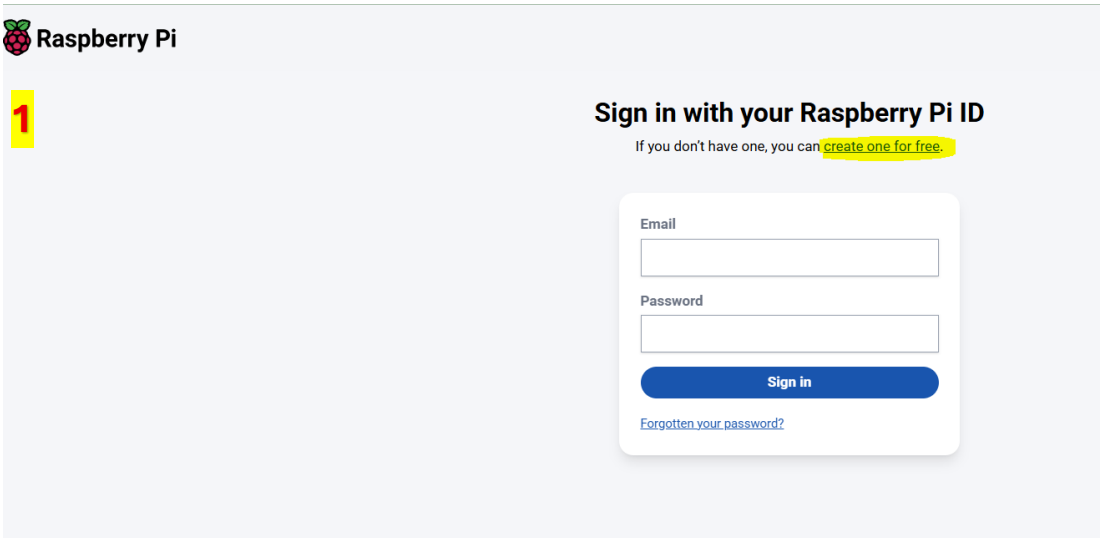


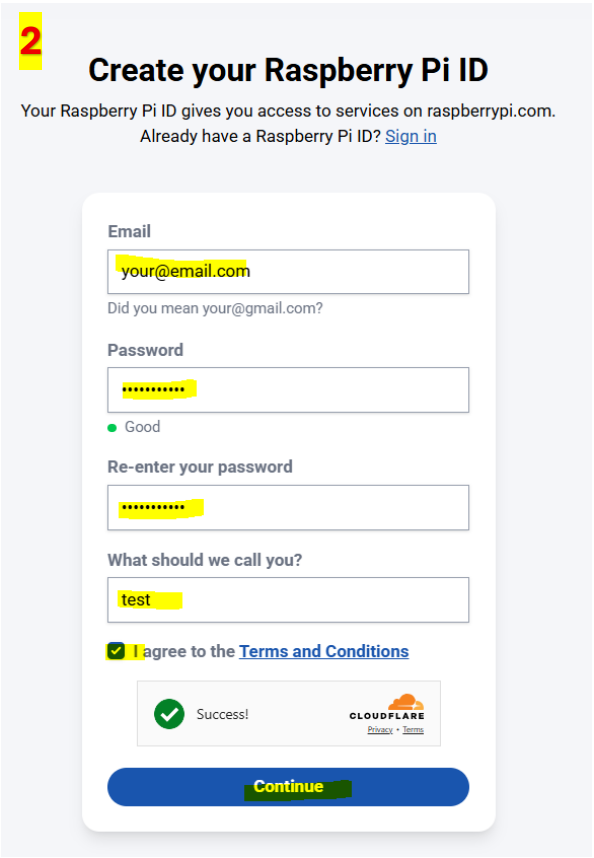


➔ Après avoir cliqué sur **Open Raspberry Pi Connect**, une nouvelle page du navigateur s'ouvrira.

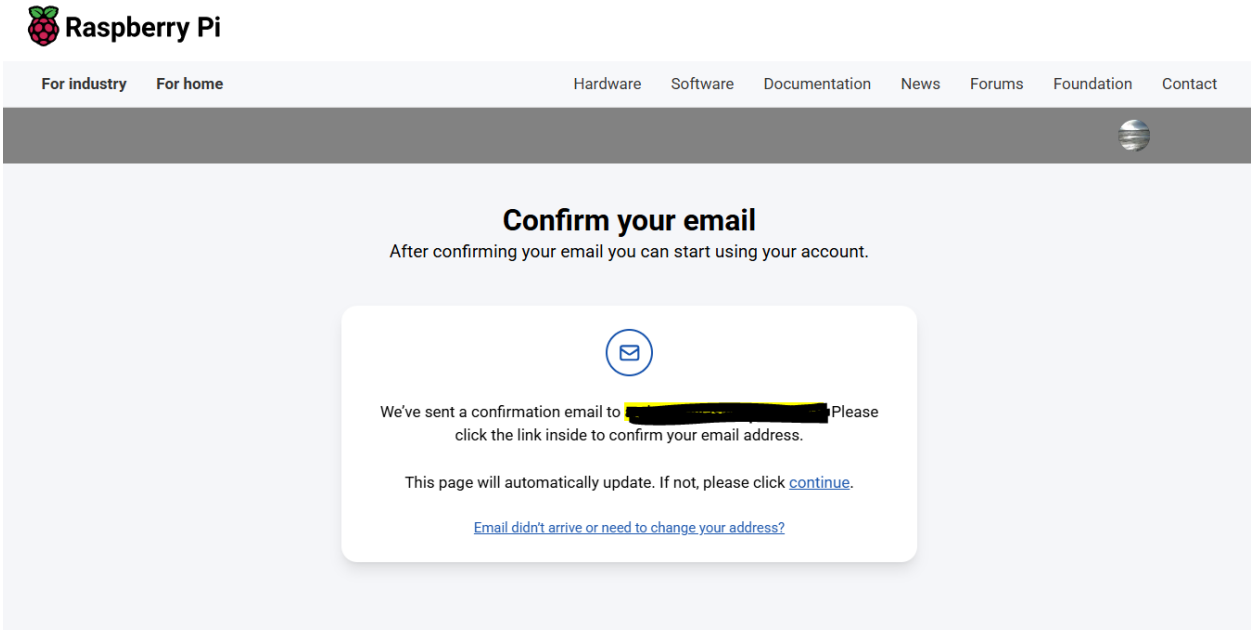


➔ Si vous ne possédez pas de compte Raspberry Pi, créez-en un.

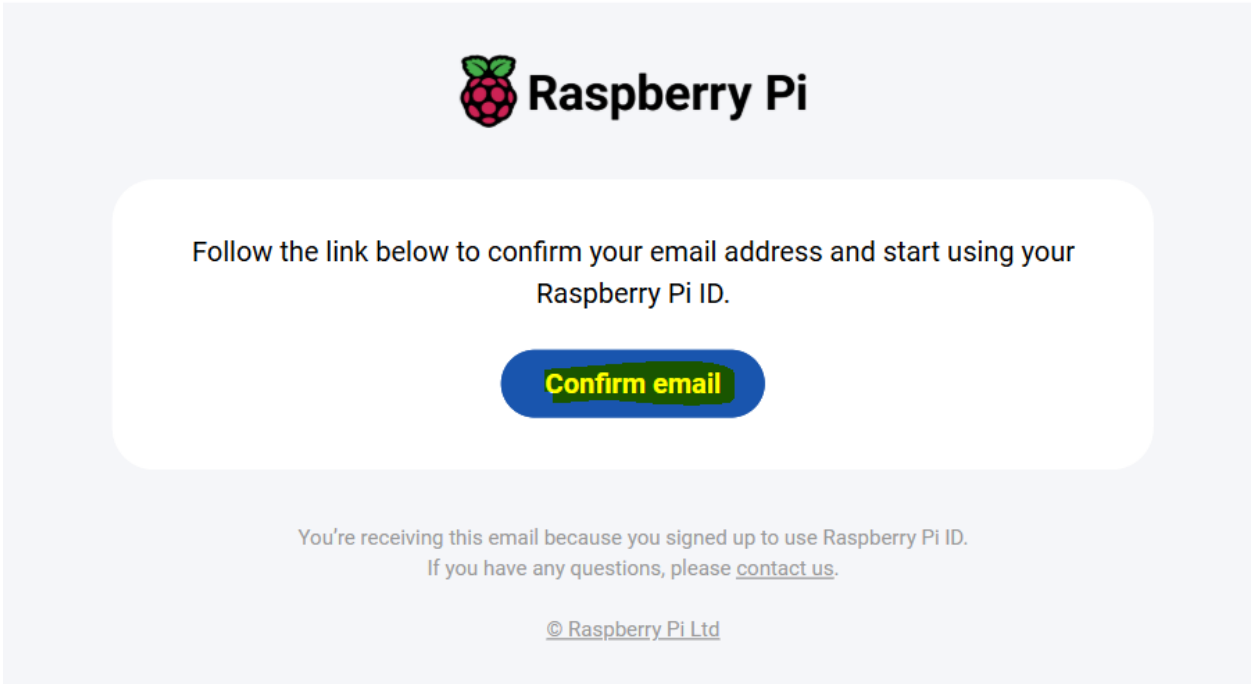




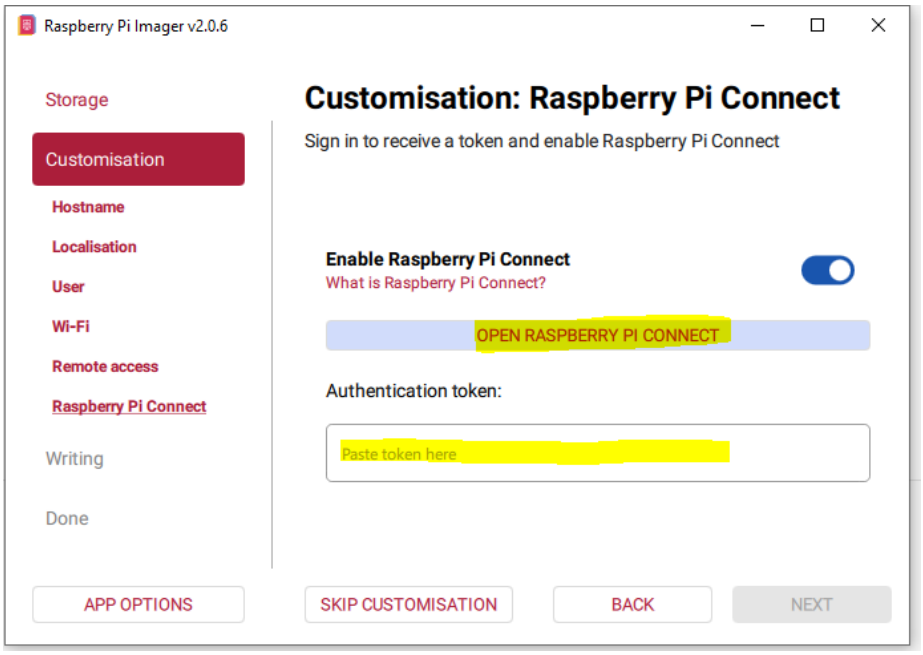
➔ Après la création de votre compte, un courriel de confirmation sera envoyé à l'adresse électronique fournie.

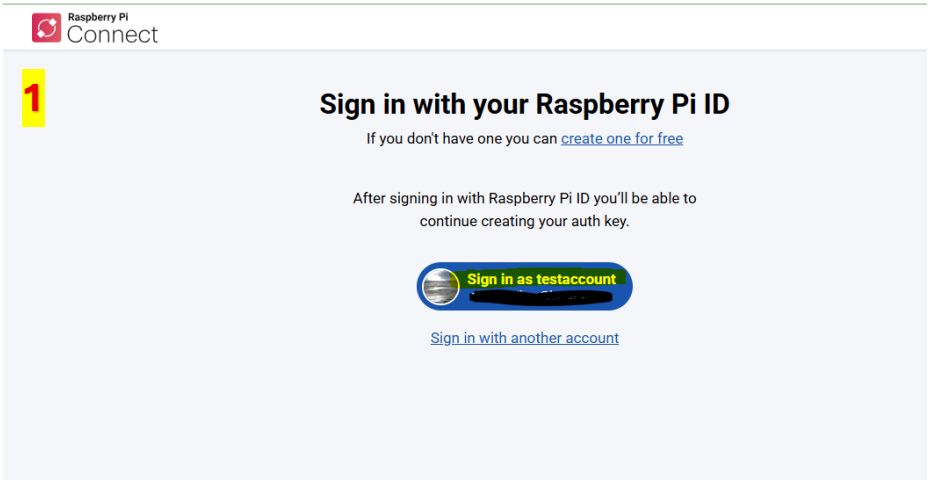


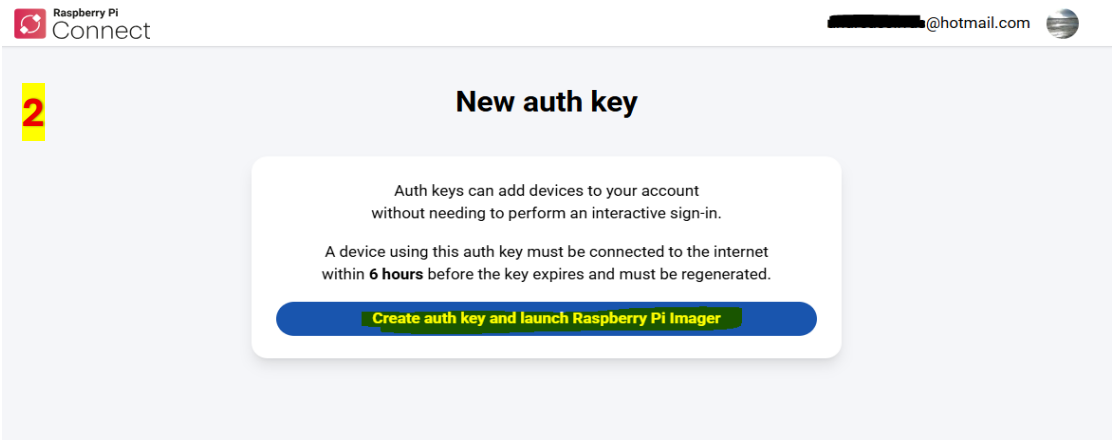
➔ Cliquez sur le lien reçu par courriel afin de confirmer votre compte.



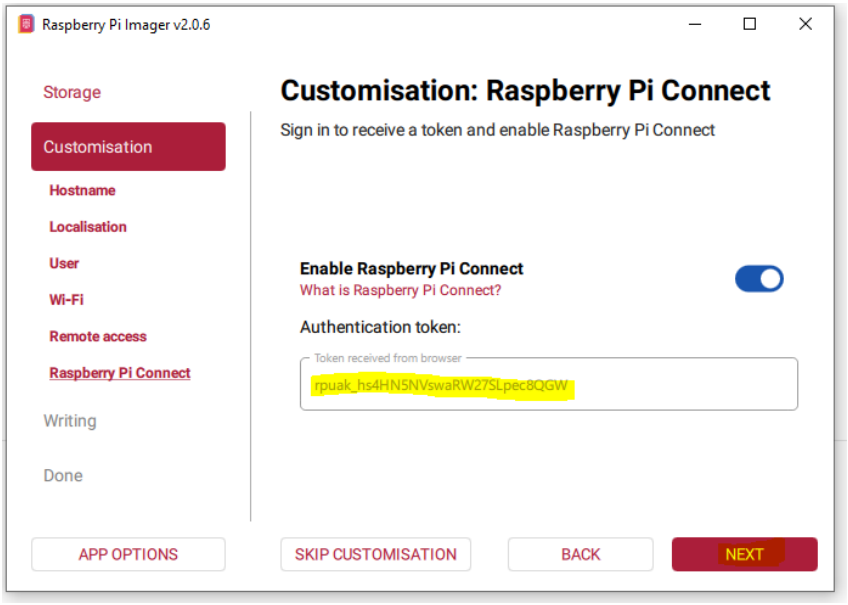
➔ Cliquez à nouveau sur **OPEN RASPBERRY PI CONNECT** pour obtenir votre clé d'authentification via Raspberry Pi Connect.



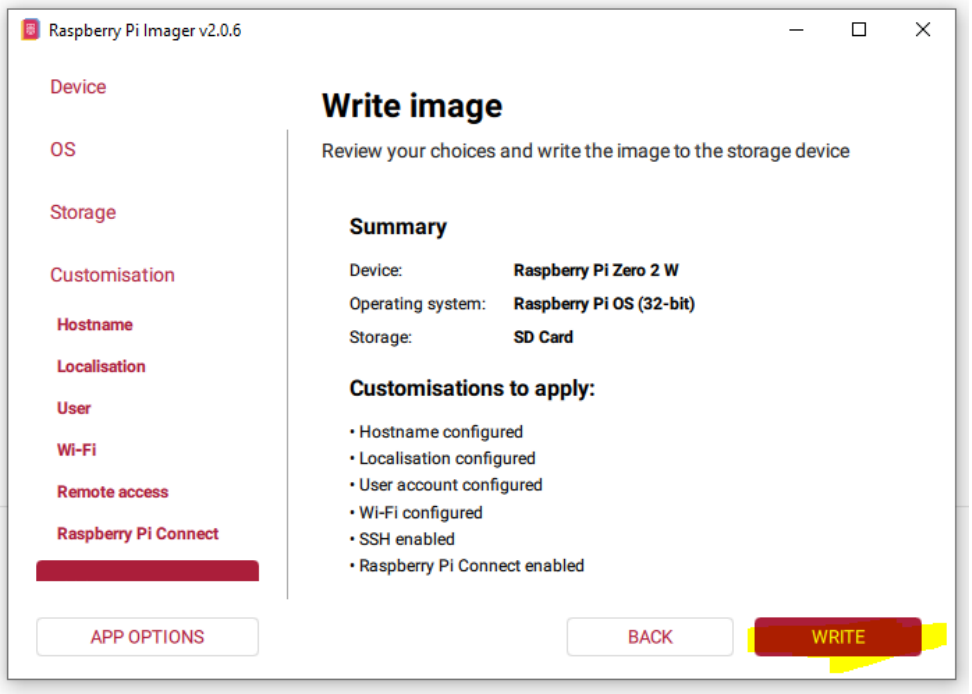




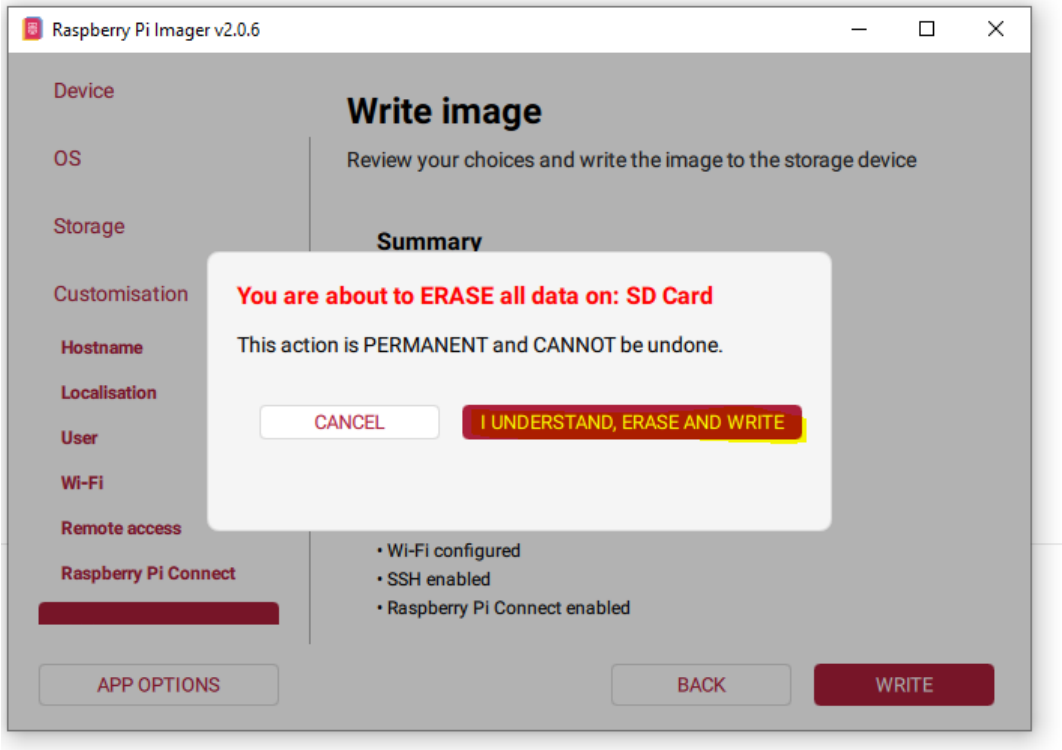
➔ La clé d'authentification apparaît dans le champ **Authentication Token.**



➔ Enregistrer l'image sur la **carte SD**.



➔ Cliquez sur **“I UNDERSTAND, ERASE AND WRITE”**. Le téléchargement démarrera automatiquement. Une fois l’installation terminée, cliquez sur **“Terminer”**.



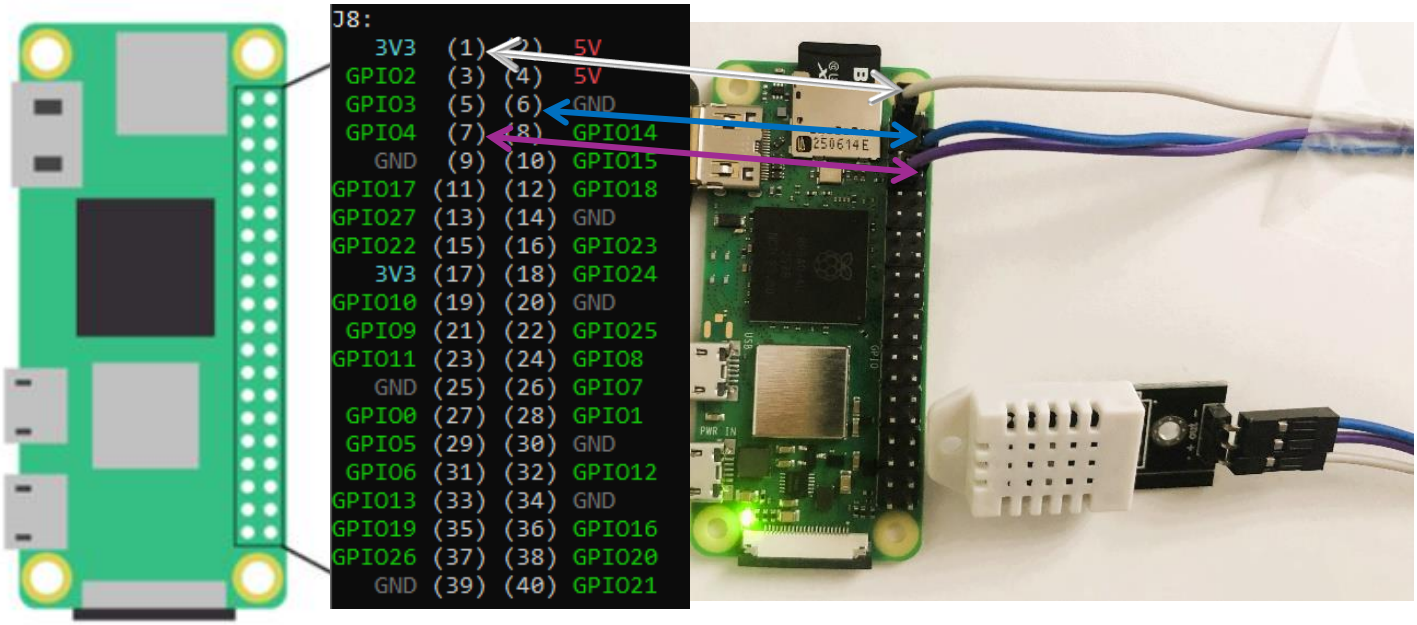
Connexion des broches du capteur DHT22 au Raspberry Pi

Après l'installation et la configuration du système sur le Raspberry Pi, l'étape suivante consiste à connecter correctement le capteur DHT22 aux broches GPIO. Cette connexion permet au Raspberry Pi de lire les données de température et d'humidité transmises par le capteur.

Un câblage correct est essentiel pour assurer un fonctionnement stable et éviter tout problème de lecture ou de communication entre le capteur et le Raspberry Pi.

Connexion physique des broches DHT

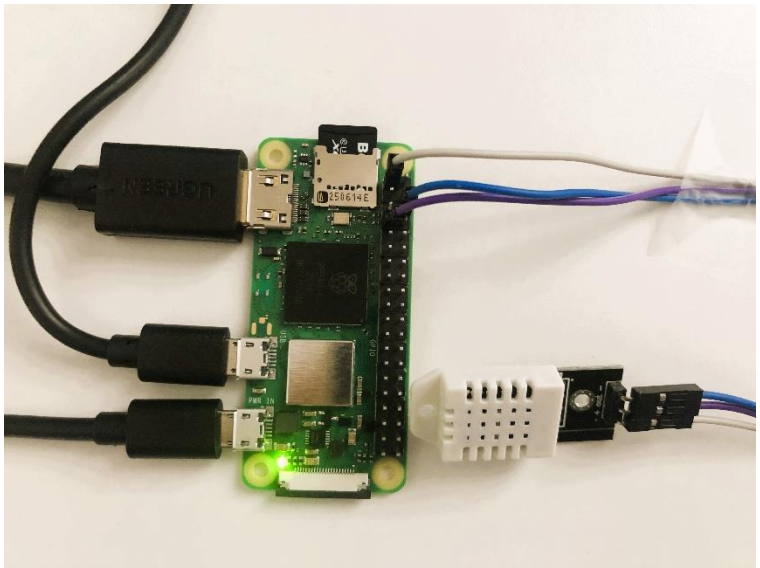
+	Pin 1	3.3V power
out	Pin 7	GPIO4 (data)
-	Pin 6	Ground



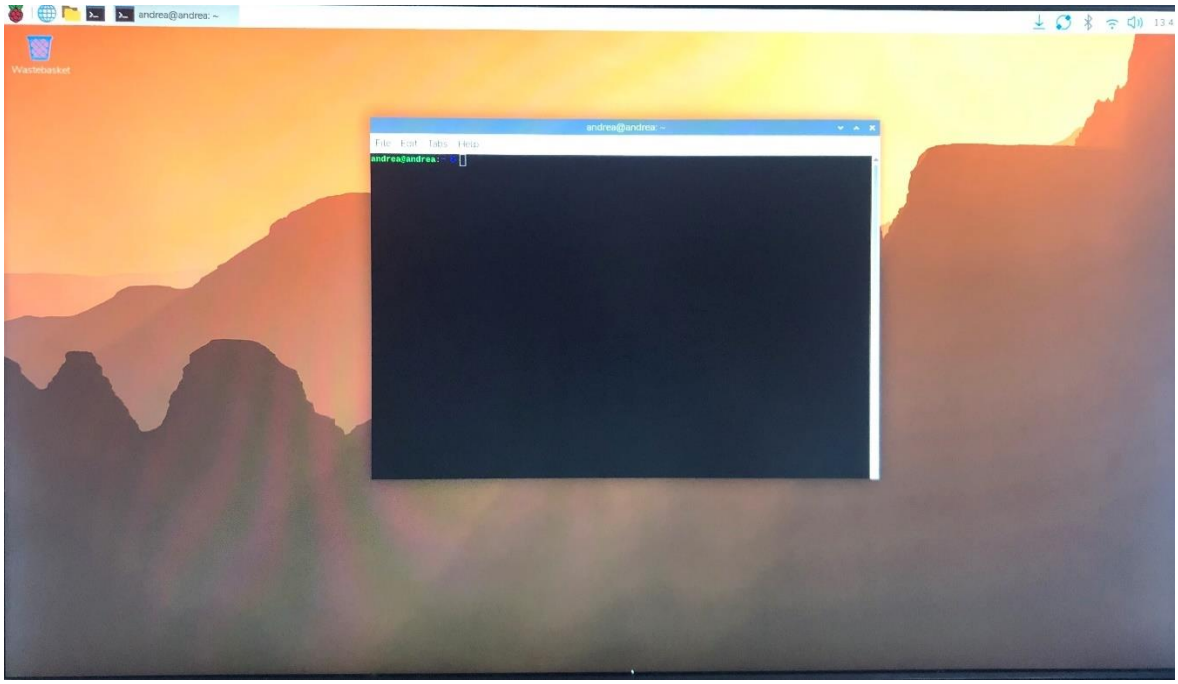
Pin check : **pinout**

```
andrea@andrea:~ $ pinout
Description        : Raspberry Pi Zero2W rev 1.0
Revision           : 902120
SoC                : BCM2837
RAM                : 512MB
Storage            : MicroSD
USB ports          : 1 (of which 0 USB3)
Ethernet ports     : 0 (0Mbps max. speed)
Wi-fi              : True
Bluetooth          : True
Camera ports (CSI) : 1
Display ports (DSI): 0

,--oooooooooooooooooooo---.
|  1oooooooooooooooooooo J8|
+---+      +---+  PiZero2W  c|
| sd|      |SoC|   Wi V1.0  s|
+---+      +---+   Fi        i|
|  hdmi                usb pwr |
`-|  |------------|  |-|  |-'

J8:
   3V3  (1) (2)  5V
 GPIO2  (3) (4)  5V
 GPIO3  (5) (6)  GND
 GPIO4  (7) (8)  GPIO14
   GND  (9) (10) GPIO15
GPIO17 (11) (12) GPIO18
GPIO27 (13) (14) GND
GPIO22 (15) (16) GPIO23
   3V3 (17) (18) GPIO24
GPIO10 (19) (20) GND
 GPIO9 (21) (22) GPIO25
GPIO11 (23) (24) GPIO8
   GND (25) (26) GPIO7
 GPIO0 (27) (28) GPIO1
 GPIO5 (29) (30) GND
 GPIO6 (31) (32) GPIO12
GPIO13 (33) (34) GND
GPIO19 (35) (36) GPIO16
GPIO26 (37) (38) GPIO20
   GND (39) (40) GPIO21
```

➔ Insérez la **carte SD** dans le Raspberry Pi, puis connectez-le à l'alimentation électrique.



➔ L'interface du système d'exploitation se présentera comme suit :



Installation de Python 3

Une fois le Raspberry Pi configuré et le capteur connecté, l'étape suivante consiste à installer Python 3. Python sera utilisé pour écrire le script qui permettra de lire les données du capteur DHT22 et de les envoyer vers Firebase.

Sur Raspberry Pi OS, Python 3 est généralement déjà installé. Cependant, il est important de vérifier sa présence et de s'assurer qu'il est à jour.

Pour vérifier la version installée :

```
python3 –version
```

Pour installer ou de mettre à jour Python 3 à l'aide du gestionnaire de paquets :

```
sudo apt update
sudo apt install python3
```

Cette étape garantit que l'environnement est prêt pour le développement et l'exécution du script de collecte des données.

Installation de la bibliothèque Adafruit pour le DHT22

Pour permettre à Python de communiquer avec le capteur DHT22, il est nécessaire d'installer la bibliothèque adaptée. J'ai utilisé la bibliothèque Adafruit DHT, qui facilite la lecture des données de température et d'humidité via les broches GPIO du Raspberry Pi.

```
sudo apt update
```

Installer des bibliothèques Python, Compiler des modules nécessaires au fonctionnement du capteur et Travailler correctement avec des dépendances matérielles :

```
sudo apt install -y python3-dev python3-pip
```

Lors de l'installation de la bibliothèque Adafruit DHT avec la commande **pip3 install Adafruit_DHT**, j'ai rencontré des problèmes de compilation sur le Raspberry Pi.

Pour résoudre ce problème, j'ai cloné directement la bibliothèque officielle depuis le dépôt GitHub d'Adafruit :

```
clone https://github.com/adafruit/Adafruit_Python_DHT
```

Cette méthode m'a permis d'installer correctement la bibliothèque et d'assurer le bon fonctionnement du capteur DHT22 avec Python.

Test de lecture du capteur DHT22

Vérifiez que la bibliothèque Adafruit DHT22 fonctionne correctement

Créez un fichier nommé dht_test.py à l'aide de la commande suivante **:**

nano dht_test.py

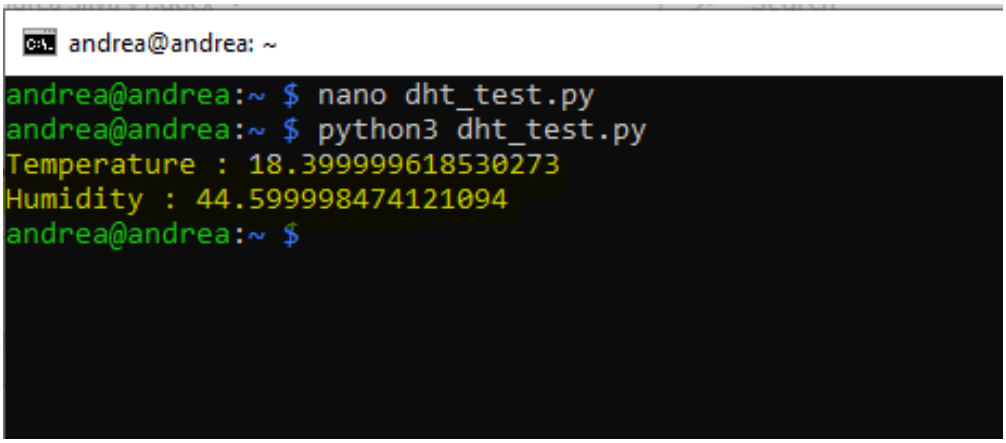
Fichier **dht_test.py** code:

andrea@andrea: ~

```
 GNU nano 8.4                                                    dht_test.py
import Adafruit_DHT

sensorDHT = Adafruit_DHT.DHT22
pin = 4

humidity, temperature = Adafruit_DHT.read_retry(sensorDHT,pin)

if humidity is not None and  temperature is not None:
        print (f"Temperature : {temperature}")
        print (f"Humidity : {humidity}")
else:
        print ('Failed to read')
```

Exécutez-le avec la commande suivante :

python3 dht_test.py



➔ Créez un script permettant d'enregistrer la température et l'humidité toutes les 5 secondes.

Fichier **dht_loop.py** code:

Select andrea@andrea: ~

GNU nano 8.4 dht_loop.py *

```
import time
import Adafruit_DHT

sensorDHT = Adafruit_DHT.DHT22
pin = 4

while True:
        humidity, temperature = Adafruit_DHT.read_retry(sensorDHT,pin)

        if humidity is not None and temperature is not None:
                print (f"Temperature: {temperature:.1f} Humidity: {humidity:.1f}")
        else:
                print ("Failed to read from DHT22 sensor")

        time.sleep(5) #read every 5 seconds
```

Exécutez-le avec la commande suivante :

python3 dht_loop.py

andrea@andrea: ~

```
andrea@andrea:~ $ python3 dht_loop.py
Temperature: 18.6 Humidity: 44.9
Temperature: 18.4 Humidity: 45.3
Temperature: 18.4 Humidity: 45.4
```

Configuration de Firebase

Après avoir configuré le Raspberry Pi et mis en place la lecture du capteur, l'étape suivante consiste à configurer Firebase pour stocker les données dans le cloud.

J'ai créé un projet sur Firebase, puis activé Realtime Database afin d'enregistrer les mesures de température et d'humidité. Pour l'envoi des données depuis le Raspberry Pi, j'ai utilisé l'API REST de Firebase. Cette méthode me permet de transmettre les données directement depuis le script Python via des requêtes HTTP.

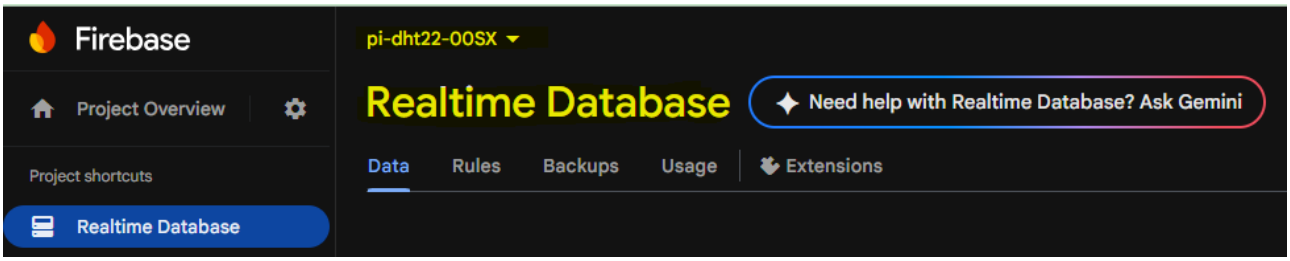
Firebase assure ainsi le stockage sécurisé des données et leur synchronisation en temps réel avec l'application web développée en React.

Création d'un nouveau projet dans Firebase :

1. Cliquez sur **Ajouter un projet**.
2. Entrez le nom du projet : **pi-dht22-00SX**.
3. Désactivez **Google Analytics** (non nécessaire pour ce projet).
4. Cliquez sur **Créer le projet**.

Activation de Realtime Database :

1. Dans le menu de gauche, sélectionnez **Build → Realtime Database**.
2. Cliquez sur **Créer une base de données**.
3. Choisissez une région **(la configuration par défaut convient)**.
4. Sélectionnez **Démarrer en mode test**.
5. Cliquez sur **Activer (Enable)**.



Test d'envoi des données vers Firebase

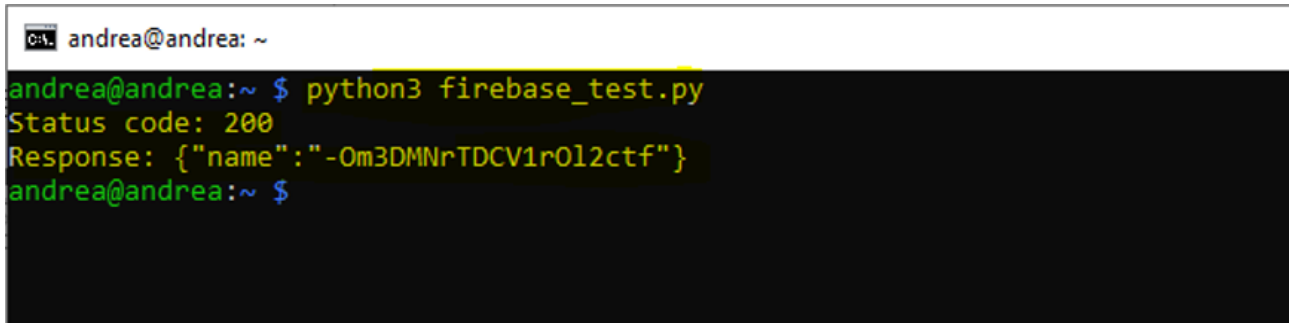
Créez un fichier nommé **firebase_test.py** à l'aide de la commande suivante :

```
nano firebase_test.py
```

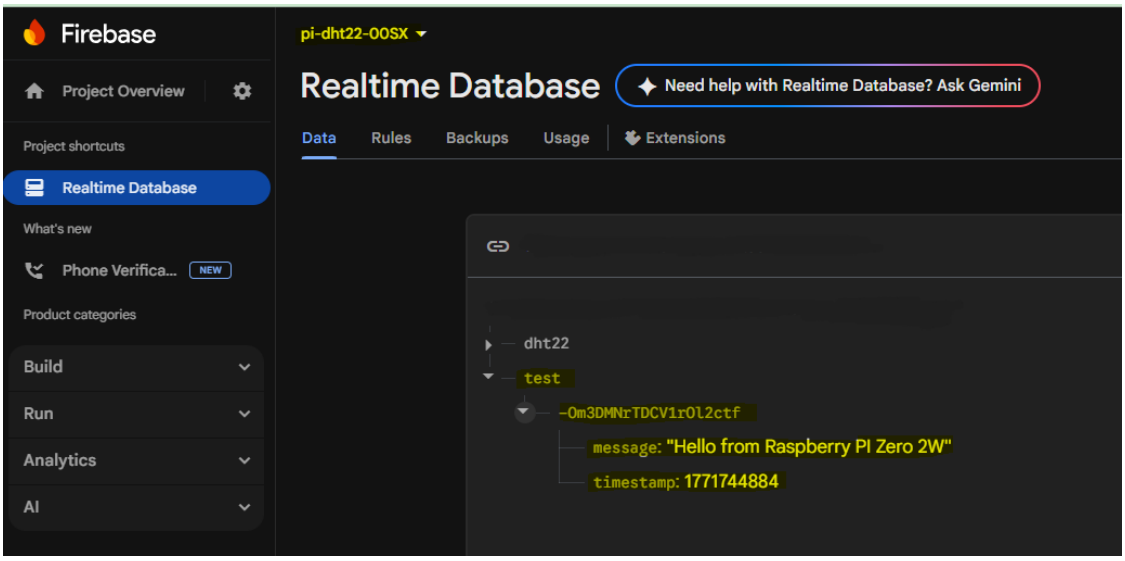
Fichier **firebase_test.py** code:

Exécutez-le avec la commande suivante :

```
python3 firebase_test.py
```



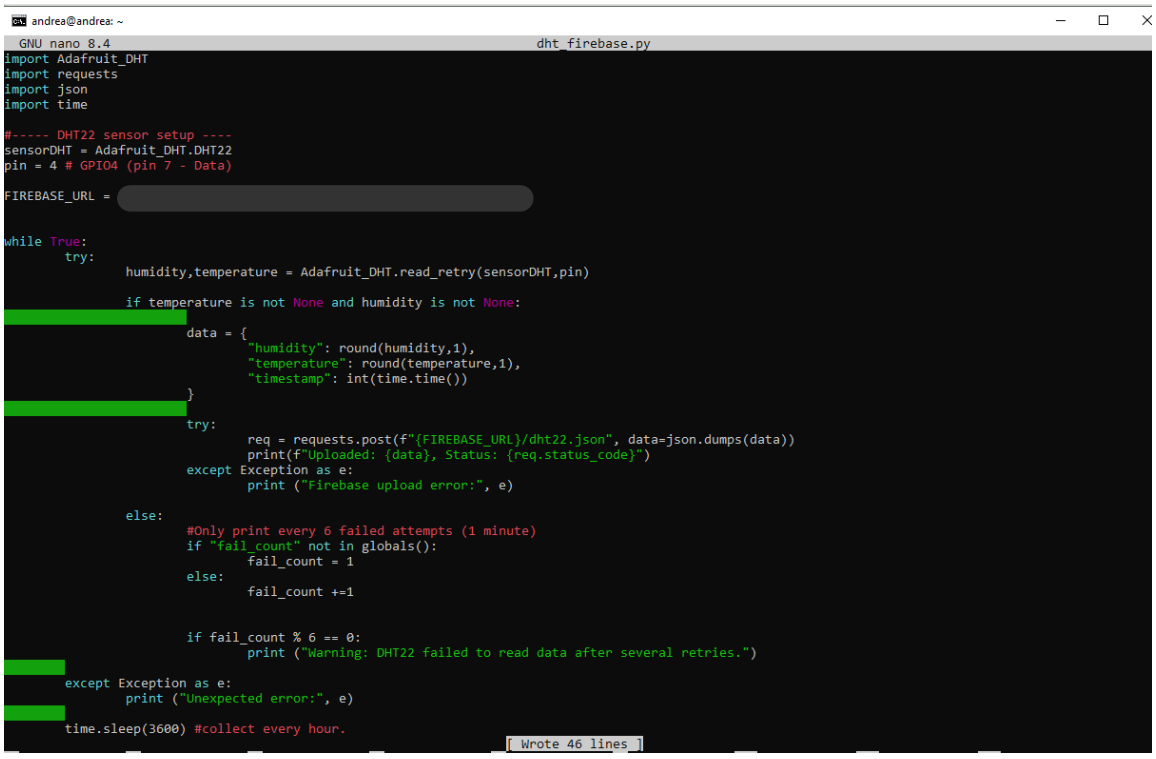
➔ Les données ont été correctement reçues et enregistrées dans Firebase.



L'étape suivante consiste à créer un script Python pour le capteur DHT22 destiné à la production, qui enverra les données à Firebase

Créez un fichier nommé **dht_firebase.py** à l'aide de la commande suivante **:**

```
nano dht_firebase.py
```

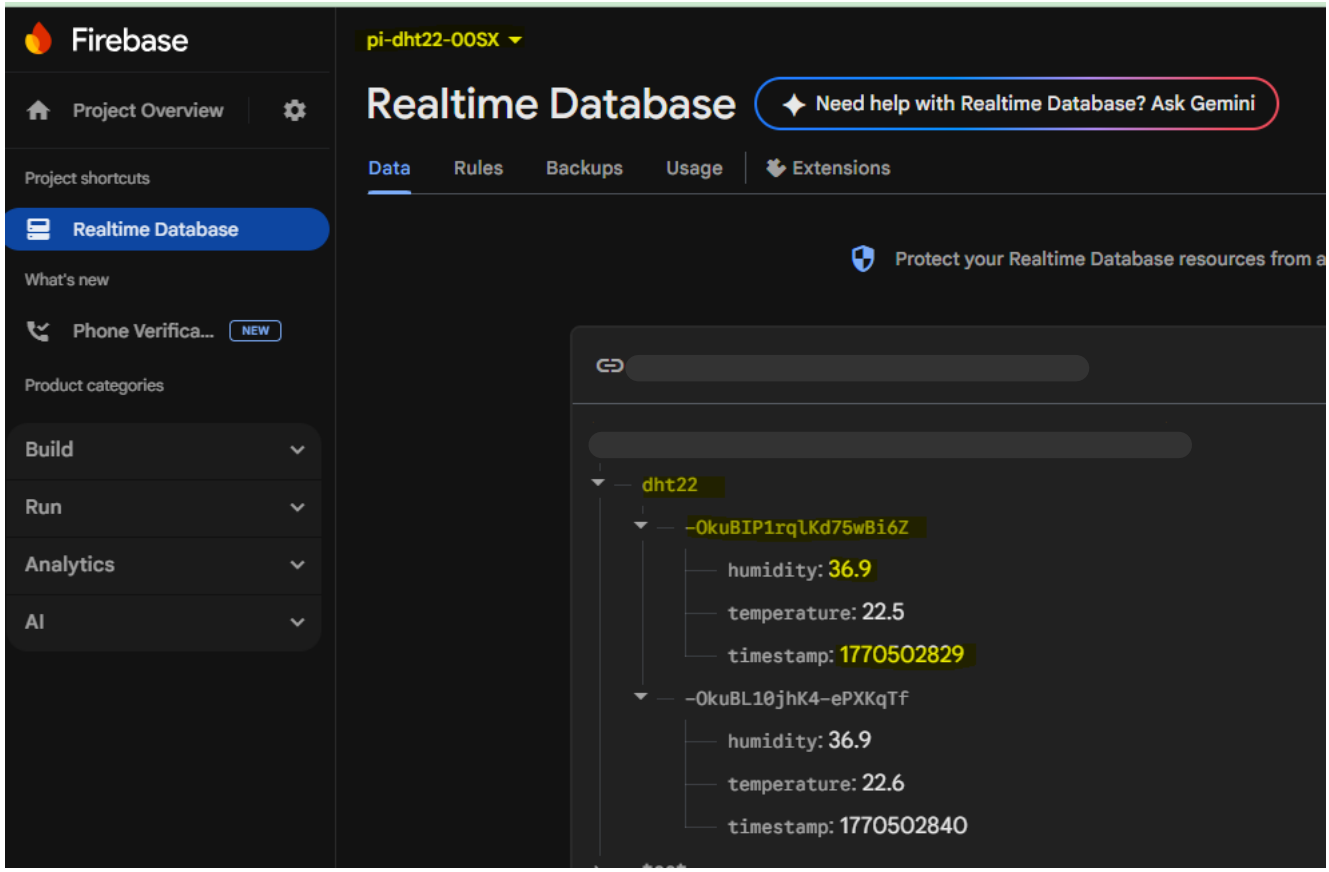


Exécutez-le avec la commande suivante :

```
python3 dht_firebase.py
```

```
andrea@andrea:~ $ python3 dht_firebase.py
Uploaded: {'temperature': 22.6, 'humidity': 34.2, 'timestamp': 1770502816}, Status: 200
Uploaded: {'temperature': 22.5, 'humidity': 36.9, 'timestamp': 1770502829}, Status: 200
```

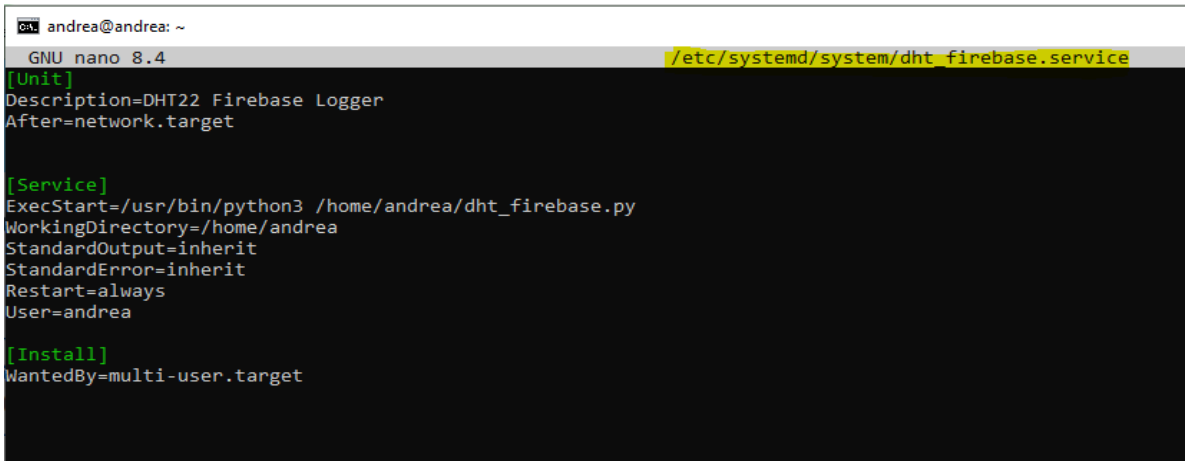
➔ Les données sont **stockées dans Firebase**.



Exécutez le script dht_firebase.py au démarrage à l'aide des services

Créer un fichier de service systemd :

```
sudo nano /etc/systemd/system/dht_firebase.service
```



Activez et démarrez le service :

```
sudo systemctl daemon-reload
```

```
sudo systemctl enable dht_firebase.service
```

```
sudo systemctl start dht_firebase.service
```

Vérifier l'état du service :

```
sudo systemctl status dht_firebase.service
```

Arrêter le service (temporairement)

```
sudo systemctl stop dht_firebase.service
```

Désactivez son démarrage au lancement du système (permanent jusqu'à ce qu'il soit réactivé)

```
sudo systemctl disable dht_firebase.service
```

```
sudo systemctl stop dht_firebase.service && sudo systemctl disable dht_firebase.service
```

Redémarrer automatiquement

```
sudo systemctl enable --now dht_firebase.service
```

Configuration SSH

Trouvez l'interface réseau IP réelle :

`ip addr show`

Assurez-vous que SSH est activé.

`sudo systemctl status ssh`

```
andrea@andrea:~ $ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/usr/lib/systemd/system/ssh.service; enabled; preset: enabled)
     Active: active (running) since Wed 2026-02-18 18:56:09 EST; 6 days ago
 Invocation: 4293418d2fc1497a9f5c0c4c26d82e10
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 1114 (sshd)
      Tasks: 1 (limit: 359)
        CPU: 2.499s
     CGroup: /system.slice/ssh.service
             └─1114 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"
```

Utilisez CMD en tant qu'administrateur sur votre ordinateur portable personnel pour vous connecter via SSH.

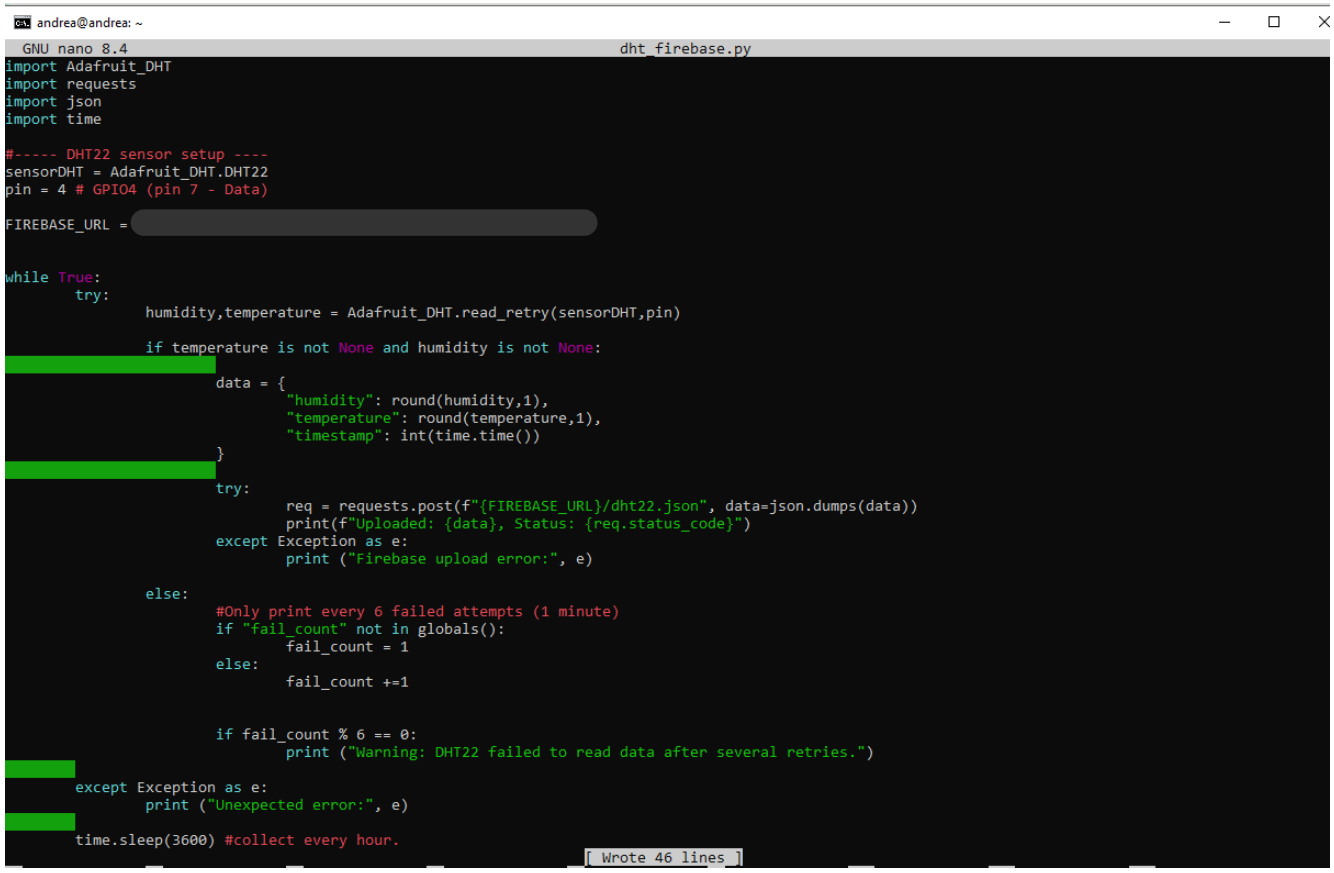
```
andrea@andrea: ~
Microsoft Windows [Version 10.0.19045.6466]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>ssh andrea@1
The authenticity of host '                    ' can't be established.
ED25519 key fingerprint is SHA256:kBlHM9zOVAcfn0W0McXP0n6wjuWLK3UWx4D3mkrBnTc.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '                    ' to the list of known hosts.
andrea@            password:
Linux andrea 6.12.47+rpt-rpi-v7 #1 SMP Raspbian 1:6.12.47-1+rpt1 (2025-09-16) armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
andrea@andrea:~ $
```

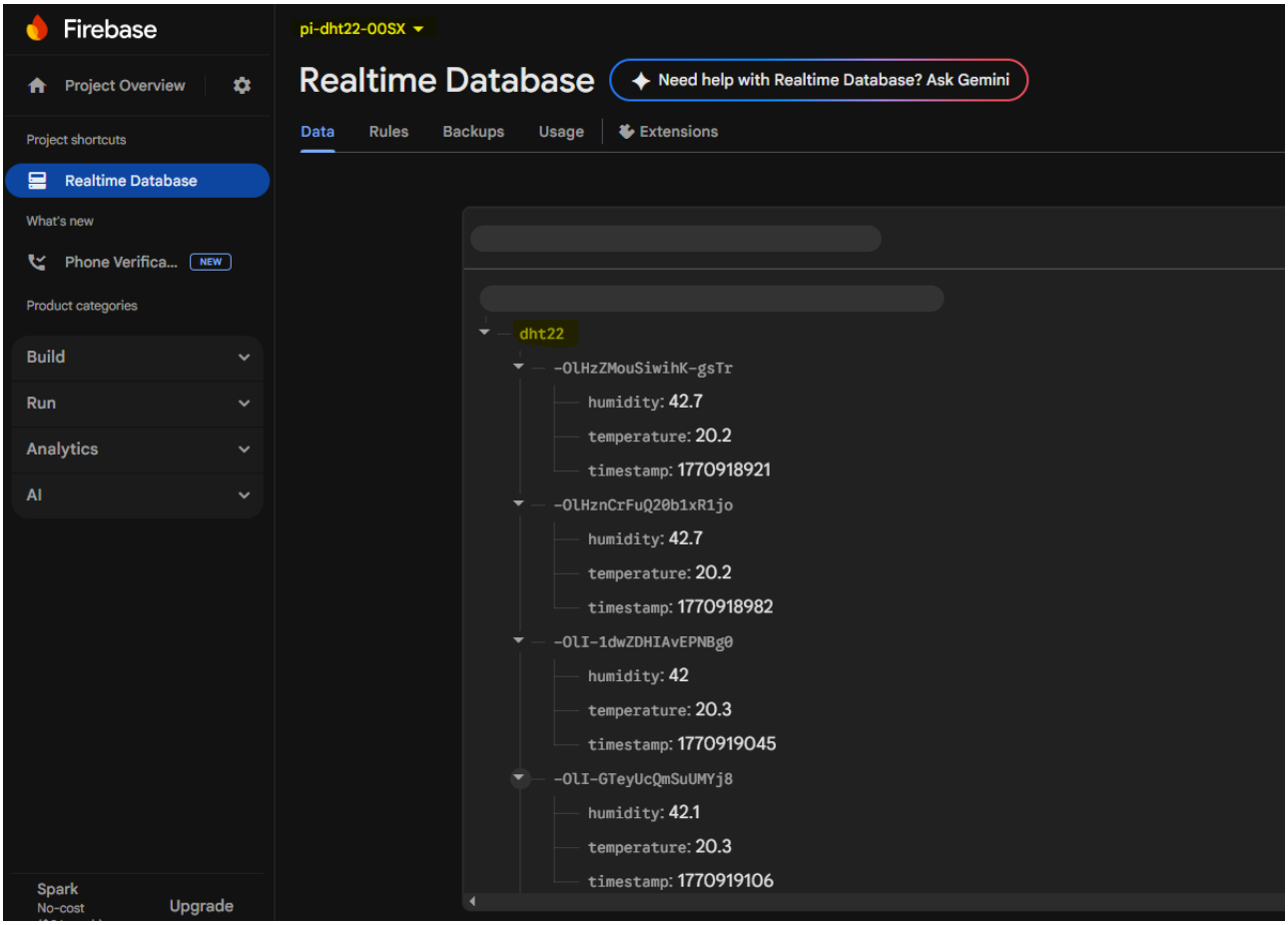
VIII - Développement

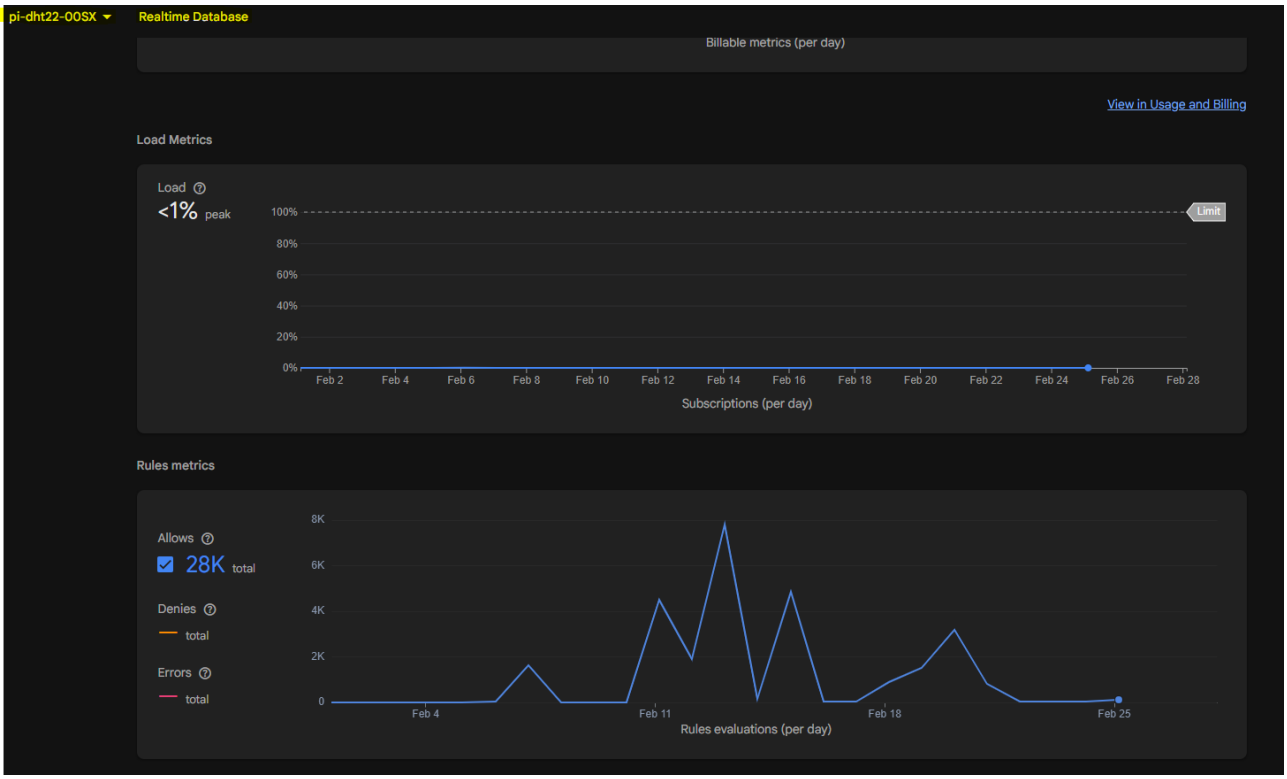
Développement backend

Le “backend” est assuré par le script Python qui tourne sur le Raspberry Pi. Il s’occupe de :

- Lire les données
- Les formater
- Les envoyer vers Firebase



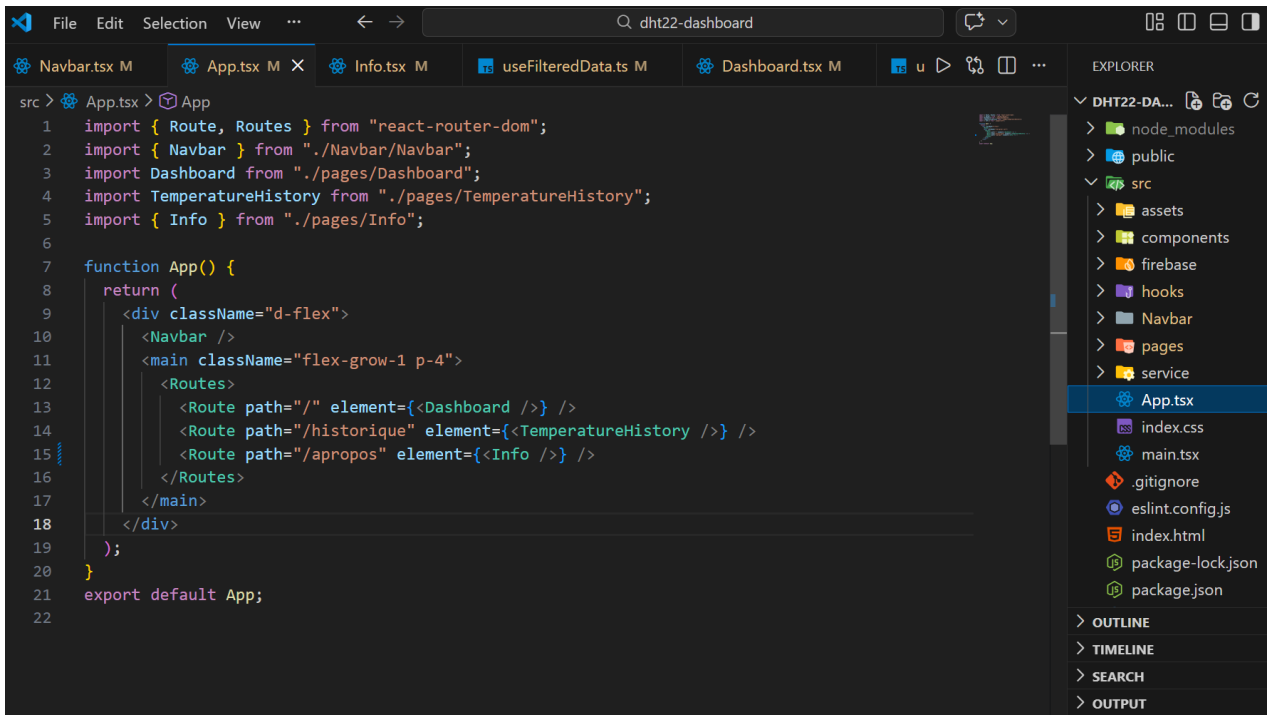

Firebase
pi-dht22-00SX
Project Overview
Realtime Database
Need help with Realtime Database? Ask Gemini
Project shortcuts
Data
Rules
Backups
Usage
Extensions
Realtime Database
What's new
Phone Verifica...
NEW
Product categories
Build
Run
Analytics
AI
dht22
-OlHzZMouSiwihK-gsTr
humidity: 42.7
temperature: 20.2
timestamp: 1770918921
-OlHznCrFuQ20b1xR1jo
humidity: 42.7
temperature: 20.2
timestamp: 1770918982
-OlI-1dwZDHIAvEPNBg0
humidity: 42
temperature: 20.3
timestamp: 1770919045
-OlI-GTeyUcQmSuUMYj8
humidity: 42.1
temperature: 20.3
timestamp: 1770919106
Spark
No-cost
Upgrade

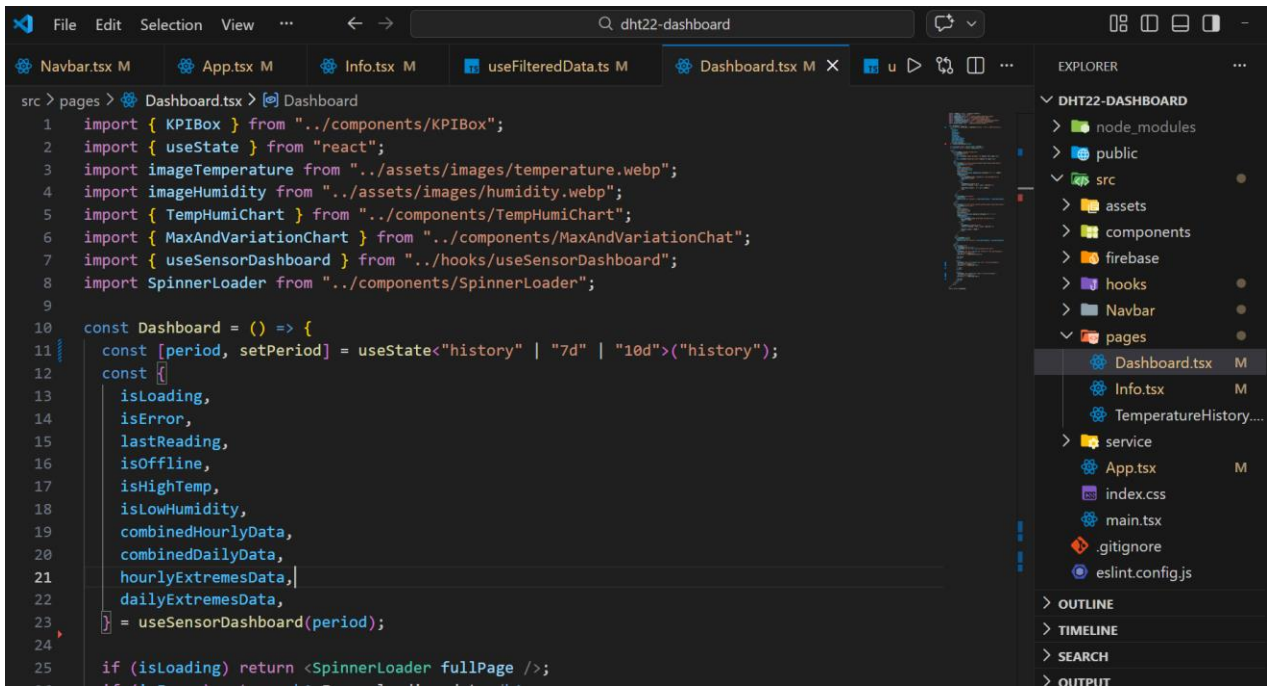

pi-dht22-00SX
Realtime Database
Billable metrics (per day)
View in Usage and Billing
Load Metrics
Load
<1% peak
100%
80%
60%
40%
20%
0%
Limit
Feb 2
Feb 4
Feb 6
Feb 8
Feb 10
Feb 12
Feb 14
Feb 16
Feb 18
Feb 20
Feb 22
Feb 24
Feb 26
Feb 28
Subscriptions (per day)
Rules metrics
Allows
28K total
Denies
total
Errors
total
8K
6K
4K
2K
0
Feb 4
Feb 11
Feb 18
Feb 25
Rules evaluations (per day)

Développement du frontend

Le frontend a été développé avec React et TypeScript.

Bootstrap a été utilisé pour rendre l'interface propre et responsive.

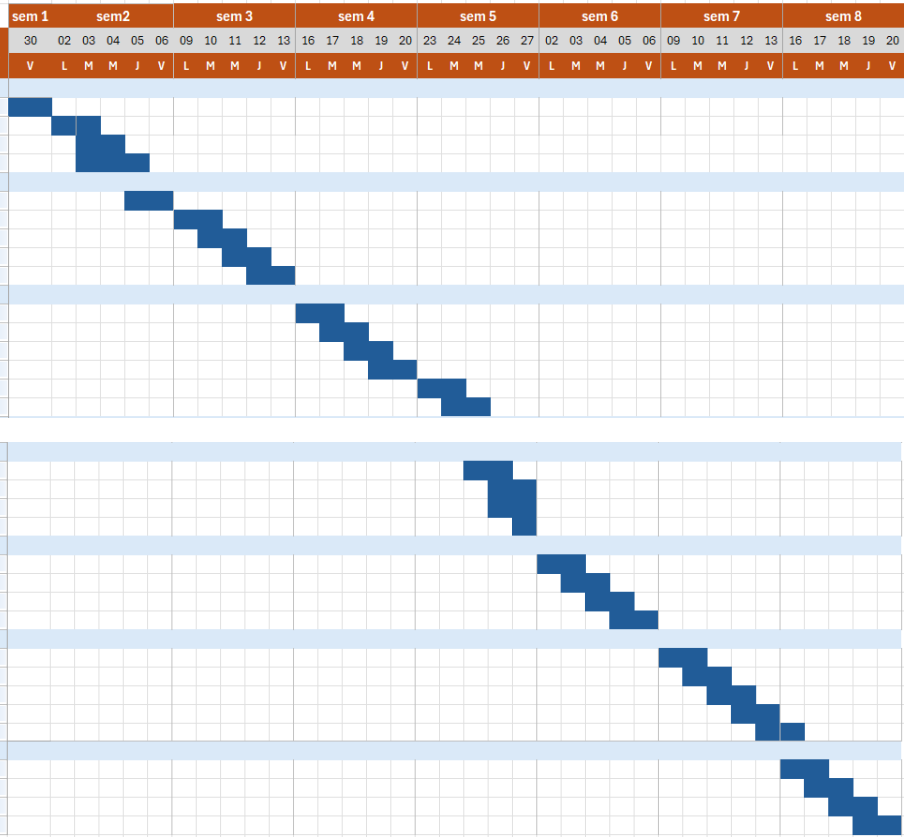




IX – Planification

Nom du Project	Lecture du capteur DHT22 - Raspberry Pi
Responsable du projet	Andrea Silva
Date de début du projet	1/30/2026
Date actuelle	2/25/2026
Date d'échéance	3/20/2026

Tâche ID	Tâche	Jour de début	Jour de fin	Durée en jours	Progrès
1	**Analyse des besoins**			**4**	
1.1	Identifier les parties prenantes	1/30/2026	1/30/2026		100%
1.2	Définir les exigences fonctionnelles	2/2/2026	2/3/2026		100%
1.3	Définir les exigences non fonctionnelles	2/3/2026	2/4/2026		100%
1.4	Délimiter la portée du projet	2/4/2026	2/5/2026		100%
2	**Phase de conception**			**5**	
2.1	Élaboration du diagramme de cas d'utilisation	2/5/2026	2/6/2026		100%
2.2	Création du diagramme de séquence	2/9/2026	2/10/2026		100%
2.3	Conception de l'architecture logique	2/10/2026	2/11/2026		100%
2.4	Définition de la structure des données Firebase	2/11/2026	2/12/2026		100%
2.5	Conception de la maquette du dashboard	2/12/2026	2/13/2026		100%
3	**Phase de préparation de l'environnement**			**6**	
3.1	Installation de Raspberry Pi OS	2/16/2026	2/17/2026		100%
3.2	Configuration SSH	2/17/2026	2/18/2026		100%
3.3	Installation de Python 3	2/18/2026	2/19/2026		100%
3.4	Installation des bibliothèques (Adafruit DHT, requests)	2/19/2026	2/20/2026		100%
3.5	Création du projet Firebase	2/23/2026	2/24/2026		100%
3.6	Configuration de React + TypeScript	2/24/2026	2/25/2026		100%
4	**Phase de développement - Développement côté Raspberry Pi**			**3**	
4.1	Lecture du capteur DHT22	2/25/2026	2/26/2026		100%
4.2	Gestion des erreurs	2/26/2026	2/27/2026		100%
4.3	Envoi des données via API REST	2/26/2026	2/27/2026		100%
4.4	Automatisation toutes les 60 minutes	2/27/2026	2/27/2026		100%
5	**Phase de développement - Développement du Frontend**			**4**	
5.1	Création du dashboard	3/2/2026	3/3/2026		100%
5.2	Intégration Firebase	3/3/2026	3/4/2026		100%
5.3	Affichage temps réel	3/4/2026	3/5/2026		100%
5.4	Responsive design avec Bootstrap	3/5/2026	3/6/2026		100%
6	**Phase de tests**			**5**	
6.1	Test de lecture du capteur	3/9/2026	3/10/2026		100%
6.2	Test d'envoi vers Firebase	3/10/2026	3/11/2026		100%
6.3	Test d'affichage en temps réel	3/11/2026	3/12/2026		100%
6.4	Test de redémarrage système	3/12/2026	3/13/2026		100%
6.5	Test de coupure réseau	3/13/2026	3/16/2026		100%
7	**Phase de déploiement**			**4**	
7.1	Configuration démarrage automatique du script	3/16/2026	3/17/2026		100%
7.2	Déploiement application React	3/17/2026	3/18/2026		100%
7.3	Sécurisation des règles Firebase	3/18/2026	3/19/2026		100%
7.4	Tests finaux	3/19/2026	3/20/2026		100%



X - Plan de déploiement

Le déploiement du système se fait en deux parties :
Le déploiement du module IoT (Raspberry Pi) et le déploiement de l'application web.

Déploiement du module IoT (Raspberry Pi)

Les étapes suivantes sont réalisées :

- Installation de Raspberry Pi OS
- Configuration réseau et activation SSH
- Connexion et câblage du capteur DHT22
- Installation de Python 3 et des bibliothèques nécessaires
- Configuration du script Python
- Mise en place de l'exécution automatique au démarrage

Déploiement de l'application React sur Firebase Hosting

Le tableau de bord développé avec React est déployé sur Firebase Hosting afin d'être accessible en ligne.

Le processus de déploiement comprend :

- Génération de la version de production de l'application : **npm run build**
- Initialisation de Firebase dans le projet : **firebase init**
- Sélection du service Hosting et association au projet Firebase existant.
- Déploiement de l'application : **firebase deploy**

Une fois le déploiement terminé, Firebase fournit une URL publique permettant d'accéder au tableau de bord.

L'application est alors synchronisée en temps réel avec Firebase Realtime Database et affiche automatiquement les données envoyées par le Raspberry Pi.