

# VOICEZY

## HUB PILOTÉ PAR COMMANDÉ VOCALE AVEC UN RASPBERRY PI

Le premier hub à commande vocale réalisé à l'IG2I en utilisant un Raspberry Pi.

### À découvrir

- > Présentation générale
- > La commande vocale
- > Les protocoles de communication
- > Les ports GPIO / Capteurs
- > L'utilisation de Firebase
- > L'application en Flutter

# VOICEZY

### TECHNOLOGIES UTILISÉES

```
pi@raspberry:~/voicezy $ ls -lh | grep Languages
-rw-r--r-- 1 pi pi 780K avril 2 15:27 C
-rw-r--r-- 1 pi pi 170K avril 2 14:49 Script Shell
-rw-r--r-- 1 pi pi 050K avril 2 16:06 Python
pi@raspberry:~/voicezy $ use deep speech as vocal
recognition

pi@raspberry:~/voicezy $ echo $GPIO_PORTS_LIB
WiringPi
pi@raspberry:~/voicezy $ export LIGHT_SENSOR=BH1750
pi@raspberry:~/voicezy $ echo $LED_MATRIX_CONTROLLER
MAX7219
pi@raspberry:~/voicezy $ sudo service Wifi|Bluetooth|
|SPI|I2C start
pi@raspberry:~/voicezy $ export TEMP_SENSOR=DHT11
pi@raspberry:~/voicezy $ ./getLinkMeteoAPI
https://openweathermap.org/api
pi@raspberry:~/voicezy $ ls -lh µController
-rw-r--r-- 1 pi pi 142K avril 2 14:49 Arduino Uno
-rw-r--r-- 1 pi pi 142K avril 2 14:49 Raspberry Pi 3B+
pi@raspberry:~/voicezy $ pip install pyrebase
```



## > Présentation générale

Le projet VOICEZY est né dans le cadre de la matière Compilation Croisée & Raspberry Pi en 3ème année à Centrale Lille IG2I. Ce projet est porté par trois élèves : Etienne Perche, Clément Hecquet et Pierre-Louis Demurger. Cette matière est enseignée par M. BOURDEAUD'HUY.

Ce projet consiste en un hub à commande vocale permettant d'effectuer diverses actions tout en utilisant la mallette **JoyPi** fournie.

Les actions sont les suivantes :

- Allumer ou Éteindre une ampoule connectée.
- Demander la météo du jour.
- Demander les informations des capteurs à l'endroit où se situe la JoyPi et les poster en ligne.
- Jouer la musique du jeu vidéo : Mario Bros.

Pour réaliser cela nous avons utilisé le matériel disponible dans la JoyPi (dont un **Raspberry Pi 3B+**), ainsi qu'un **Arduino Uno**.

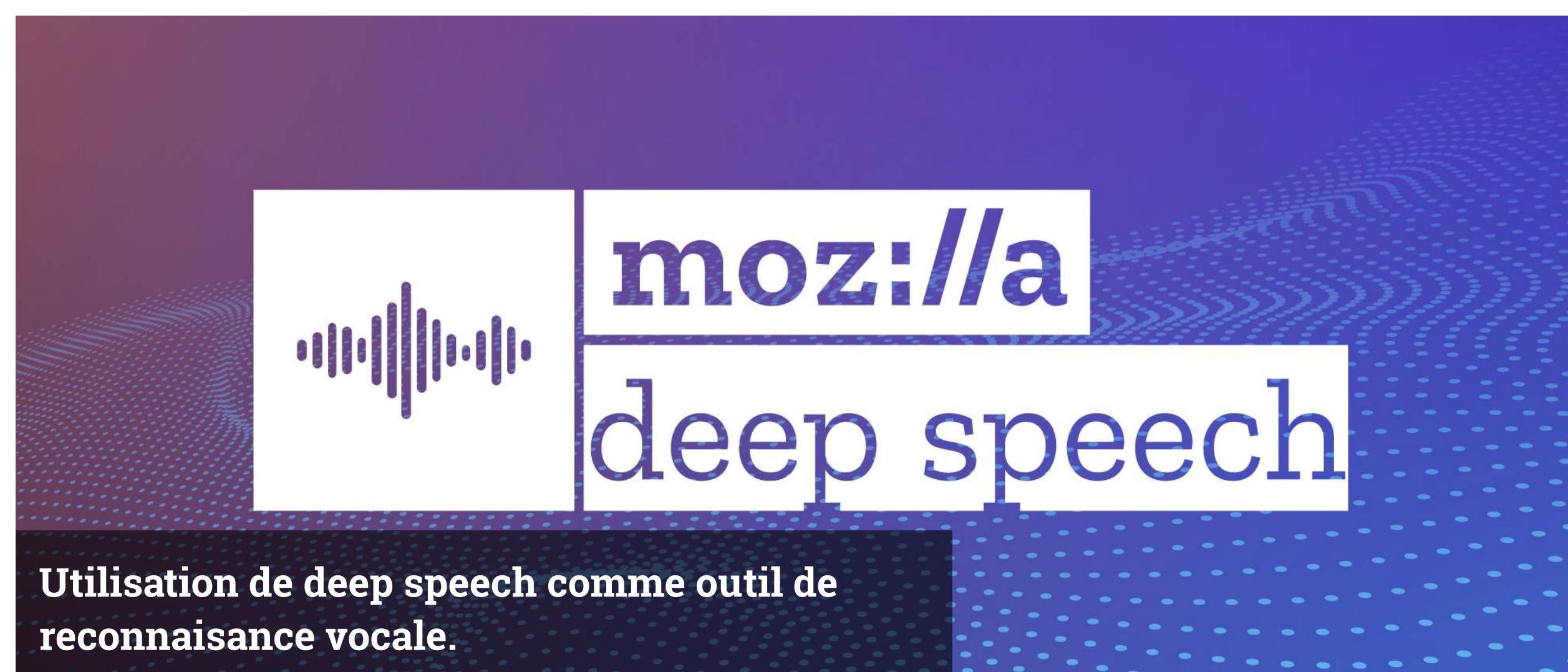
## « La puissance du projet est telle qu'il sera bientôt racheté par le géant : Google »

Ce projet a été développé en **C** et compilé en croisé afin de respecter les attentes de la matière. Le **script shell** et le **Python** ont également été utilisés dans le projet.

Nous allons donc maintenant vous expliquer la démarche suivie lors de la réalisation de ce projet. Nous commençons par la reconnaissance vocale !

En savoir plus sur : <https://github.com/clementh59/voicezy>

## > La commande vocale



Pour effectuer la commande vocale, il faut d'abord enregistrer une commande vocale. C'est pourquoi nous avons mis en place l'enregistrement via le micro lors de l'appui sur un bouton de la JoyPi. En effet cette action déclenche le lancement d'un script Shell permettant l'enregistrement des consignes que vous lui demandez. Celui-ci se coupe lors du second appui sur le bouton d'enregistrement. Au final ceci permet au Raspberry Pi d'obtenir un

fichier audio au format .wav qui sera interprété par la suite grâce à **deepspeech** de Mozilla. Vous devez certainement vous demander pourquoi on a utilisé deepspeech et pas un autre service de reconnaissance vocale parmi le large choix disponible. Nous avons choisi deepspeech car c'est une application qui est bien documentée et facile à implémenter dans notre application à l'aide d'un script Shell. Nous aurions pu partir sur des systèmes de reconnaissance

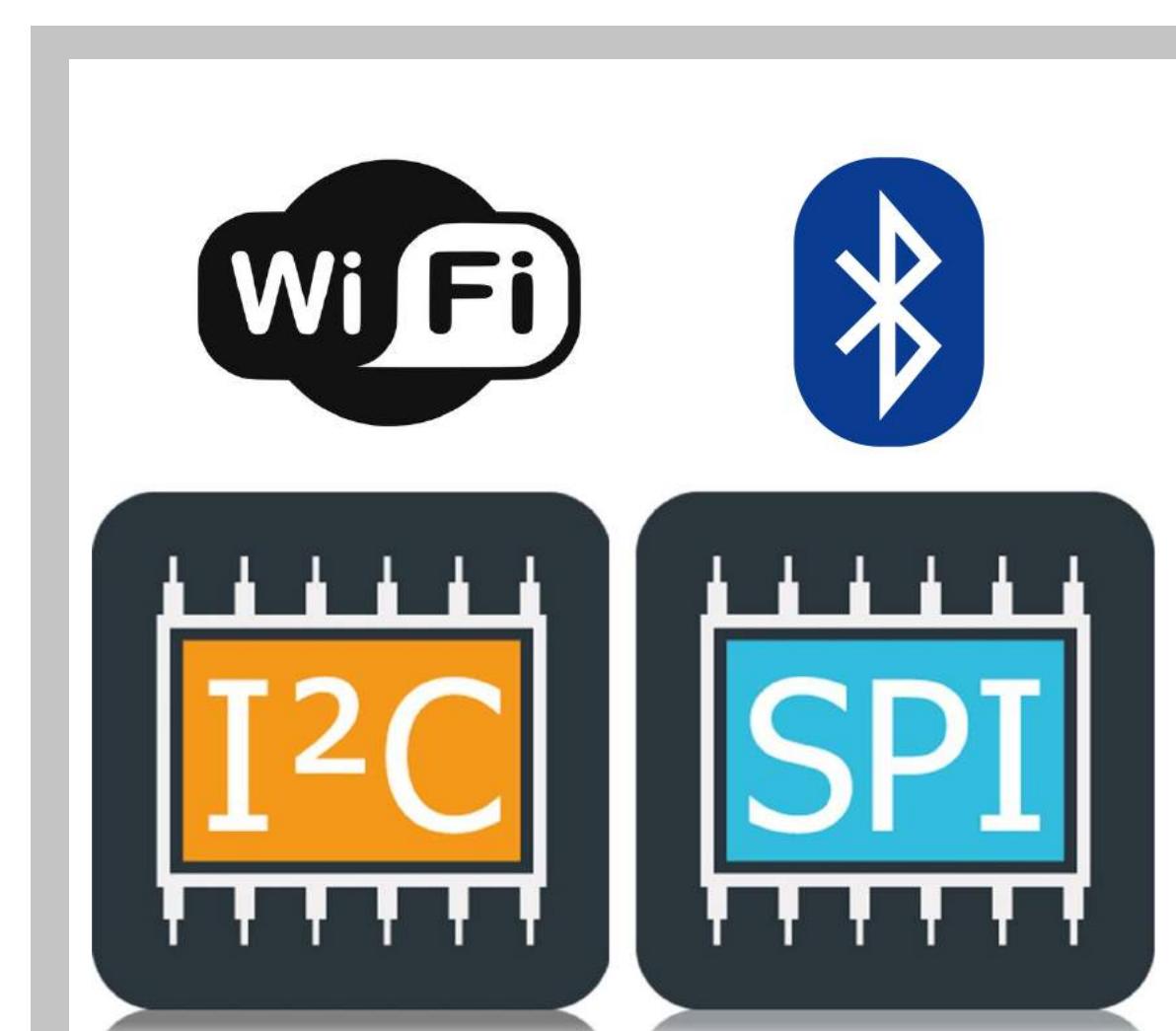
plus performant comme google speech mais cela demandait des coordonnées bancaires et était donc moins accessible. Notre application se commande en anglais pour que celle-ci soit accessible par le plus grand nombre. De plus, les modèles pour faire fonctionner deepspeech en français ne sont pas encore assez performants. Vous devez maintenant vous demander comment nous avons implémenté le service. L'application analyse le fichier audio obtenu précédemment à l'aide du script Shell de deepspeech puis le résultat de la **reconnaissance vocale** de ce script est envoyé dans un fichier texte. Ensuite le fichier texte est ouvert par l'application pour récupérer la chaîne de caractère de la reconnaissance vocale. Après cela le fichier texte est supprimé et le résultat de la

reconnaissance vocale stockée dans la chaîne de caractère est envoyé dans l'interpréteur, celui-ci permet de savoir ce que demande l'utilisateur à l'application (la météo, la température, jouer la musique, ...)

Voilà pour le fonctionnement de la reconnaissance vocale. Le résultat de celle-ci permet ensuite à l'application de lancer les différentes sous applications qui ont été présentées dans la rubrique précédente.

Si vous souhaitez utiliser deepspeech sur votre Raspberry Pi, nous avons mis à disposition un script d'installation sur Github avec un fichier Readme qui permet de comprendre les procédures à faire, celui-ci dans "reconnaissance vocale" du git, ainsi que leur fonction permettant d'utiliser deepspeech dans un programme C.

## > Les protocoles de communication



Dans ce projet, nous utilisons plusieurs protocoles de communication. Parmi eux, on retrouve le WiFi, le Bluetooth, le bus I2C ainsi que le bus SPI.

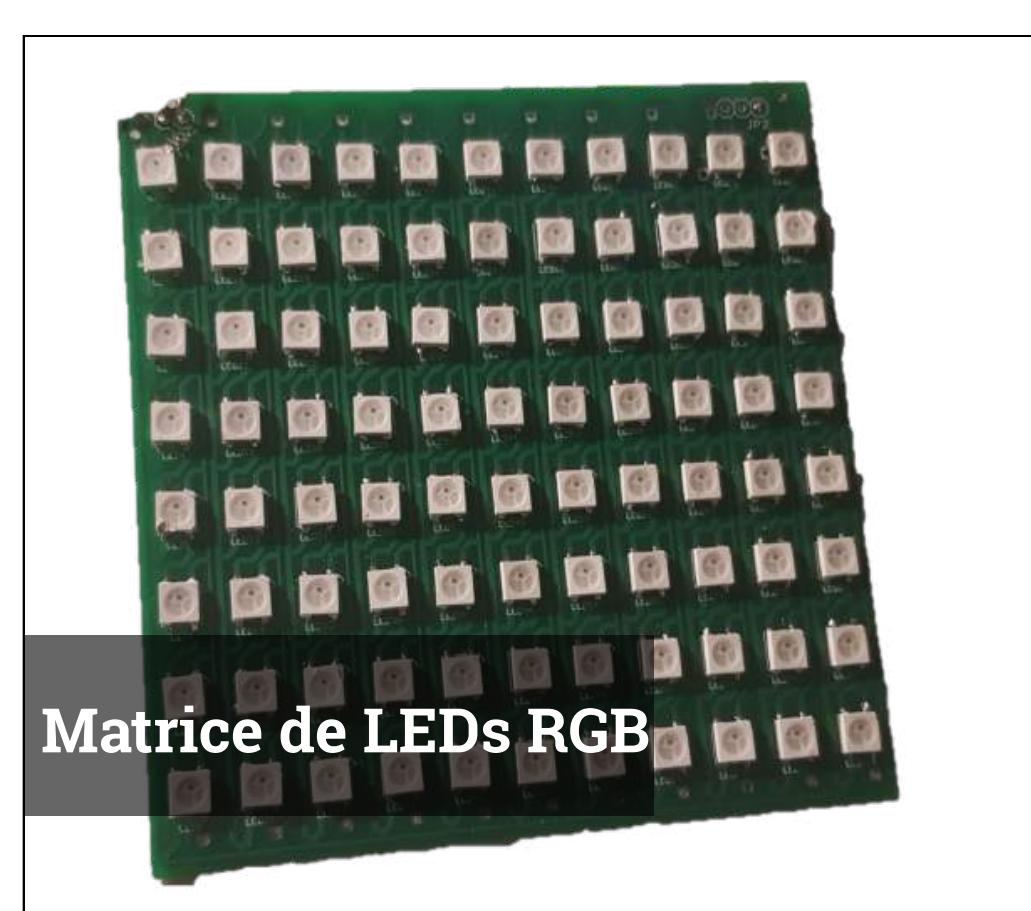
Dans un premier temps, le **WiFi** est utilisé pour plusieurs choses. Il est tout d'abord utile pour communiquer avec l'API OpenWeatherMap, permettant de récupérer la météo du jour. Pour effectuer cette requête, nous avons choisi d'utiliser la librairie **curl**, couplée avec **openSSL**, ce qui nous permet de faire des requêtes **HTTPS** et d'avoir leur réponse. Une fois la requête sur l'API lancée, nous

recevons un **JSON** qui nous donne un tas d'information sur la météo du jour. Nous avons choisi de sélectionner uniquement 2 champs dans ce projet.

Nous récupérons la météo (Soleil, nuage, pluie, ...) et la température.

Ensuite, nous utilisons ces informations pour deux applications distinctes : dans un premier temps, nous affichons sur une matrice de LEDs RGB un dessin correspondant à la météo. Puis nous stockons la température dans une base de données, ce qui nous permettra d'y avoir accès à tout moment, et depuis n'importe quel appareil. (Cela sera détaillé dans une prochaine partie).

Pour afficher des dessins sur une matrice de LEDs RGB nous avons dû faire des choix. En effet, la mallette JoyPi ne dispose pas d'un tel



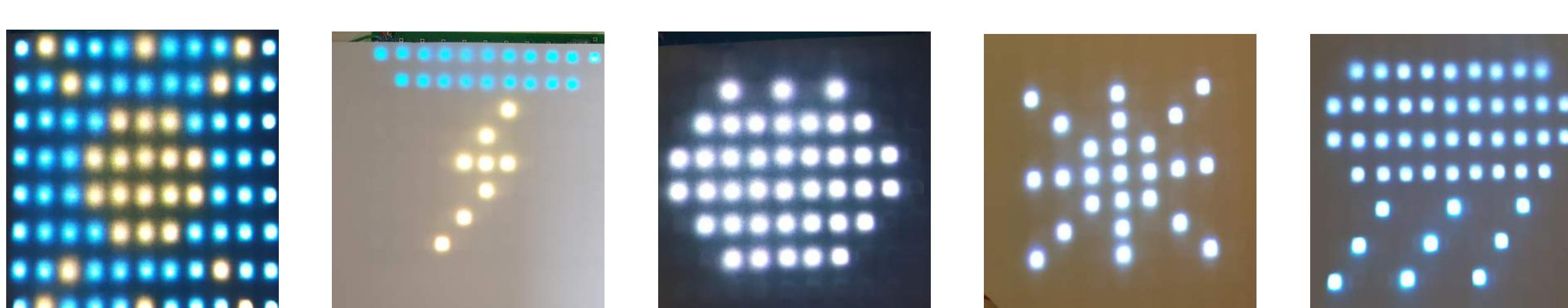
équipement et nous avons donc dû trouver un moyen de s'adapter. Nous avons choisi de relier une matrice de LEDs RGB, que nous avons nous même créée, à un Arduino Uno.

La matrice de LEDs est composée de 88 WS2812B, qui sont des LEDs adressables, ce qui permet de piloter la matrice avec uniquement une pin. L'arduino qui pilote cette matrice communique avec le Raspberry Pi par **Bluetooth**. Nous avons fait face à plusieurs dilemmes lorsque nous avons implémenté la communication Bluetooth. Notamment lors du choix du protocole que nous allions utiliser. Nous avons

finalemment opté pour un protocole qui se rapproche du **TCP**. De plus nous avons choisi que ce soit le Raspberry Pi qui contrôle entièrement la matrice. Vous allez nous dire que nous nous contredis et que quelques lignes au-dessus nous avions dit que c'était l'Arduino qui contrôlait la matrice. Et vous n'avez pas tort! L'Arduino contrôle la matrice puisque c'est lui qui est électriquement connecté à elle. Mais c'est le Raspberry qui choisit la couleur de chaque LED individuellement. En effet, le Raspberry Pi envoie **264** octets à l'Arduino à chaque fois qu'il veut changer l'affichage de la matrice. Vous vous demandez certainement pourquoi 264 octets. Cela correspond à 88 LEDs multipliées par 3 couleurs. En effet, toutes les couleurs peuvent être représentées par une combinaison de rouge, bleu et vert. Dans notre cas, chacune de ces trois couleurs est codée sur 1 octet (entre 0 et 255).

Nous pouvons donc coder toutes les couleurs possibles sur 3 octets. Et nous pouvons donc entièrement piloter la matrice avec 264 octets.

Comme dit précédemment, nous utilisons un protocole se rapprochant du TCP. Pour chaque octet envoyé, nous demandons une confirmation d'octet reçu. Cela permet d'être sûr que l'Arduino reçoit bien toutes les données envoyées par le Raspberry Pi. Cela est certes plus long (le transfert des 264 octets met environ 2 secondes) mais dans l'application, le délai n'est pas un problème.

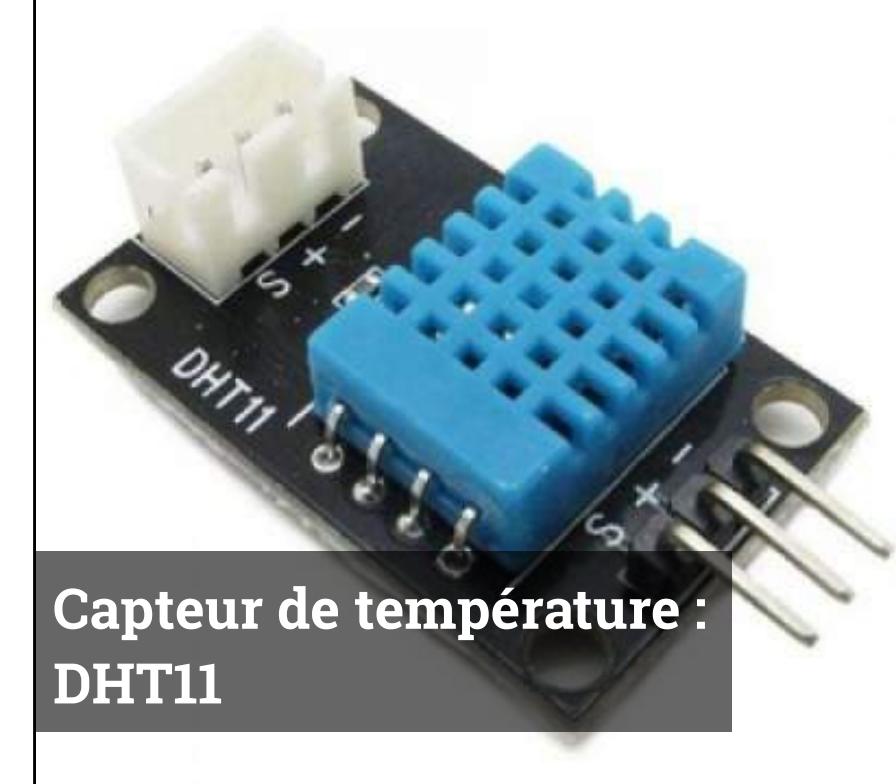


Les 5 dessins possibles sur la matrice de LEDs RGB

### > Les Ports GPIO / Capteurs

Pour interagir avec des composants électroniques le Raspberry Pi dispose de ports GPIO (General Purpose Input Output). Dans le projet, nous avons utilisé la librairie WiringPi, qui permet de piloter les GPIOs.

Nous avons interagis avec 4 différents composants. Le premier est le **DHT11**, il permet de récupérer la température ainsi que l'humidité. Le second est le **BH1750**, il s'agit d'un capteur de luminosité. Il y a ensuite une matrice de LED pilotée par un **MAX7219**, ce qui permet d'interagir visuellement avec l'utilisateur de voicezy. En effet, elle permet d'indiquer si la reconnaissance vocale est en cours et si la commande a été reconnu ou non. Finalement, nous interagissons avec le **buzzer** pour lui faire jouer la musique de Mario.



### > L'Utilisation de Firebase



**Firebase** à été choisi pour remplacer l'API Twitter, car en effet, nous n'avons jamais réussi à obtenir un accès à cette API malgré plusieurs demandes.

Firebase est un ensemble de services d'hébergement pour n'importe quel type d'application. Il propose d'héberger en NoSQL et en temps réel des bases de données, du contenu, de l'authentification sociale, et des notifications, ou encore des services, tel que par exemple un serveur de communication temps réel.

Nous utilisons ici qu'une partie du service qui est la **"Realtime Database"**. N'ayant pas trouvé de librairie en C pour ce service, nous avons utilisé la librairie python Pyrebase qui permet de connecter notre Raspberry Pi à notre base de données temps réel et de publier les informations que l'on souhaite. La "Realtime Database" de Firebase fonctionne en JSON. La base de données est comme un énorme JSON auquel on peut venir ajouter des données. Vous devez certainement vous dire que cela ne doit pas être très

performant et ... nous allons pas vous contredire, mais dans une application comme la nôtre. Cette base de données est parfaite : il nous suffit de

"publier" une donnée **JSON** depuis le Raspberry pour que la donnée soit accessible depuis n'importe quel appareil.

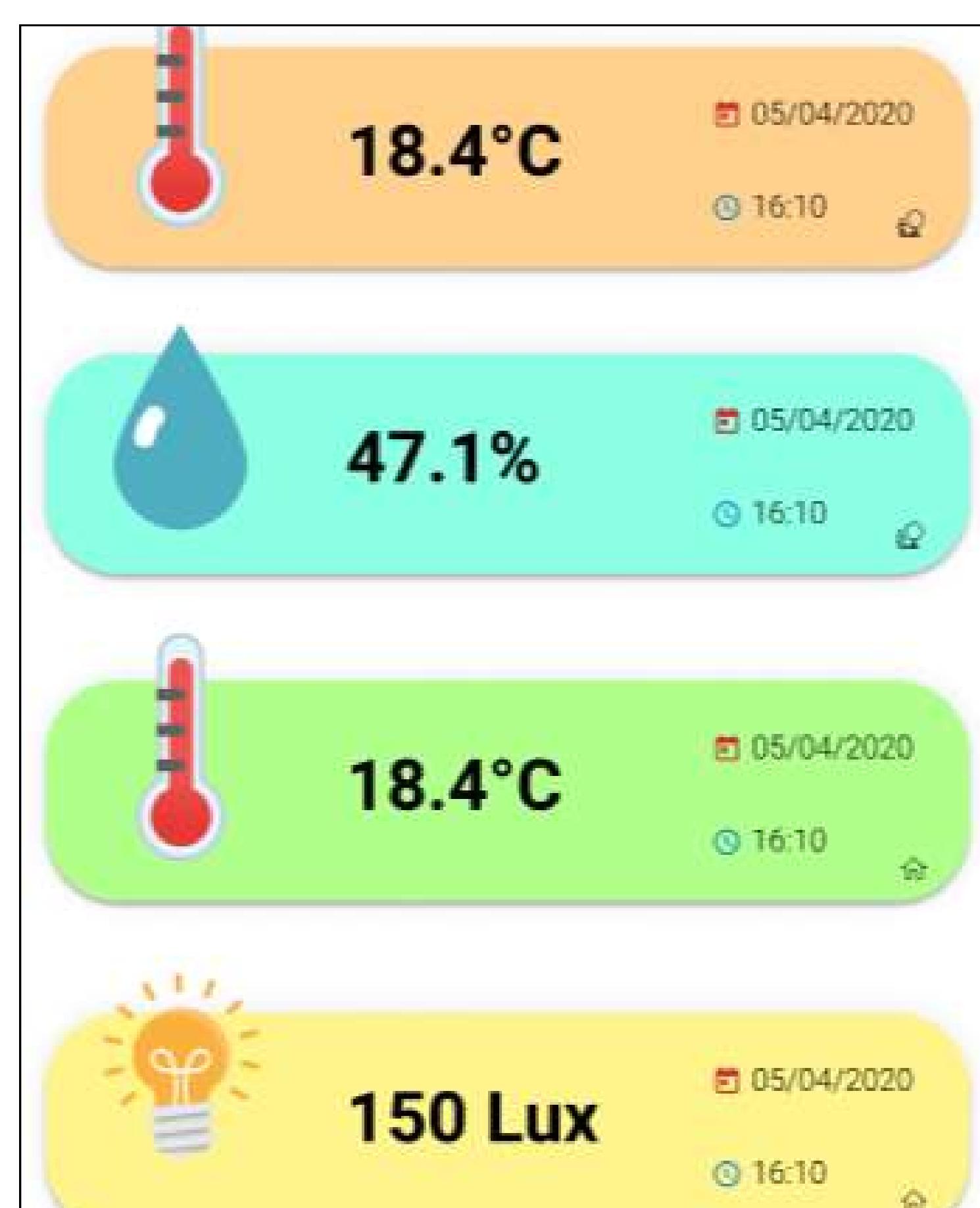
### > L'application en Flutter



Une fois l'implémentation de la base de données effectuée, il nous a fallu choisir un support, une interface qui puissent nous afficher les données que le Raspberry Pi envoie. Nous avons choisi d'utiliser **Flutter**, le fameux framework de Google. Il permet de concevoir des applications multi-plateformes pour Android et iOS.

Ce framework est le plus récent de tous. De ce fait, les ingénieurs ont pu observer les points forts et les faiblesses de chaque outil existant pour n'en extraire que la quintessence. C'est cela qui fait de Flutter un langage très puissant.

Nous n'allons pas rentrer dans les détails de la conception de jour. cette application, car ce n'est pas le sujet. .



Voici comment sont affichés les informations envoyées par le Raspberry

## LISEZ-NOUS PARTOUT

**ÉCONOMISEZ 25 %** avec un abonnement à Newsstand

**APPRENEZ À PROGRAMMER AVEC SCRATCH** avec notre nouveau livre numérique « ESSENTIALS » SEULEMENT 2,99 € 3,36 €

**GRATUITS : TÉLÉCHARGEZ LES 30 NUMÉROS ORIGINAUX**

Disponible maintenant sur Smartphones et sur tablettes

Available on the App Store    GET IT ON Google Play

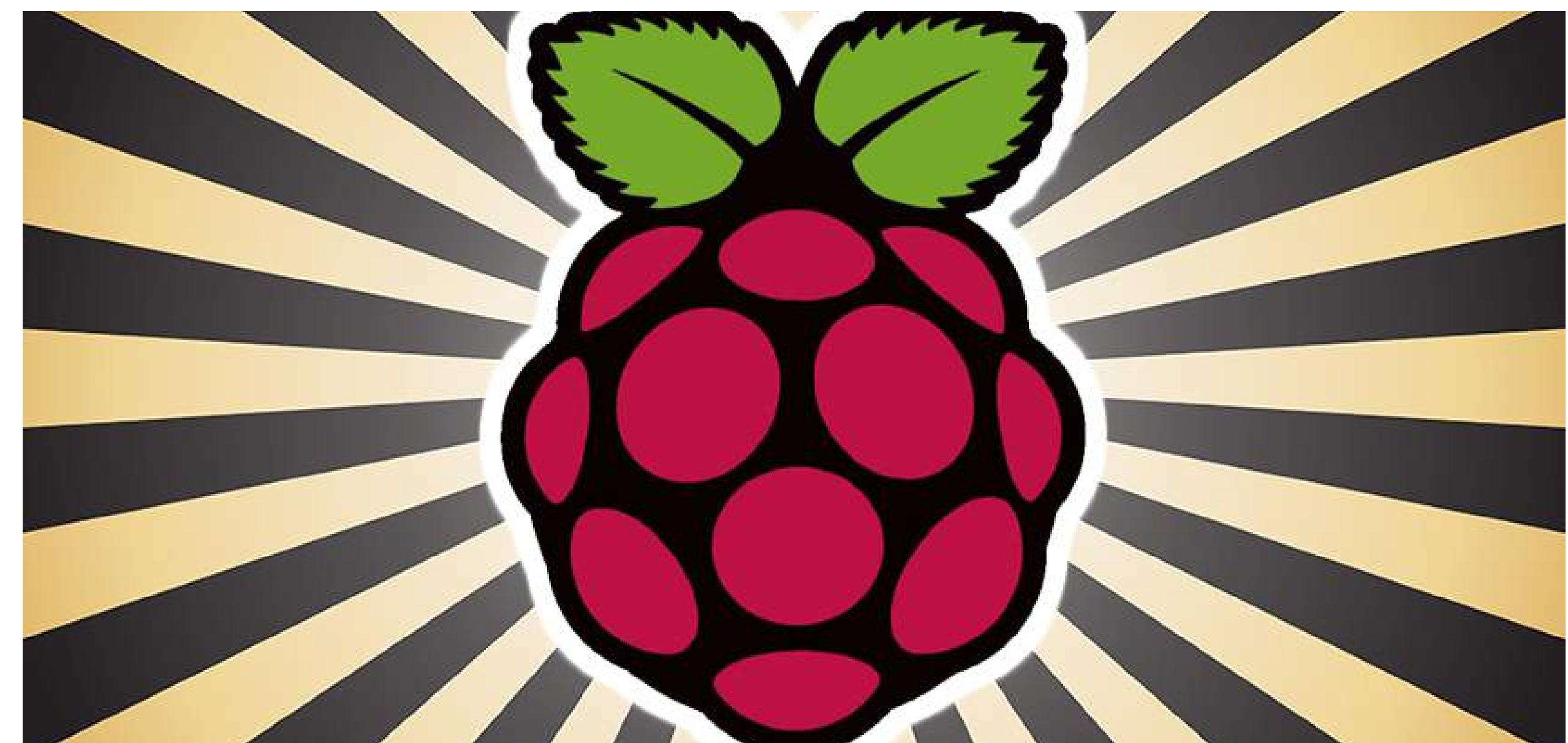
**Abonnez-vous**

**3,36 € ou 30,33 €**

abonnement reconduit    abonnement d'un an

**Téléchargez-le aujourd'hui, gratuitement !**

- Obtenez gratuitement les 30 numéros originaux
- Téléchargement instantané chaque mois
- Performance de rendu rapide
- Liens actifs et interactivité



**MagPi Offer:** LE MAGAZINE MAGPI PENDANT 1 AN + UN RPi 4 (2GB) OFFERT !

POUR 54,95 €

PIDAY20FB

offre réservée aux nouveaux membres, valable jusqu'au 16 mars 2020