PROGRAMMEZ!

Le magazine des développeurs

07/08/09 2020 N°1 Spécial

SPÉCIAL ÉTÉ 2020

/ **Créer des jeux 2D**
avec GDevelop

/ **Raspberry PI**
Créer un GPS
Comment utiliser Lora ?
Hacker les Sonoff

/ **Makers**
Les modules M5

/ **Hack**
Je hacke une carte bleue

/ **Lapin**
La nouvelle vie
du Nabaztag

/ **Matériel**
Diskio Pi :
une tablette française

© François Tonic

En partenariat avec
Framboise 314

M 01642 - 1H - F: 6,50 € - RD

Le seul magazine écrit par et pour les développeurs

Printed in EU - Imprimé en UE - BELGIQUE 7 € - Canada 9,80 $ CAN - SUISSE 13,10 FS - DOM Surf 7,50 € - TOM 1020 XPF - MAROC 55 DH

EDITO

Spécial été 2020

Après les mois difficiles que nous avons vécus, il était temps que l'été arrive… Avant de penser à la rentrée...

Et nous vous proposons notre premier trimestriel. Comme nous vous l'avons précisé dans le 241, Programmez! fait sa révolution : 6 numéros standards et 4 numéros spéciaux. Ces numéros seront thématiques.

Ce numéro thématique spécial été reprend la philosophie que nous appliquons depuis longtemps au magazine : profitons de l'été pour nous amuser un peu ! Nous allons parler matériel, maker, DIY, hacking, IoT, jeux et algorithme !

La Raspberry Pi sera une des vedettes de ce numéro. Framboise 314 vous a réservé 3 excellents articles : Lora, Sonoff et GPS ! Les modules M5 n'auront plus de secrets non plus. Une bonne excuse pour découvrir cette curieuse plateforme.

Avec GDevelop, vous pourrez développer, en quelques semaines, vos propres jeux 2D. Nous parlerons aussi de la PybStick série Programmez! Philippe nous montre comment utiliser un TFT 2,4'' tout en repoussant les limites de l'animation sur cette petite carte, qui possède une puissance insoupçonnée.

Nous parlerons du petit lapin Nabaztag et de la 3e génération grâce à la nouvelle carte, la TagTagTag et à une pile logicielle open source. Le lapin reprend vie et, cette fois-ci, il est réellement autonome.

Nous aurons aussi du hacking de carte bleue et les techniques possibles. Attention : dossier à manipuler avec prudence.

Bon été et rendez-vous le 4 septembre pour le numéro 242 !

François Tonic - ftonic@programmez.com

SOMMAIRE

Agenda — 4
Brèves — 6

La renaissance du Nabaztag — 8

Créer son NAS — 13

DiskioPi : une tablette française — 18
Découvrez la PybStick — 21
Pybstick + TFT — 23

Développer des jeux avec GDevelop — 27

Je débute en algorithmique… — 36

Abonnez-vous ! — 42
Boutique Programmez! — 43

Raspberry Pi + Lora — 40
Raspberry Pi + Sonoff — 49
Raspberry Pi + GPS — 53

Des quilles connectées — 58
Les modules M5Stack — 65

IoT partie 2 : la pratique — 71
Hacker une carte bleue — 77
CommitStrip — 82

Prochain numéro

PROGRAMMEZ! 242
DISPONIBLE LE 4 SEPTEMBRE

François Mocq
Electronicien, radioamateur (F1GYT), Technicien de maintenance puis formateur d'adultes en maintenance informatique, réseaux et télécom. Créateur du blog www.framboise314.fr *et auteur des livres "Raspberry Pi 4" et "Scratch et Raspberry Pi" parus aux Editions ENI*

Passerelle et nœud LoRaWan

L'IoT utilise différents systèmes de communication radio. Parmi ceux-ci, LoRaWan est un protocole à très bas débit et faible consommation. La passerelle LoRa peut être utilisée en autonome (site industriel, milieu agricole, ville intelligente, campus…) mais elle peut aussi se connecter à Internet. Les données recueillies par les nœuds peuvent alors être partagées, mises à disposition en OpenData.

L'origine du projet

J'ai un atelier situé à plus d'un kilomètre et je souhaite avoir régulièrement des informations sur la température, l'humidité et une éventuelle intrusion. Il serait bien également de pouvoir monter la température avant de m'y rendre quand les températures sont basses.

Le pont WiFi était une solution mais pour avoir quelque chose de fiable, le prix piquait un peu car il fallait partir sur une solution ayant une portée de 3 Km. L'autre solution, c'était de mettre en place une passerelle LoRa. Les anglais de PiSupply [1] proposent tout un ensemble de matériels adaptés au Raspberry Pi mais aussi à Arduino ou micro:bit. J'ai arrêté mon choix sur une gateway [2] (passerelle) pour le Raspberry Pi et une carte node [3] (nœud) pour le Raspberry Pi Zero.

Pour pouvoir consulter et piloter à distance, j'ai opté pour The Things Network [4] (TTN) qui est un réseau ouvert permettant de construire un réseau pour l'IoT à moindre coût. Ce réseau collaboratif regroupe des milliers de passerelles réparties sur le monde et utilisées par des millions de personnes. Le chiffrement de bout en bout permet de faire transiter les données par ces passerelles en sécurité.

Le matériel utilisé

Pour la passerelle

- Carte passerelle LoRa Gateway HAT pour le Raspberry Pi (170€)
- Raspberry Pi 4 – 2Go RAM (39€)
- Carte SD 16Go class 10 (10€)
- Alimentation 5v/3A (8€)
- Boîtier (6€)

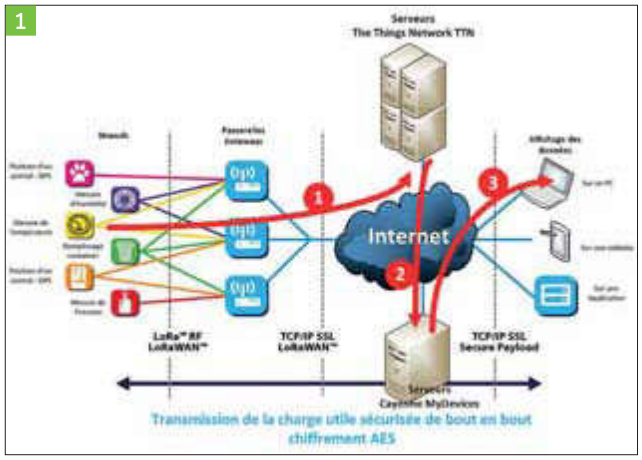
Pour le noeud

- Carte LoRa Node pHAT pour le Raspberry Pi (40€)
- Rapsberry Pi Zero WH (15€)
- Carte SD 16Go class 10 (10€)
- Alimentation 5v/3A (8€)

Principe de la réalisation

L'image 1 représente le synoptique de cette réalisation. A gauche le **Nœud LoRa** envoie les valeurs mesurées par radio vers la passerelle. Il n'y a qu'un nœud dans ce projet, mais plusieurs nœuds pourront envoyer tout un tas de données. La passerelle est chargée de recevoir les infos et de les relayer vers le serveur **The Things Network** (TTN). Ici il n'y aura qu'une passerelle. Lorsque plusieurs passerelles sont accessibles et qu'un nœud envoie les infos vers

1



plusieurs passerelles, le serveur se charge de supprimer les doublons, et ne garde que la trame provenant de la gateway qui reçoit le mieux le signal du nœud. Ce sera aussi sur cette passerelle qu'un éventuel envoi de données vers le nœud sera effectué. Dans mon cas j'ai un seul nœud et une seule passerelle, ce qui simplifie le schéma.

Une fois que les données sont arrivées sur le serveur TTN elles sont stockées pendant un temps limité. Il faut donc les renvoyer vers une base de données (BDD) pour les enregistrer de façon pérenne. Vous pouvez utiliser votre propre base de données. J'ai choisi la solution toute faite de MyDevices Cayenne dans sa version gratuite. Il suffit de créer un compte pour bénéficier des services de base de la plateforme. On pourra créer sur MyDevices un tableau de bord (DashBoard) qui sera accessible via internet sur tout périphérique (PC, tablette, smartphone…).

LoRa et LoRaWan

LoRa est l'acronyme de Long Range (Grande portée). C'est une technologie radio (sans fil) dans laquelle un émetteur à très faible consommation envoie de petits paquets de données à faible vitesse (0.3kb/s à 5,5kb/s) à un récepteur situé à grande distance.

LoRaWAN est un protocole de télécommunication permettant la communication à bas débit, par radio. Il est utilisé pour des objets à faible consommation électrique communiquant selon la technologie LoRa et connectés à l'Internet via des passerelles, participant ainsi à l'Internet des objets (IoT). (source Wikipedia)

L'image 1 montre l'architecture d'un réseau LoRaWAN. A gauche les "nœuds" collectent les données. Ils sont capables de les envoyer en consommant une énergie minime, grâce à une transmission à

bas débit. La portée est très variable mais on atteint couramment plusieurs kilomètres. La carte LoRa Node pHAT de Pi Supply est annoncée pour 15Km environ. Dans de (très) bonnes conditions, un signal LoRa a été reçu à 766Km, l'émetteur avait une puissance de 25mW. C'est actuellement le record du monde [13].

Les Gateways sont chargées de collecter les données des nœuds et de les transférer à un serveur qui pourra les traiter, les redistribuer… Il existe des solutions propriétaires, pour ma part j'ai choisi d'utiliser The Things Network (TTN) un réseau international gratuit, open source et communautaire. Les serveurs de TTN vous permettent de recevoir dans vos applications les données de vos appareils, même à des milliers de kilomètres de chez vous !

Actuellement (janvier 2020) le réseau se compose de plus de 40000 contributeurs regroupés en plus de 400 communautés dans 90 pays ayant déployé plus de 4000 passerelles. Il est possible pour les particuliers, universités, entreprises, ou encore les communes de contribuer au déploiement ou d'utiliser gratuitement The Things Network.

En Europe, la bande réservée à LoRaWan est la bande 868MHz qui est une bande ISM *(industriel, scientifique, et médical)* utilisable sans autorisation préalable des autorités. La puissance d'émission et le taux d'utilisation temporel sont limités par la réglementation. Pour les autres régions du monde, référez-vous à la réglementation locale.

La modulation LoRa utilise un étalement de spectre appelé **Chirp Spread Spectrum** pour coder l'information. SigFox utilise l'Ultra Narrow Band. Un chirp ou gazouillis est un signal sinusoïdal dont la fréquence varie au cours du temps. Initialement destinée à des applications militaires, cette modulation est également utilisée par les radars et les sonars. Elle est robuste contre différents types de perturbations.

L'image 2 donne une indication de la portée du signal en fonction du facteur d'étalement (SF). La portée réelle dépend des conditions locales (immeubles, montagne, vallée…). Si l'étalement augmente, la portée augmente, mais le débit se réduit et la durée de transmission augmente, réduisant l'autonomie du dispositif. Il faudra adapter ces facteurs ainsi que la puissance pour obtenir la portée souhaitée.

On confond souvent LoRa et LoRaWAN, ce sont deux couches différentes du système :

LoRa correspond à la couche physique (radio et modulation) ;

LoRaWAN gère les couches MAC (Media Access Control) et Application.

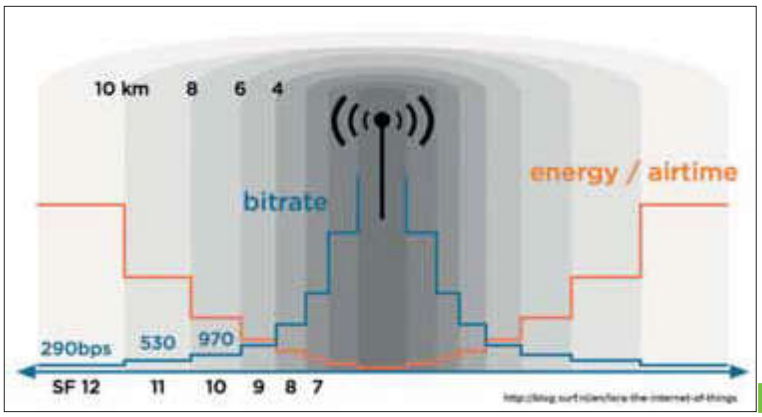
The Things Network

The Things Network (TTN) est un réseau LoRaWAN communautaire et open source conçu pour l'Internet des Objets (IoT en anglais). Il assure un chiffrement robuste de bout en bout. Il faudra créer un compte avant de commencer à utiliser le réseau. TTN envoie un mail de confirmation pour valider l'inscription.

Inscription

Avant de démarrer la passerelle, il faut créer le compte TTN pour récupérer la clé de la passerelle. Sur la page d'accueil cliquez sur **Sign In** et créez un compte. 3

Cliquez sur l'icône **Gateways** pour déclarer votre passerelle. Dans



2

3

la fenêtre qui s'ouvre, vous allez remplir les champs suivants :

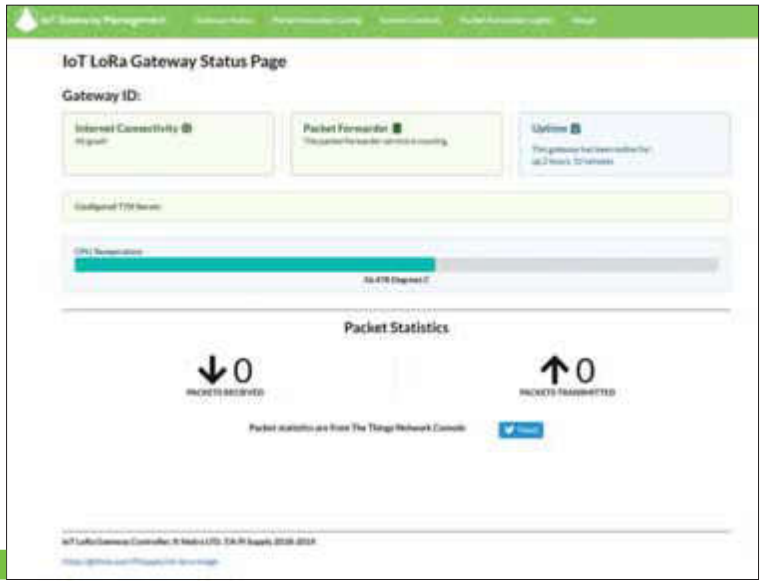
- **Gateway ID/ID de la passerelle** : il doit s'agir d'un identifiant unique, qui peut être n'importe quoi, mais qui doit être unique.
- **Description** : texte libre, c'est pour que vous ayez une description de la passerelle.
- **Frequency plan/Plan de fréquences** : il s'agit de la fréquence de la passerelle qui doit correspondre à la région. Comme nous sommes en Europe, j'ai choisi le plan **Europe 868Mhz**.
- **Router** : c'est le serveur sur lequel la passerelle va se connecter. Il devrait être le plus proche de votre lieu de résidence. Comme nous sommes en Europe, j'ai sélectionné **ttn-router-eu**, mais vous trouverez peut-être un routeur plus proche de vous.
- **Location/Emplacement** : cette indication n'est pas obligatoire mais indiquer l'emplacement de la passerelle peut être utile à l'avenir pour la cartographie de la couverture du signal LoRa. Cliquez à l'emplacement de la passerelle, sur la carte.
- **Antenna placement/Placement de l'antenne** : indiquer si la passerelle et l'antenne sont à l'intérieur ou à l'extérieur. Pour moi elle est actuellement à l'intérieur, ce qui réduit la zone de couverture.

Vous pourrez revenir sur tous ces paramètres, excepté l'ID de la passerelle qui ne peut être modifié. Une fois ces informations validées, une fenêtre **Gateway Overview** vous donne un résumé des paramètres. La clé **Gateway Key** est visible en cliquant sur l'œil à gauche de la zone. Vous pourrez la copier en cliquant sur l'icône de la feuille à droite de la zone. Cela vous servira un peu plus tard car il faudra coller cette clé lors de la configuration de la passerelle sur le Raspberry Pi.

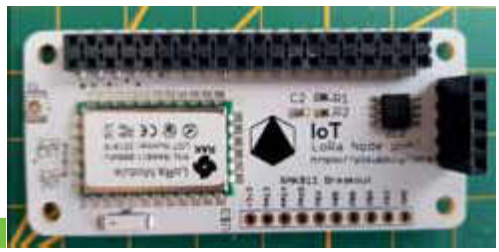
Il reste à modifier quelques paramètres. Cliquez sur l'onglet "**Settings**" et vous devriez pouvoir voir les paramètres que nous avons configurés, puis sur l'onglet "**Informations**" à gauche.

4

5



6



7

Indiquez la marque "**passerelle multicanal Raspberry Pi**" et pour le modèle, sélectionnez "**Raspberry Pi avec Pi Supply Gateway HAT**". Ne connaissant pas le type exact de l'antenne je n'ai rien indiqué. Pour terminer cliquez sur **Update Gateway** pour enregistrer toutes les informations relatives à la passerelle que vous venez de créer. Il est maintenant possible de rattacher la passerelle Raspberry Pi à ce compte.

Passerelle LoRaWAN

La passerelle IoT LoRa Gateway HAT [5] pour Raspberry Pi (868MHz/915MHz) est proposée par PiSupply (4). Elle utilise le module concentrateur de passerelle LoRa RAKWireless **RAK833** mPCIe qui est basé sur la technologie LoRa Semtech SX1301 et reçoit les 8 canaux simultanément. Elle est entièrement compatible LoRaWAN. La passerelle est fournie sous forme de carte HAT. Un radiateur non représenté ici doit obligatoirement être collé sur le RAK833 pour assurer son refroidissement. Il faudra le mettre en place AVANT de connecter la carte (5).

Le module RAK833 comporte une prise miniature u.FL sur laquelle l'antenne fournie sera connectée.

PiSupply a facilité la mise en œuvre de cette passerelle en mettant à disposition une distribution toute prête, basée sur Raspbian Lite. Elle est compatible avec toutes les versions de Raspberry Pi, y compris le Raspberry Pi 4. Elle est disponible en téléchargement [6] et représente environ 500Mo. Après avoir récupéré la distribution, transférez-la sur une carte micro SD avec Etcher si vous êtes sous Windows ou `dd` sous Linux. La Fondation Raspberry Pi a également développé une solution disponible sur Windows, Mac et Ubuntu : Raspberry Pi Imager.

Connectez un câble Ethernet au Raspberry Pi et mettez l'ensemble sous tension. Le nom de la passerelle est ***iotloragateway***, retrouvez son adresse sur votre box ou utilisez **http://iotloragateway.local**. Ouvrez un navigateur web sur cette adresse pour ouvrir la page web de configuration de la passerelle (6), cliquez sur **Packet Forwarder Config** dans la barre supérieure.

Saisissez les identifiants par défaut (*pi*, *raspberry*). Renseignez votre adresse électronique, puis les champs correspondant au compte que vous avez créé dans The Things Network : identifiant et clé de la passerelle (**Gateway Key**) fournie précédemment par TTN. Cliquez ensuite sur **Update Configuration**.

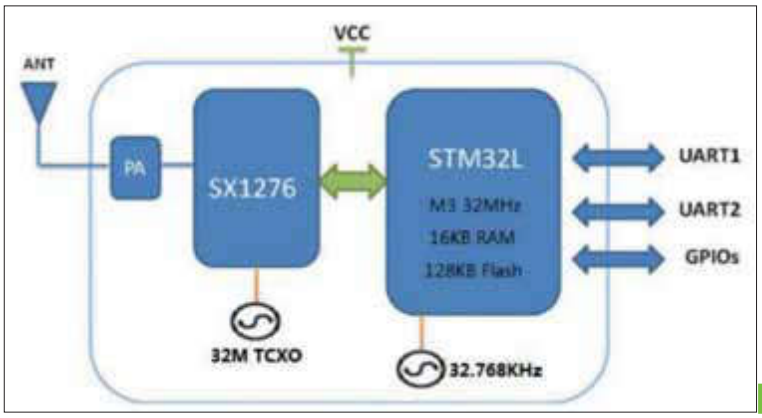
Pour démarrer la passerelle, nous devons maintenant aller dans l'onglet "**System Controls**" dans le menu du haut puis cliquer sur le bouton **Restart** pour prendre en compte les modifications et redémarrer la passerelle. Une barre de progression indique l'évolution de l'opération et à la fin, la lumière bleue du module LoRa devrait s'allumer et vous êtes redirigé(e) vers la page d'accueil. Cette page indique l'ID de votre passerelle, le nombre de paquets qu'elle a reçus et d'autres informations pratiques.

La passerelle est prête, elle peut recevoir et retransmettre les paquets envoyés par un nœud vers les serveurs TTN. Le nœud peut être celui que nous allons mettre en place maintenant, ou tout objet LoRaWAN qui sera reçu par la passerelle.

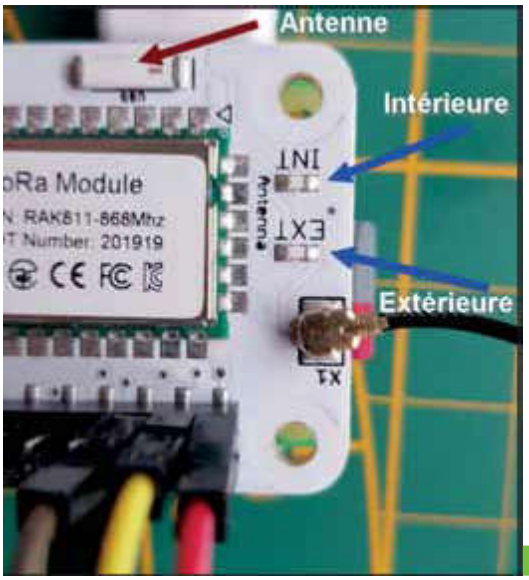
Le nœud LoRaWAN

La carte LoRa Node pHAT [3] utilisée provient également de PiSupply (7). Elle est au format du Raspberry Pi Zero. La carte

utilise le module **LoRa RAKWireless RAK811** qui est basé sur la technologie LoRa Semtech SX1276 (8). La carte peut être configurée pour différentes fréquences, pour l'Europe les fréquences vont de 863 à 870 MHz.
La carte communique avec le Raspberry Pi par l'UART en utilisant seulement 3 broches GPIO. Les autres restent libres pour connecter des capteurs. Elle prend en charge les connexions LoRaWAN et LoRaP2P. Elle dispose d'une antenne intégrée et d'un connecteur u.FL permettant de choisir son antenne et de faciliter l'intégration en boîtier. J'ai utilisé une queue de cochon u.FL – SMA et une antenne SMA 868MHz. La carte consomme moins de 50mA en émission.
La carte est fournie avec un rehausseur et deux jeux d'entretoises. Si vous utilisez la carte seule sur un Raspberry Pi, les petites entretoises feront l'affaire. Si vous souhaitez brancher autre chose pardessus la carte sur le GPIO, il faudra mettre en place le rehausseur de GPIO et les grandes entretoises. Les broches du rehausseur débordent alors sur le dessus du connecteur et permettent de connecter d'autres périphériques. J'utiliserai un BME680 pour relever les valeurs de **Température**, **Pression** et **Humidité** qui seront transmises via LoRa.
9 La carte Comporte deux straps (résistances de 0 ohm) qui relient l'antenne **INT**érieure **et** l'antenne **EXT**érieure à la sortie du circuit. L'antenne embarquée est le rectangle en haut de l'image. L'utilisation simultanée des deux antennes en parallèle perturbe l'adaptation d'impédance et le rayonnement. Pour mon utilisation (plus d'un Km de portée) j'ai utilisé l'antenne extérieure et supprimé le strap **INT** avec le fer à souder. Il faut brancher le connecteur u.FL sur la prise. ***Attention, le petit connecteur u.FL n'est pas conçu pour être branché/débranché de nombreuses fois***.



8



9

Installation du nœud LoRaWAN

Les instructions pour installer le nœud LoRaWAN sont fournies [en anglais] par Pi Supply sur leur site web [7]. Ce qui suit en est une adaptation.
Montez la carte Node pHAT sur le Raspberry Pi Zero.
Créez une carte micro SD avec Raspbian Lite (utilisez la dernière version lite - sans Desktop texte seulement). Mettez le système à jour. Éventuellement faites les mises à jour sur un Pi 3 ou un Pi 4… Ça ira plus vite. Pour utiliser le Raspberry Pi Zero il va falloir effectuer quelques réglages.
Lancez raspi-config :

```
sudo raspi-config
```

Configurez les locales pour franciser le système puis dans **2 Network Options > N2 WiFi** configurez les paramètres pour vous connecter à votre réseau WiFi (SSID et passphrase).
On va maintenant modifier la configuration pour utiliser l'UART matériel et désactiver la console.
Dans **5 Interfacing Options > P6 Serial** Répondez <**Non**> (voulez-vous qu'un login shell soit accessible via le port série) puis sélectionnez <**Oui**> pour activer le port série matériel.
Enfin on va intervertir les interfaces série pour que la meilleure interface (matérielle) attribuée par défaut au Bluetooth, soit attribuée à la carte LoRa. Dans un terminal saisissez :

```
sudo nano /boot/config.txt
```

Positionnez le curseur en bas du fichier et ajoutez :

```
dtoverlay=pi3-miniuart-bt
```

Sauvegardez et fermez le fichier avec "**CTRL+O**" et "**CTRL+X**".
Redémarrez le Raspberry Pi pour prendre les modifications en compte.
Nous allons maintenant installer la bibliothèque RAK811 :

```
sudo apt install python3-pip
sudo pip3 install rak811
```

A partir de là, Raspbian a les éléments pour fonctionner avec la carte LoRa.

Connexion au compte TTN

Il existe deux méthodes différentes pour rejoindre le réseau The Things Network à partir du nœud Raspberry Pi pHAT : OTAA et ABP.

Activation en direct (OTAA)

L'activation en direct (OTAA) est le moyen recommandé (et le plus sûr) pour se connecter au réseau The Things Network. Les appareils exécutent une procédure pour rejoindre le réseau, au cours de

laquelle une Adresse de module Dynamique (DevAddr) est attribuée et des clés de sécurité sont négociées avec le module.

Activation par personnalisation (ABP)

Dans certains cas, vous devrez peut-être adresser en dur le DevAddr ainsi que les clés de sécurité de l'appareil. Cela signifie qu'il faudra utiliser l'activation par personnalisation (ABP). Cette stratégie peut sembler plus simple, car vous évitez la procédure utilisée pour rejoindre le réseau, mais elle présente certains inconvénients liés à la sécurité. J'ai donc opté pour la Connexion OTAA
Pour connecter un nœud au réseau TTN, il faut d'abord déterminer l'EUI (Extended Unique Identifier = identificateur unique sur 64 bits) de notre appareil. Cette valeur unique pour chaque composant permet 2^{64} valeurs, soit plus de 1.8×1019 possibilités. (de 0 à 18 446 744 073 709 551 615). Même avec un grand nombre de dispositifs LoRa, on devrait être tranquille un moment.
Ouvrez un terminal et tapez les deux lignes suivantes pour obtenir l'EUI : (la première ligne resette le rak811, vous n'avez besoin de l'utiliser qu'une seule fois après avoir démarré le système).

```
rak811 hard-reset
rak811 get-config dev_eui
```

Ce qui donne :

```
pi@raspberrypi:~ $ rak811 get-config dev_eui
3038383659386A0C
```

Et vous récupérez l'identifiant unique de votre matériel.

Le programme Python

Le programme est chargé de relever les valeurs T, P, RH toutes les minutes et de les envoyer à la passerelle. Pour ajouter un capteur, vous devrez ajouter les bibliothèques correspondant au capteur que vous utilisez. Ici j'ai installé celle du BME680.

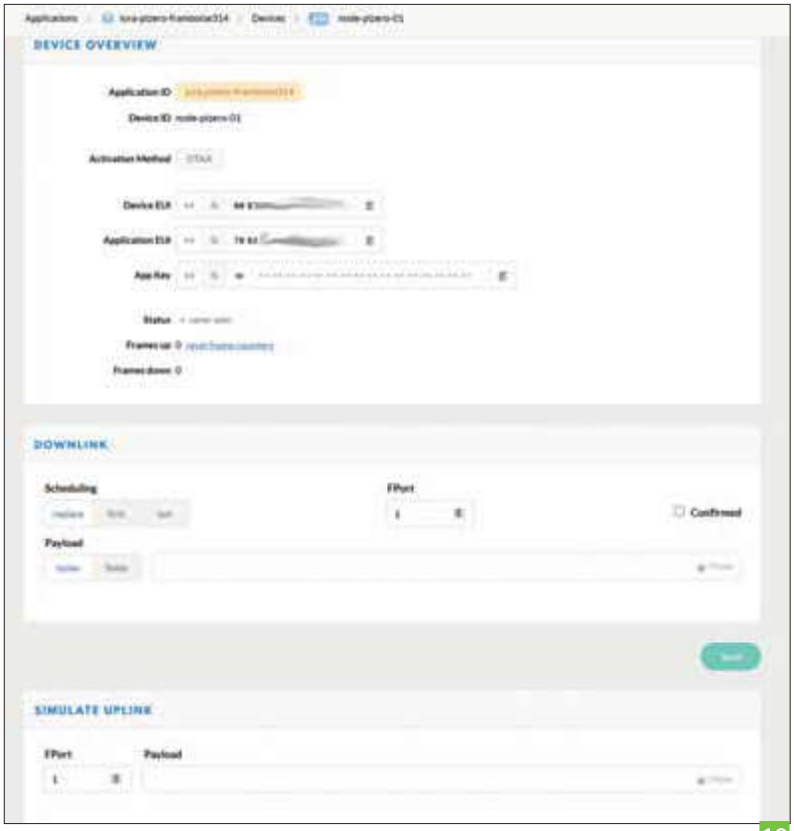
```
#!/usr/bin/env python3
"""RAK811 OTAA demo.

Ajout de la lecture du BME680
Ajout de pression, température, humidité
Minimalistic OTAA demo
Copyright 2019 Philippe Vanhaesendonck
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.
SPDX-License-Identifier: Apache-2.0
"""
from random import randint
from sys import exit
from time import sleep
import bme680
from rak811 import Mode, Rak811
# from ttn_secrets import APP_EUI, APP_KEY

lora = Rak811()

# Most of the setup should happen only once...
print('Setup')
# Crée le capteur en testant les deux adresses possibles 0x76 et 0x77
try:
  capteur = bme680.BME680(bme680.I2C_ADDR_PRIMARY)
except IOError:
  capteur = bme680.BME680(bme680.I2C_ADDR_SECONDARY)

# Initialisations LoRa
lora.hard_reset()
lora.mode = Mode.LoRaWan
lora.band = 'EU868'
lora.set_config(app_eui='70xxxxxxxxxxxxF',
          app_key='05C0xxxxxxxxxxxxxxxxxxxxxxxxx47F')

print('Joining')
lora.join_otaa()
# Note that DR is different from SF and depends on the region
# See: https://docs.exploratory.engineering/lora/dr_sf/
# Set Data Rate to 5 which is SF7/125kHz for EU868
lora.dr = 5

print('Sending packets every minute - Interrupt to cancel loop')
print('You can send downlinks from the TTN console')
try:
  while True:
    # Si les données du capteur sont lues
    if capteur.get_sensor_data():
      # Afficher la température
      Temp = int(10*round(capteur.data.temperature, 1))
#       print(f'Température : {capteur.data.temperature:.1f} °C')
      print('Température : ', Temp/10)
      # Afficher la pression
      Pression = int(10*round(capteur.data.pressure, 1))
      print('Pression : ', Pression/10)
#       print(f'Pression : {capteur.data.pressure:.1f} hPa')
      # Afficher l'humidité relative
      HR = int(2*capteur.data.humidity)
#       print(f'Humidité : {capteur.data.humidity:.1f} %HR')
      print('Humidité : ', HR/2)
      print("==============")

    print('Envoi du paquet de données')
    # Cayenne lpp random value as analog
#     lora.send(bytes.fromhex('0167{:04x}'.format(randint(0, 0x7F))))
    lora.send(bytes.fromhex('0167{:04x}0273{:04x}0368{:02x}'.format(Temp, Pression, HR)))

    while lora.nb_downlinks:
      print('Received', lora.get_downlink()['data'].hex())
```

```
    sleep(60)
except: # noqa: E722
  pass

print('Cleaning up')
lora.close()
exit(0)
```

Vous pouvez également récupérer le programme sous forme d'archive .zip en [8]. Ce programme est disponible sur le Github d'Amédée Bulle [9]. Le programme ci-dessus est un test, il comporte encore des éléments inutiles, reliquats du programme original. On y trouve également la réception d'un downlink (message descendant vers le nœud) qui pourrait être utilisé pour déclencher une action (allumer la chaudière, déclencher une alarme, allumer une lampe…).
En ligne 39 on resette le circuit LoRa (à faire au moins une fois après la connexion de la carte à l'alimentation).
En ligne 42 on entre les identifiants de l'appli et la clé correspondante. (Il faudra rentrer vos propres identifiants).
En ligne 46 on se connecte à la passerelle en OTAA. Over The Air : les clefs de chiffrement sont obtenues par un échange avec le réseau.
Des lignes 56 à 70 on récupère les données climatiques et on les met en forme pour les inclure dans la payload.
En ligne 75 on crée la trame au format CayenneLPP et on l'envoie. Le programme transmet la payload (voir ci-dessous) actualisée toutes les minutes pour les tests (à modifier selon vos besoins et en fonction de la réglementation).
Lorsque vous lancez le programme en lignes de commandes puis que vous fermez la fenêtre, il sera arrêté si vous étiez en SSH (puTTY). Pour continuer l'exécution après fermeture de la fenêtre ajoutez un **&** à la fin de la ligne.

Payload Cayenne LPP

Pour diminuer le nombre d'octets transmis, il est intéressant d'utiliser Cayenne LPP. C'est un codage open source créé par Cayenne disponible sous forme de bibliothèque. Une payload CayenneLPP contient des métadonnées qui permettent de savoir quelles données sont transmises. Pour chaque valeur le premier octet envoyé est l'identificateur, l'identifiant du canal. Le deuxième octet est le type de données, le Data Type. On trouve ensuite la valeur mesurée. Vous trouverez plus d'informations sur Cayenne LPP en [10].
Décortiquons cette charge utile de 11 octets :
01 67 00 D7 02 73 25 C9 03 68 51
01 67 00 D7 :
• **01** = 1d => canal 1
• **67** = 103d => Capteur de température valeur sur 2 octets
• **00 D7** = 215d => **21.5°C** (valeur/10)
02 73 25 C9 :
• **02** = 2d => canal 2
• **73** = 115d => Baromètre valeur sur 2 octets
• **25 C9** = 9673d => **967.3 hPa** (valeur/10)
03 68 51 :
• **03** = 3 => canal 3
• **68** = 104d => Capteur d'humidité valeur sur 1 octet



10

• **51** = 81d => **40.5 %** (valeur/2 car bit de poids faible =0,5% HR)
En 11 octets, CayenneLPP a donc permis de passer les valeurs de 3 capteurs et le type des informations transmises. Ce format est très utilisé dans le monde de l'IoT car il facilite la transmission des données, tout en conservant une occupation réduite de la bande passante.

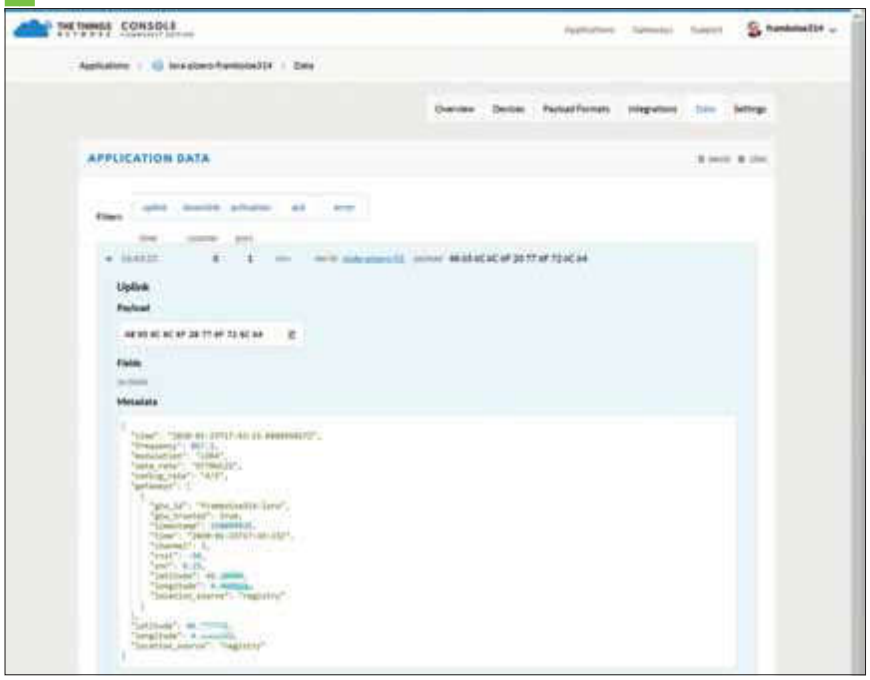
Inscription du nœud sur The Things Network

Nous avions précédemment configuré la Gateway sur TTN, maintenant on passe à la configuration du nœud. Connectez-vous à votre compte The Thing Networks. Sur l'écran d'accueil de TTN cliquez sur l'icône **APPLICATIONS** (image 6) puis sur **Add Application**.
Donnez un nom à votre application (attention tous les caractères ne sont pas autorisés) et ajoutez une description. Sélectionnez le serveur TTN qui sera sollicité, ici le serveur européen **ttn-handler-eu**. Validez vos choix avec le bouton **Add Application**. Vous récupérez un numéro d'application **EUI** unique, qu'on va associer avec votre carte LoRa.
Dans l'onglet **Devices**, cliquez sur **Register Device**. Donnez un nom à votre matériel, ce nom ne pourra pas être modifié. Collez le **Device EUI** récupéré plus haut en ligne de commande sur le Raspberry Pi Zero. **App Key** sera générée par la suite. Cliquez sur **Register** pour enregistrer vos modifications.
Après un moment vous avez accès à la fenêtre **Device Overview** (10). L'**App Key** existe et vous pouvez la voir (clic sur l'icône œil) et la copier (clic sur l'icône feuille à droite du champ). En haut de la fenêtre (dans le fil d'Ariane) cliquez sur le nom de votre applica-

tion puis dans le menu sur **Payload Format**. Dans la rubrique **Payload Format**, choisissez CayenneLPP qui est le format utilisé dans ce tutoriel. L'utilisation de Cayenne LPP n'est bien entendu pas obligatoire, et vous pourrez créer votre propre format par la suite. Validez avec **Save** en bas de la fenêtre.

A partir de là 11, vous devriez voir apparaître des données dans la rubrique **Data**. La payload brute est affichée. En dépliant les données vous aurez une vue sur l'ensemble des informations reçues par The Things Network.

11



Conclusion

Avec ce kit, PiSupply propose un ensemble qu'on peut rapidement mettre en œuvre pour découvrir le monde LoRaWAN. La disponibilité d'une distribution « clé en main » pour la passerelle est un plus incontestable. Il suffira ensuite de quelques lignes de Python pour développer un nœud opérationnel rapidement. Les transmissions proposées ici sont des tests. En exploitation il faudra respecter un certain nombre de contraintes [11] : 30 secondes d'émission par dispositif et par jour. Cela correspond pour une payload de 10 octets à 2 messages par jour en SF12 ou 500 messages par jour en SF7

Il est également possible d'envoyer des données vers le nœud (downlink). Les conditions sont encore plus restrictives que pour la remontée d'informations.

Les données ne sont pas conservées indéfiniment sur The Things Network [11]. Elles sont supprimées au bout de 7 jours. Si vous avez besoin de les conserver vous pouvez les récupérer avec l'API ou utiliser le service Cayenne pour créer des tableaux de bord (dashboard) et afficher les données.

Sources

[1] https://uk.pi-supply.com/search?type=product&q=lora
[2] https://uk.pi-supply.com/products/iot-lora-gateway-hat-for-raspberry-pi?lang=fr
[3] https://uk.pi-supply.com/products/iot-lora-node-phat-for-raspberry-pi?lang=fr
[4] https://www.thethingsnetwork.org/
[5] https://uk.pi-supply.com/products/iot-lora-gateway-hat-for-raspberry-pi?lang=fr
[6] https://github.com/PiSupply/iot-lora-image/releases
[7] https://learn.pi-supply.com/make/getting-started-with-the-raspberry-pi-lora-node-phat/
[8] https://www.framboise314.fr/wp-content/uploads/2020/01/lora_node_progpy.zip
[9] https://github.com/AmedeeBulle/pyrak811/blob/master/examples/otaa.py
[10] https://community.mydevices.com/t/cayenne-lpp-2-0/7510
[11] https://www.thethingsnetwork.org/docs/applications/storage/
[12] https://developers.mydevices.com/cayenne/signin/
[13] https://www.thethingsnetwork.org/article/lorawan-distance-world-record
[14] https://www.raspberrypi.org/downloads/

1 an de Programmez!

ABONNEMENT PDF :
35 €

Abonnez-vous sur :
www.programmez.com

François Mocq
Electronicien, radioamateur (F1GYT), Technicien de maintenance puis formateur d'adultes en maintenance informatique, réseaux et télécom. Créateur du blog www.framboise314.fr *et auteur des livres "Raspberry Pi 4" et "Scratch et Raspberry Pi" parus aux Editions ENI*

Programmer un module Sonoff Basic avec un Raspberry Pi

La marque itead recouvre de nombreux produits de domotique. Parmi ceux-ci, la gamme Sonoff dans laquelle on trouve des prises connectées ou des commutateurs comme le Sonoff Basic. On trouve ce commutateur WiFi pour moins de 10 euros sur les sites en ligne. Pourquoi s'intéresser à ce commutateur me direz-vous ? Il est à base d'ESP8266 et il est tout à fait possible de le programmer avec un Raspberry Pi. Pas besoin de dégainer la grosse artillerie et d'allumer votre super PC, un Raspberry Pi suffira largement pour vous amuser avec ce module sur un coin d'établi !

Avertissement : ***Ce module est alimenté par la tension du secteur 240v. Le contact avec les fils soumis à cette tension est mortel ! Il vous appartient de prendre toutes les précautions pour éviter tout contact même accidentel avec le secteur. Les pistes de circuit imprimé du module transportent le 240v. Faites toujours comme si le module était connecté au secteur. On oublie vite que la prise est branchée ! Remettez systématiquement le module dans son boîtier avant de le connecter au secteur. L'auteur de l'article et Programmez! ne pourront être tenus pour responsable en cas d'accident ou de dommage.***

Le module Sonoff Basic

Le module Sonoff Basic WiFi Wireless est un **interrupteur piloté en WiFI**, protégé dans un boîtier **IP66** (protégé contre les poussières et les jets d'eau). Enfin ça, c'est la doc qui le dit. Pour les projections d'eau je serais plutôt réservé.

Caractéristiques

- Allumage et extinction à distance ;
- Application gratuite pour iOS et Android mobile eWeLink ;
- Contrôle sur réseau local – Allumez ou éteignez l'appareil même sans accès à Internet ;
- État du dispositif affiché en temps réel sur l'application ;
- Réglage de minuteries programmées/ compte à rebours/boucles pour activer/désactiver à une heure précise via l'application ;
- Intégration à la smarthome (maison intelligente) ;
- Allumage/Extinction d'un groupe d'appareils d'un seul coup ;
- Déclenchement possible par la température, le bruit … en fonction des capteurs installés ;
- Fonctionne avec Amazon Alexa, Google Assistant, IFTTT, Google Nest et de nombreux logiciels domotiques.

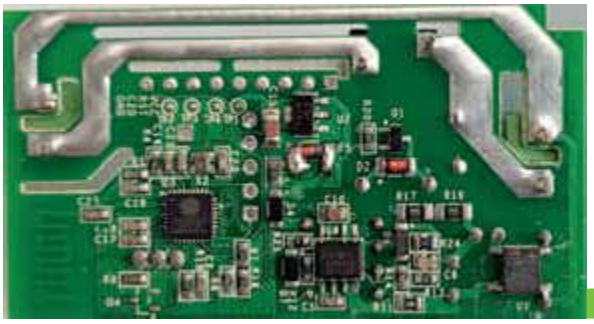
Sympa tout ça, non ? Moi ce qui me gêne un peu c'est de passer par l'appli eWelink pour allumer ma lampe de salon. Pas parce que ça me gêne qu'un serveur chinois enregistre cette action (quoique), mais parce qu'il semble inconcevable de mettre en œuvre toute cette technologie pour allumer… une lampe.

Le matériel

Le module reçu était présenté dans une boîte en carton fin (1). Les vis de fixation des plaques de blocage des fils ne sont pas montées, elles sont à part, dans un petit sachet en plastique. Ces deux plaques s'enlèvent donc très facilement. Le boîtier est juste clipsé à la base et s'ouvre très facilement avec l'ongle ou avec un tournevis fin.

1

2


3

Dans le boîtier du Sonoff Basic

A l'intérieur (2) on découvre le module monté sur son circuit imprimé. Le secteur 240v entre par la droite et ressort par la gauche après avoir transité via le relais. Un bouton poussoir accessible via l'extérieur du boîtier et une LED nous permettront d'interagir avec le module par la suite.

Sous la carte (3) on repère les pistes transportant la tension secteur (les grosses pistes étamées), séparées par des vides sans circuit pour éviter les amorçages, étincelles… Sur la gauche le circuit U3 qui est notre ESP8266. Le schéma de la carte est disponible en ligne [1].

Comment programmer ce module ?

Facile… En attaquant les entrées Tx et Rx de l'ESP8266. Sauf que…le fabricant a juste oublié de monter le connecteur pour y

accéder. On voit sur 4 les pastilles marquées 1, 2, 3, 4 et 5 qui n'attendent que l'arrivée d'un connecteur au pas de 2,54mm (1/10 pouce) pour servir. Sur votre module, il y a de fortes chances que les pastilles soient obstruées par de la soudure. Il faudra d'abord les en débarrasser avant de monter le connecteur et de le souder avec précaution.

Voilà ce que vous devriez obtenir. A présent notre ESP8266 est accessible via son port série (5) et nous allons pouvoir le programmer et recevoir les messages qu'il renvoie sur le port série. Vous avez accès à l'alimentation 3,3v et GND, au port série Rx et Tx, ainsi qu'à un GPIO, le N° 14. Notez bien que :

Le circuit doit être alimenté en 3,3v, **pas en 5v**

Il ne faut **pas connecter** l'alimentation 3,3v quand le module est relié au secteur. Sous une alimentation de 3,3v seule la LED s'allume, **le relais ne colle pas**

PROGRAMMER L'ESP8266 AVEC LE RASPBERRY PI

Si vous faites de l'Arduino, vous avez déjà tout ce qu'il faut pour programmer l'ESP8266, il suffit d'adapter l'IDE Arduino pour qu'il soit compatible avec les ESP. Si vous ne l'utilisez pas encore, voilà une occasion d'installer l'IDE sur votre Raspberry Pi.

L'IDE Arduino sur Raspbian

L'IDE Arduino s'installe aisément sur Raspbian. Naviguez sur la page arduino.cc et téléchargez depuis la page Download [2] la version arduino-1.8.12-linuxarm.tar.xz. (ou une version plus récente en fonction de la date à laquelle vous suivrez ce tutoriel). Installez l'IDE Arduino sur Raspbian. Dans un terminal tapez les lignes suivantes :

```
tar xvf arduino-1.8.12-linuxarm.tar.xz
cd arduino-1.8.12/
sudo ./install.sh
```

4

5

6

Vous trouverez une nouvelle icône dans le menu Programmation : **Arduino IDE**. Dans les types de cartes (Outils => Type de carte) l'ESP8266 ne figure pas, nous allons l'ajouter.

Nota : *Lors de l'installation j'ai eu un message d'erreur, je l'ai contourné en créant le fichier manquant manuellement.*

Le message :

```
pi@raspberrypi:~/Downloads/arduino-1.8.12 $ sudo ./install.sh
Adding desktop shortcut, menu item and file associations for Arduino IDE...
touch: impossible de faire un touch '/root/.config/mimeapps.list': Aucun fichier ou dossier de ce type
/usr/bin/xdg-mime: 848: /usr/bin/xdg-mime: cannot create /root/.config/mimeapps.list.new: Directory nonexistent
done!
```

Le contournement :

```
pi@raspberrypi:~/Downloads/arduino-1.8.12 $ sudo mkdir /root/.config
pi@raspberrypi:~/Downloads/arduino-1.8.12 $ sudo touch /root/.config/mimeapps.list
```

Puis relancez l'installation et il n'y a plus d'erreur.

Ajouter l'ESP8266 à l'IDE Arduino

Rendez-vous sur la page web [3] et récupérez le lien .json (https://arduino.esp8266.com/stable/package_esp8266com_index.json).

Dans l'IDE Arduino, ouvrez la fenêtre **Fichier => Préférences** et collez le lien json dans la zone de texte **URL de gestionnaire de cartes supplémentaires**.

Ouvrez **Outils => Type de Carte => Gestionnaire de carte**. Cherchez la bibliothèque **ESP8266 by ESP8266 Community** (6) et Cliquez sur le bouton **Installer**. Vous pouvez suivre l'installation sur une barre de défilement. Attendez la fin de l'installation, fermez la fenêtre et redémarrez l'IDE Arduino.

Vous avez maintenant accès aux ESP8266 dans la liste des cartes (**Outils => Type de carte**). Choisissez celui qui correspond à votre modèle. Ici ce sera **Generic ESP8266 Module**. Vérifiez que le port USB **/dev/ttyUSB0** est bien indiqué. Si vous ne voyez pas d'USB0, laissez l'ESP connecté au port USB et redémarrez le Raspberry Pi. Sélectionnez le port **/dev/ttyUSB0** pour communiquer avec l'ESP8266.

MODIFIER LE PROGRAMME DU MODULE SONOFF

Attention : pour cette partie, NE CONNECTEZ PAS LE MODULE AU SECTEUR.

Utilisez un adaptateur USB-Série (7) qui fonctionnera aussi bien sur un PC que sur un Raspberry Pi. Connecter les fils à l'adaptateur USB. On a besoin des fils **TxD** et **RxD** (émission et réception de données du port série) et de l'alimentation +3,3v et GND. Connectez ces fils sur les broches correspondantes (Image 05) du module Sonoff. Attention de ne pas faire d'inversion, surtout au niveau de l'alimentation, ce qui serait fatal au module. La pin du GPIO14, à droite, reste libre.

Connectez l'adaptateur USB-Série à un port USB du Raspberry Pi. Sur ce modèle d'adaptateur, une LED indique que l'adaptateur est sous tension. La LED du module Sonoff devrait se mettre à clignoter au rythme d'une fois par seconde. On peut maintenant passer le module en mode programmation : débranchez l'adaptateur USB du Raspberry Pi. Appuyez sur la tige du bouton poussoir. Maintenez le bouton appuyé et rebranchez l'adaptateur USB. **Maintenez encore**

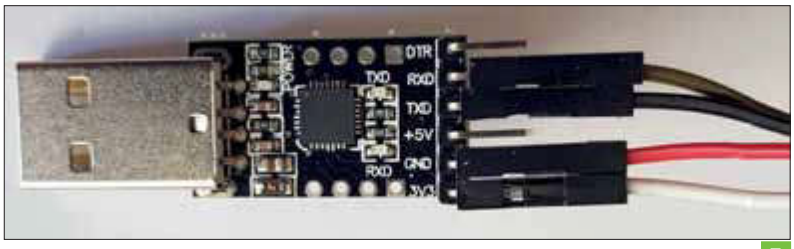
le bouton appuyé plus de 3 secondes. Vous pouvez relâcher le bouton. La LED ne doit pas clignoter. Si elle clignote c'est que vous n'avez pas appuyé assez longtemps ! Recommencez.
Votre module Sonoff Basic est maintenant passé en mode programmation, on peut charger un programme dans l'ESP8266 du module Sonoff Basic. Lancez l'IDE Arduino sur le Raspberry Pi (**Menu principal** > **Programmation** > **Arduino IDE**). Saisissez ce programme, disponible également en téléchargement [4].

```
/*
SONOFF Blink par Michel Deslierres
Modifié par François Mocq
Basé sur le programme Blink pour l'ESP8266 de Simon Peter
Fait clignoter la LED verte du Sonoff : deux flash rapides, un temps de pause
Ce code exemple est dans le domaine public
La LED verte du module Sonoff Basic est connectée au GPIO 13
*/
const int LED_PIN = 13;

void setup() {
pinMode(LED_PIN, OUTPUT); // Initialiser la pin LED_BUILTIN en sortie
}

// cette fonction loop tourne indéfiniment
void loop() {
digitalWrite(LED_PIN, LOW);  // Allume la LED (Noter que LOW est le niveau de la tension
                             // mais ici la LED est allumée car
                             // elle est active à l'état bas sur l'ESP-01)
delay(10);                   // Attendre 10ms
digitalWrite(LED_PIN, HIGH); // Eteindre la LED en mettant la sortie à 1 (3,3v)
delay (100);                 // Attendre 100ms
digitalWrite(LED_PIN, LOW);  // Allumer la LED
delay(10);                   // Attendre 10ms
digitalWrite(LED_PIN, HIGH); // Eteindre la LED
delay(500);                  // Attendre 500ms
}
```

Cliquez sur le bouton de téléversement de l'IDE. A la fin de la programmation, vous devriez avoir un message « **Hard resetting via RTS pin…** » dans le bas de la fenêtre de l'IDE Arduino.
Pour lancer le programme il faut redémarrer le module Sonoff. Débranchez et rebranchez l'adaptateur USB-Série sur la prise USB du Raspberry Pi. Après ce reset la LED doit clignoter en envoyant deux flashs brefs suivis d'une pause d'une demi seconde. J'ai modifié le programme blink dans ce sens car si on a un blink qui fait clignoter la LED exactement comme avant qu'on reprogramme le module… on se demande si ça a fonctionné ou pas.

Piloter le module Sonoff depuis une page web

On a fait la première partie du chemin en transférant avec succès un nouveau programme dans le module Sonoff. Au passage vous noterez qu'il ne pourra plus être utilisé comme le constructeur le prévoyait, c'est à vous de gérer tout ça maintenant. Dans l'IDE Arduino, saisissez le programme présenté en 8, que vous pouvez télécharger en ligne [5]. Modifiez les identifiants de connexion en fonction de vos identifiants. Déconnectez l'adaptateur USB pour mettre le



7

Sonoff hors tension. Appuyez sur le bouton du Sonoff, tenez appuyé. Rebranchez la prise USB. Attendez au moins 3 secondes que l'ESP8266 passe en mode programmation. Cliquez sur le bouton de téléversement de l'IDE Arduino. L'IDE compile votre programme (croquis) puis le transfère dans l'ESP8266.
Si tout se passe bien, à la fin du transfert vous obtenez un message «**Leaving**…Hard resetting via RTS pin… » Par contre le reset c'est vous qui allez le faire en débranchant et en rebranchant l'adaptateur USB du Raspberry Pi. L'ESP8266 redémarre et la LED doit clignoter plusieurs fois (ligne 52 à 64) en attendant que l'ESP8266 se voie attribuer une adresse IP par le DHCP de la box. Lorsque la LED s'arrête de clignoter (normalement ça clignote juste 3 ou 4 fois maxi) votre module Sonoff est prêt à recevoir vos ordres.

Utiliser le moniteur série

Si vous regardez le programme, il comporte de nombreuses lignes commençant par **Serial.println**. Ces lignes sont inutiles en utilisation normale du module Sonoff car il n'a pas d'écran. Par contre, pour la mise au point c'est un outil très intéressant. Les lignes Serial.println envoient les informations que vous décidez vers le port série. Il suffit d'ouvrir le terminal via le menu de l'IDE (**Outils** > **Moniteur Série**), de vérifier que la vitesse du moniteur (115200) correspond à celle que l'ESP8266 a reçue du programme (ligne 45) et vous verrez apparaître des indications sur ce qu'il se passe. Vous pouvez bien entendu ajouter d'autres infos comme l'état de la LED et du relais par exemple.

```
Connection au WiFi en cours...
.(IP unset)
6
.(IP unset)
6
.(IP unset)
6
.192.168.1.26
3
Connecté à Livebox-57A9
Accédez au serveur à l'adresse : 192.168.1.26
Serveur web prêt..
```

Avec ces informations et l'arrêt des clignotements de la LED vous savez que le module Sonoff est connecté à la box. Il n'y a plus qu'à commander le commutateur à distance.

Commande du module Sonoff Basic sur une page web

Ouvrez une page web sur l'adresse du module. C'est facile si vous avez ouvert le moniteur série, il affiche l'adresse IP du module

8

```
/*
  D'après Simple Wifi Switch de Jeffrey Kosbab
  https://gist.github.com/jeffreykrosbab/

  Adaptation Française : François Mocq
  www.framboise314.fr
  */

// Inclure les bibliothèques utilisées dans le script

// Gestion du WiFi pour l'ESP8266
#include <ESP8266WiFi.h>

// Client capable de se connecter à une adresse IP
#include <WiFiClient.h>

// Serveur web capable de gérer les requêtes HTTP POST et GET d'un client unique
#include <ESP8266WebServer.h>


// Identifiants de connexion à la box
const char* ssid = "Livebox-57A1";
const char* password = "2545[illegible]";


ESP8266WebServer server(80);

String web_on_html = "<h1>Etat actuel : ON</h1><p><a
href=\"on\"><button>ON</button></a> <a
href=\"off\"><button>OFF</button></a></p>";
String web_off_html = "<h1>Etat actuel : OFF</h1><p><a
href=\"on\"><button>ON</button></a> <a
href=\"off\"><button>OFF</button></a></p>";

int gpio_13_led = 13;
int gpio_12_relais = 12;

void setup(void){
  //  Initialisation des sorties LED et Relais
  // Eteindre la LED
  pinMode(gpio_13_led, OUTPUT);
  digitalWrite(gpio_13_led, HIGH);

  // Dévoiler le Relais
  pinMode(gpio_12_relais, OUTPUT);
  digitalWrite(gpio_12_relais, LOW);

  // Configurer la vitesse du port série
  Serial.begin(115200);
  delay(100);

  // Connecter l'ESP8266 au WiFi
  WiFi.begin(ssid, password);
  Serial.println("Connection au WiFi en cours...");

  // Attendre la connexion en faisant clignoter la LED
  while (WiFi.status() != WL_CONNECTED) {
    // Allumer la LED
    digitalWrite(gpio_13_led, LOW);
    delay(250);
    Serial.print(".");
    // Attente de l'adresse IP
    Serial.println(WiFi.localIP());
    Serial.println(WiFi.status());
    // Eteindre la LED
    digitalWrite(gpio_13_led, HIGH);
    delay(500);
  }

  Serial.println("");
  Serial.print("Connecté à ");
  Serial.println(ssid);
  Serial.print("Accédez au serveur à l'adresse : ");
  Serial.println(WiFi.localIP());

  // Réponse du serveur
  server.on("/", [](){
    if(digitalRead(gpio_12_relais)==HIGH) {
      server.send(200, "text/html", web_on_html);
    } else {
      server.send(200, "text/html", web_off_html);
    }
  });

  // Réponse lors de l'appui sur ON
  server.on("/on", [](){
    server.send(200, "text/html", web_on_html);
    // Allumer la LED, coller le Relais
    digitalWrite(gpio_13_led, LOW);
    digitalWrite(gpio_12_relais, HIGH);
    delay(1000);
  });

  // Réponse lors de l'appui sur OFF
  server.on("/off", [](){
    server.send(200, "text/html", web_off_html);
    // Eteindre la LED, décoller le Relais
    digitalWrite(gpio_13_led, HIGH);
    digitalWrite(gpio_12_relais, LOW);
    delay(1000);
  });

// Démarrage du serveur web
  server.begin();
  Serial.println("Serveur web prêt...");
}

// Boucle principale du programme
// Gestion du client web
void loop(void){
  server.handleClient();
}
```

Sonoff. Sinon allez fouiner dans le DHCP de votre box. La page web peut être ouverte sur le Raspberry Pi (on en profite puisqu'il est à côté), sur un PC, un MAC, un smartphone ou une tablette.

Normalement vous obtenez une page web comme celle qui est présentée sur 9. La LED est éteinte sur le module Sonoff. Cliquez sur le bouton **ON** : La LED s'allume. Attention, comme le module n'est pas encore relié au secteur, le relais ne colle pas et vous n'entendez rien.

La page web change et vous obtenez ce qui est présenté sur 10. La LED est allumée sur le module Sonoff. Cliquez sur le bouton **OFF** : La LED s'éteint et le relais ne fait toujours pas de bruit. Mais cette manipulation nous permet de vérifier que tout se passe bien et que le programme fonctionne. Vous pouvez voir cet essai en vidéo [6].

Utilisation en réel du module Sonoff Basic

NE CONNECTEZ RIEN AU SECTEUR POUR LE MOMENT

Débranchez tous les fils provenant de l'adaptateur USB qui sont connectés sur le module Sonoff. Branchez les fils secteur sur l'entrée du module. *Remontez le module dans son boîtier* **avant de le connecter au secteur**. Prenez toutes les précautions pour ne pas entrer en contact avec la tension 240 v.

Branchez la prise de courant. La LED va clignoter plusieurs fois en attendant l'adresse IP. Quand la LED ne clignote plus, allez sur la page web et actionnez les boutons. Cette fois la LED s'allume et s'éteint, mais vous entendez aussi le relais qui claque à chaque commande et si vous avez connecté une lampe 240v à la sortie du module, elle va s'allumer et s'éteindre en même temps que la LED.

Conclusion

Nous voici à la fin de ce tutoriel. Vous aurez vu comment modifier le programme d'un module Sonoff Basic peu cher pour l'adapter à vos besoins. Ce n'est qu'une mise en bouche, car avec ces infos, vous pouvez maintenant automatiser la commande. Il suffit de piloter depuis un programme en Python pour pouvoir gérer par programme l'allumage et l'extinction de ce qui est branché à la sortie du module. Vous avez sans doute remarqué que le GPIO14 qui était lui aussi accessible n'a pas été utilisé ici, et puis pourquoi ne pas récupérer les broches Tx et Rx comme E/S ? On pourrait y brancher d'autres choses : un autre relais… une sonde de température DS18B20 pour faire une espèce de thermostat… ? Mais ça, c'est une autre histoire.

Sources

[1] https://www.framboise314.fr/wp-content/uploads/2020/04/Sonoff-Schema.pdf
[2] https://www.arduino.cc/en/Main/Software
[3] https://github.com/esp8266/Arduino#installing-with-boards-manager
[4] https://www.framboise314.fr/wp-content/uploads/2020/04/blink_sonoff.zip
[5] https://www.framboise314.fr/wp-content/uploads/2020/04/test_web_FR.zip
[6] https://bit.ly/2RxECUe

9

10

François Mocq
Electronicien, radioamateur (F1GYT), Technicien de maintenance puis formateur d'adultes en maintenance informatique, réseaux et télécom. Créateur du blog www.framboise314.fr *et auteur des livres "Raspberry Pi 4" et "Scratch et Raspberry Pi" parus aux Editions ENI*

GPS USB avec le Raspberry Pi

Aujourd'hui le GPS en tant qu'appareil autonome a plus ou moins disparu, distancé par les GPS embarqués et par les applications sur les smartphones. Monter son propre système GPS est un moyen de découvrir ce système et les informations qu'il transmet. Le Raspberry Pi est tout à fait capable de gérer le GPS et l'affichage de la carte, c'est lui que j'ai retenu pour cette présentation.

Le GPS, comment ça marche ?

Principe

Le système GPS repose sur une constellation de satellites en orbite autour de la Terre (1). Chaque satellite embarque plusieurs horloges atomiques (4 pour Galileo). Chaque satellite transmet des informations permettant de le localiser précisément dans l'espace ainsi que l'heure exacte de l'émission du message. Avec ces informations, le récepteur mesure le temps que met le signal pour lui parvenir. Il peut donc calculer la distance précise qui le sépare du satellite.

Avec un seul signal reçu d'un satellite, le récepteur sait uniquement qu'il est positionné sur un cercle centré sur le satellite.

Avec 2 satellites, le récepteur se situe sur un des deux points d'intersection.

Avec 3 satellites, le récepteur peut déterminer le point unique où il se situe. Ici on a visualisé la position sur des cercles. Dans la réalité, le récepteur se positionne sur une sphère. L'intersection de 3 sphères correspond à 2 points. Il faudra donc un quatrième satellite pour connaître la position réelle. Le deuxième point d'intersection des sphères ne se trouve pas sur la sphère terrestre et peut aussi être éliminé par calcul. Plus le nombre de satellites reçu est élevé, plus la précision est grande. Avec plus d'informations il devient également possible de déterminer l'altitude par rapport au niveau de la mer.

Pour sécuriser son accès aux données de positionnement, chaque pays ou communauté développe son propre système : **GPS** pour les américains, **GLONASS** pour les russes, **Beidou** pour les chinois, **Galileo** pour l'Europe, etc. Des accords entre certains acteurs permettent une interopérabilité des systèmes.

Trame NMEA

La norme NMEA 0183 définit une communication série permettant de transmettre une trame à de multiples récepteurs. Le format utilisé est l'ASCII. Cette norme propriétaire est payante, mais elle a été étudiée par rétro-ingénierie et à partir de sources publiques. Le récepteur reçoit les informations des satellites, les traite, et génère des trames NMEA contenant les informations qu'il a extraites/calculées. Ce sont ces trames qui sont envoyées en série via le port USB par le récepteur.

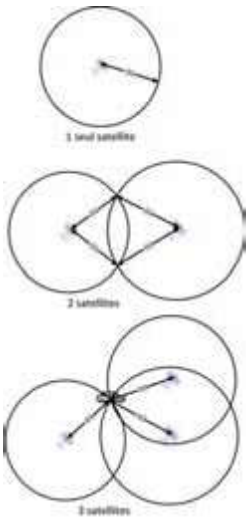
Il existe différents types de trames (une trentaine) chaque type est défini par les premiers caractères émis. Par exemple **$GPGGA**, … indique une trame GPS de type GGA. Les autres acteurs possèdent leurs propres identifiants : GA = Galileo, BD ou GB = Beidou, GL = GLONASS ; GN = Signaux mixtes GPS+GLONASS.

Exemple de trame NMEA

Voici un exemple de trame NMEA reçue avec ce GPS :
$GNGGA,145144.00,4648.64138,N,00426.90498,E,1,10,1.20,389.8,M,47.1,M,,*43

- GNGGA : Type de trame : GGA (GPS Fix et Date)
- 145144.00 : Trame envoyée à 14 h 51 min 44 s (heure UTC)
- 4648.64138, N : Latitude 46,4864138° Nord
- 00426.90498, E : Longitude 4.2690498° Est
- 1 : Type de positionnement 1 signifie GPS
- 10 : Nombre de satellites utilisés pour calculer les coordonnées
- 1.20 : Précision horizontale (où 1 est la valeur optimale, de 2 à 3 précision excellente, de 5 à 6 bonne, au-delà de 8 mesures non acceptable)
- 389.8, M : Altitude du récepteur en mètres
- …,…,… : D'autres informations peuvent figurer ici
- *43 : Somme de contrôle de parité

D'autres trames contiennent l'heure, la latitude et la longitude, ainsi que la vitesse et la direction de déplacement. C'est l'ensemble des informations récupérées dans ces trames qui est utilisé par les logiciels GPS.



1

Récepteur GPS

Choix du récepteur

J'ai opté pour un GPS miniature GN803-G à moins de 15€, trouvé en ligne [1] mais d'autres modèles conviendront. On trouve des GPS USB de 10€ à plusieurs centaines d'euros, en fonction de la puissance de calcul, de la précision, de la résistance aux intempéries… 2

Le récepteur contient l'ensemble du matériel, antenne et récepteur, dans un boîtier quasiment circulaire. Une LED clignote toutes les secondes, elle indique le verrouillage du récepteur sur les satellites. Le boîtier a une épaisseur de 20mm 3. Il est équipé à la partie inférieure d'un aimant puissant permettant de coller le récepteur sur la carrosserie d'une voiture. Une protection souple évite les rayures. Le câble USB d'un mètre cinquante est suffisant pour amener le signal dans l'habitacle, ou pour poser le récepteur sur un rebord de fenêtre si vous êtes chez vous. La sensibilité du récepteur est suffisante pour faire les tests à l'intérieur, avec la possibilité d'obtenir une précision dégradée.

2

3

Mise en service du GPS

Installez la dernière version Desktop complète de Raspbian et mettez-la à jour. Démarrez le Raspberry Pi. Quand le bureau est affi-

ché, Connectez le récepteur GPS sur un port USB. Ouvrez un terminal et tapez **dmesg** :

```
[ 705.861298] usb 1-1.1: new full-speed USB device number 6 using xhci_hcd
[ 705.995790] usb 1-1.1: New USB device found, idVendor=1546, idProduct=01a8, bcdDevice= 3.01
[ 705.995807] usb 1-1.1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[ 705.995820] usb 1-1.1: Product: u-blox GNSS receiver
[ 705.995833] usb 1-1.1: Manufacturer: u-blox AG - www.u-blox.com
[ 706.000716] cdc_acm 1-1.1:1.0: ttyACM0: USB ACM device
```

Vous verrez apparaître les lignes ci-dessus prouvant que le GPS a bien été détecté (u-blox GNSS receiver) et identifié. La commande lsusb vous affiche cette ligne :

```
Bus 001 Device 006: ID 1546:01a8 U-Blox AG [u-blox 8]
```

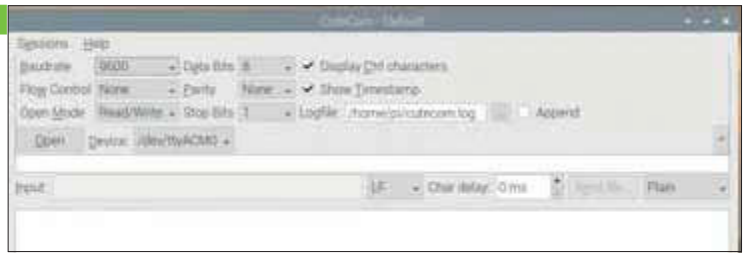
Qui confirme l'identification de l'appareil par le système.

Installer un terminal série

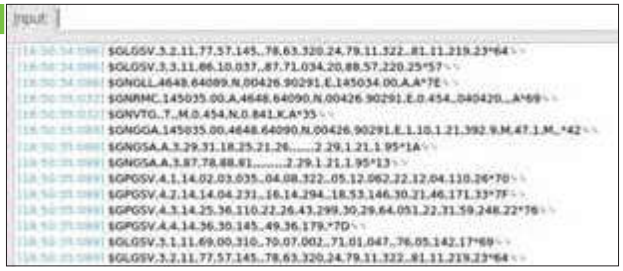
Un terminal série peut afficher les trames NMEA mais Raspbian n'en propose pas par défaut, il faut en installer un. En mode graphique **CuteCom** est une solution légère et assez conviviale. Dans un terminal tapez

```
sudo apt-get install cutecom
```

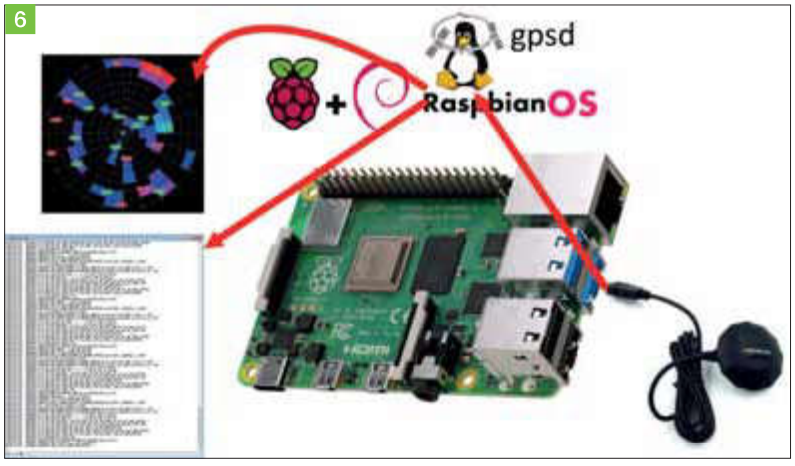
4



5



6



A la fin de l'installation, vous pouvez lancer CuteCom en tapant cutecom dans une console ou via le menu principal dans lequel une rubrique **Outils système** est apparue. Vous y trouverez une icône **CuteCom**. La fenêtre de CuteCom s'ouvre, cliquez sur **Settings** pour configurer la liaison série.

4 Dans la fenêtre **Settings** réglez la vitesse à 9600 bits/s et choisissez le **Device** correspondant à la connexion de votre GPS, ici */dev/ttyACM0*. Cliquez sur le bouton **Open** pour commencer à recevoir les trames NMEA.

La fenêtre du terminal va faire défiler les trames reçues. On peut voir (5) que les différents types de trames sont présents. Si les données n'apparaissent pas, n'allez pas plus loin. Il faut impérativement recevoir les trames NMEA avant de passer à la suite.

Installation de gpsd

Gpsd, c'est quoi ?

gpsd est un démon (ou daemon : un programme lancé automatiquement au démarrage du système et qui s'exécute en arrière-plan). gpsd reçoit les données d'un ou plusieurs récepteurs GPS ou AIS connectés à un ordinateur hôte par des ports série ou USB, fournissant des informations sur la position, le parcours et la vitesse des récepteurs connectés. Il peut être interrogé sur le **port TCP 2947** de l'ordinateur hôte. On voit sur 6 le GPS à droite. Les informations sont transmises à gpsd via le port USB. Deux consommateurs viennent lire les données GPS pour les afficher sous forme graphique (répartition des satellites dans le ciel) ou en mode texte (trames NMEA).

Installer gpsd

Dans un terminal saisissez la commande suivante :

```
sudo apt-get install gpsd gpsd-clients
```

Une fois le logiciel installé, on passe à la configuration. Vous pouvez faire une copie de sauvegarde du fichier avant de le modifier, par sécurité.

```
sudo nano /etc/default/gpsd
```

Modifiez les lignes du fichier suivant ces indications :

- **START_DAEMON="true"** signifie que vous voulez que **gpsd** se lance au démarrage comme daemon (service du système) ;
- **USBAUTO="true"** permet de connecter automatiquement les récepteurs GPS à **gpsd** quand on les connecte sur un port ;
- **GPSD_OPTIONS="-n"** si vous souhaitez que le GPS recherche les satellites à la mise en route pour synchroniser l'heure ;
- **DEVICES="/dev/ttyACM0** dépend de votre système. C'est le même nom que celui que vous avez trouvé avec **CuteCom** à l'étape précédente.

Sauvegardez les modifications avec CTRL X, puis rebootez Raspbian.

Tester gpsd

Parmi les clients de ***gpsd,*** nous avons installé ***gpsmon*** qui est un moniteur permettant de se connecter à ***gpsd*** et de vérifier que tout fonctionne bien, en mode texte. Tapez ***gpsmon*** dans un terminal en ligne de commande. Vous devriez obtenir une fenêtre présentant les informations GPS sous forme de texte (7).

7 Si vous avez un écran comme celui-ci tout va bien, vous pouvez continuer. Si l'écran est vide ou si vous avez des erreurs qui s'affichent il va falloir mettre tout ça d'aplomb avant de continuer.

Logiciel de navigation Navit

Navit, c'est quoi ?

Navit est un **logiciel libre** de navigation routière (ou de guidage routier) sous Linux et autres systèmes d'exploitation. Il est capable d'utiliser les données OSM (**OpenStreetMap**). Il est non seulement **compatible Linux** mais il est aussi utilisable sur un certain nombre d'appareils mobiles sous Android ou Windows. Ce projet utilise un fichier ".bin" généré à partir des données cartographiques d'OSM. Il peut ainsi effectuer un rendu de la carte en temps réel. Navit est capable d'effectuer des recherches par adresse, de vous guider en temps réel, de recalculer l'itinéraire en cas d'erreur. Consultez le site du projet [3] pour plus d'informations. (d'après OpenStreetMap [4]).

Navit n'est pas le logiciel de navigation le plus évolué, il a l'avantage d'exister sur le Raspberry Pi et d'être facile à installer. Des pages d'aide en ligne existent en français [7].

Installer Navit

Le logiciel Navit est disponible dans le dépôt Raspbian. Pour l'installer il suffira de saisir la commande suivante en ligne de commande :

```
sudo apt-get install navit
```

Pour exécuter Navit vous pouvez simplement taper navit en ligne de commande, vous retrouverez également une icône Navit dans la rubrique **Accessoires** du menu principal. Une fois lancé, déception ! On obtient un écran vide avec un point rouge qui indique la position du récepteur GPS… Et la carte n'apparaît pas. En fait, il va falloir télécharger la carte qui nous intéresse puis indiquer à Navit de l'utiliser. Attention, plus la zone que vous sélectionnerez sera grande, plus le fichier sera encombrant !

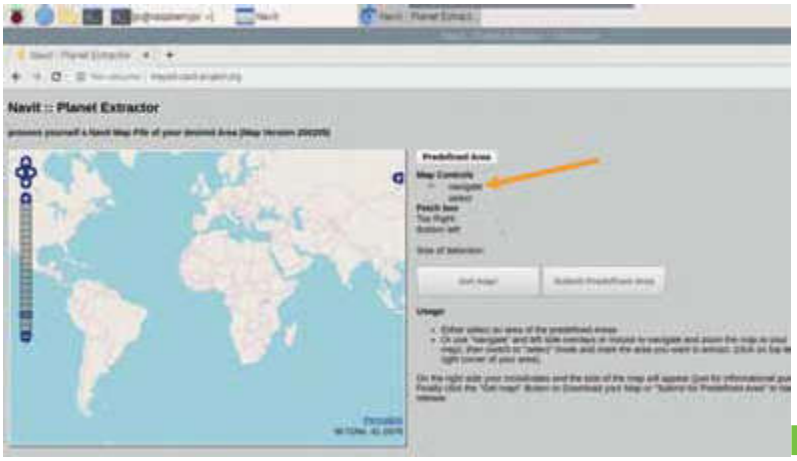
Télécharger la carte

Dans le navigateur web de Raspbian, rendez-vous sur le site de téléchargement de Navit [5].

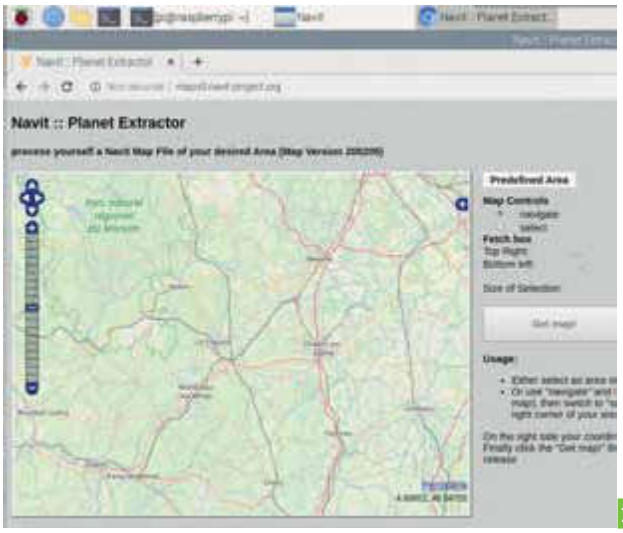
Le bouton **Navigate** est coché (9), il va permettre de choisir la zone qui vous intéresse. Utilisez **la molette de la souris** (Zoom) et le **clic gauche** (déplacement) pour choisir une zone.

Si vous souhaitez récupérer une zone déjà sélectionnée, passez par le bouton **Predifined Area**.

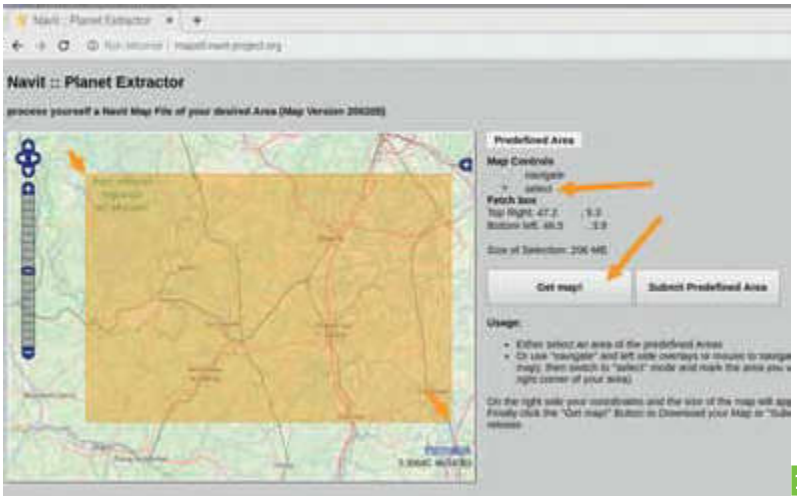
Lorsque vous êtes satisfait(e) de la zone affichée, cliquez sur le bouton **select**, situé sous le bouton **navigate** pour "couper" la partie de la carte que vous voulez utiliser. (10)



9



10



11

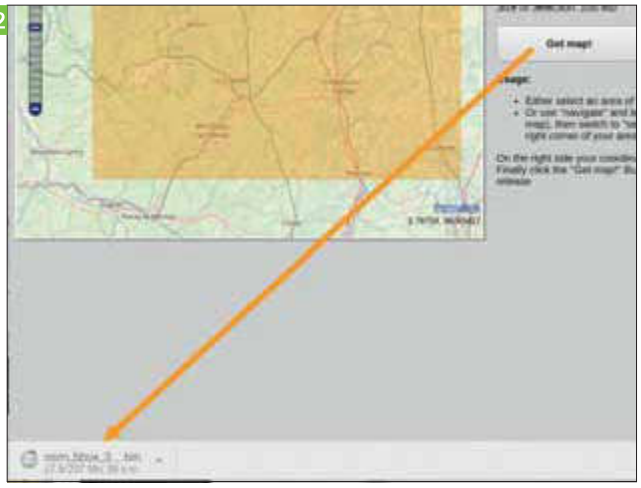
Avec la souris sélectionnez la zone à récupérer (clic gauche et tracer le rectangle). Quand la zone est définie, cliquez sur le bouton **Get map!** (11)

Le fichier ***osm_bbox_3….bin*** est téléchargé (voir en bas à gauche de la fenêtre). Attendez la fin du téléchargement. (12)

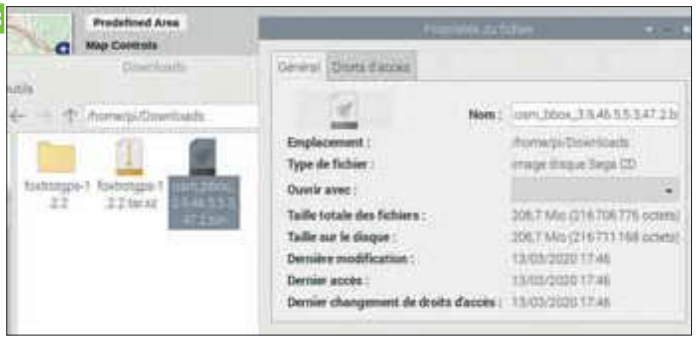
Le fichier est disponible dans le dossier **Downloads** (13).

Pour la zone que j'ai sélectionnée, il représente 207Mo environ. Il reste à indiquer à Navit que c'est cette carte qu'il doit utiliser. Avant

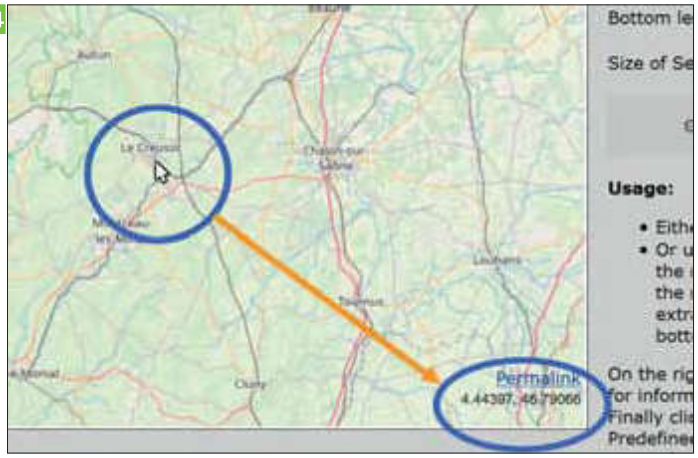
12



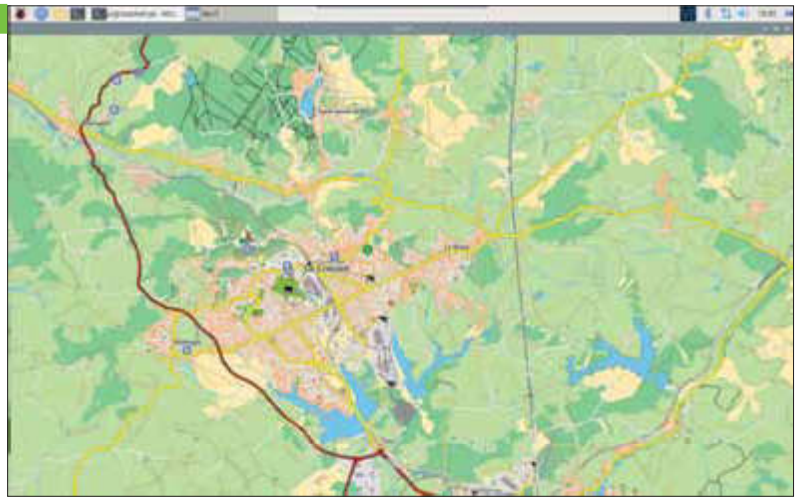
13



14



8



de continuer j'ai renommé la carte **creusot.bin** pour faciliter les choses.

Pour le centrage de la carte, placez la souris sur l'endroit que vous voulez voir apparaître au centre de la fenêtre GPS, et relevez les coordonnées en bas à droite. (14)

Sauvegardez le fichier */etc/navit.xml* avant de le modifier avec sudo nano /etc/navit.xml

Recherchez la ligne **navit center** (CTRL W dans nano) Dans **navit.xml**, reportez les coordonnées du centre de la carte dans cette ligne :

```
<navit center="4.44397 46.79066" zoom="256" tracking="1" orientation="-1"
recent_dest="250" drag_bitmap="0">
```

Ensuite recherchez la partie qui permet d'utiliser la carte openStreetMap et modifiez comme suit

```
<mapset enabled="no">
   <xi:include href="$NAVIT_SHAREDIR/maps/*.xml"/>
</mapset>
<!– Mapset template for openstreetmaps -->
<mapset enabled="yes">
   <map type="binfile" enabled="yes" data="/home/pi/Downloads/creusot.bin"/>
</mapset>
```

Il n'y a plus qu'à relancer **Navit** et il affichera la carte que vous avez choisie. (8)

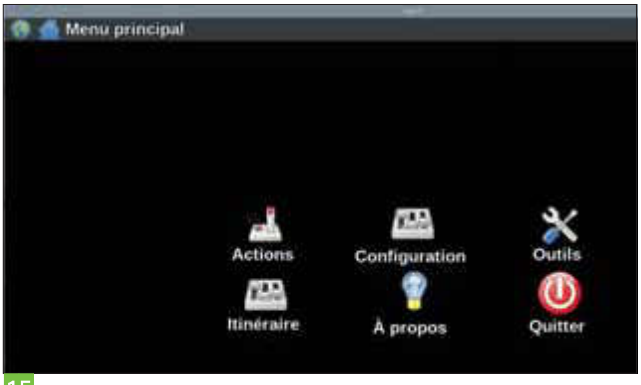
Comme prévu, la carte est centrée sur Le Creusot. Vous pouvez aussi changer d'autres paramètres, mais je vous laisse découvrir tout ça ainsi que l'utilisation de **Navit** avec **OpenStreetMap**.

Utiliser Navit 15

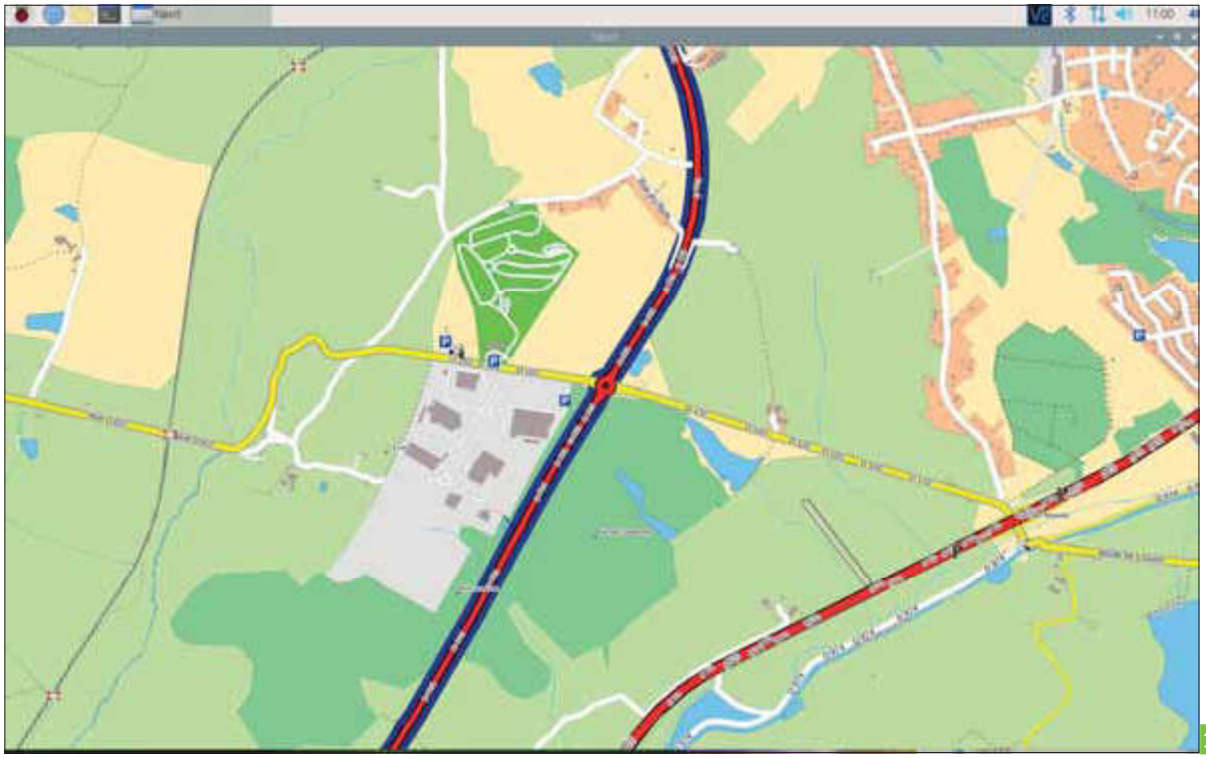
Dans **Navit** tapez sur la touche **Entrée**, vous accédez au menu principal de l'application. Vous obtiendrez des informations détaillées sur l'utilisation de Navit en consultant le wiki [en anglais] de l'application [6].

Actions vous permet de définir le point de départ (**Définir comme position**). En cliquant sur l'icône **Ville** vous pouvez saisir le nom de la ville au clavier ou sur un écran tactile. Le tiret (obligatoire pour les noms de villes composés) est disponible dans la partie **123** du clavier virtuel.

Saisissez ensuite l'adresse d'arrivée et entrez-la avec **Définir comme destination**. Navit calcule l'itinéraire (de quelques secondes à quelques dizaines de secondes), et l'affiche sous forme d'un trait bleu épais (16). Il vous appartiendra de suivre cet itinéraire. Si vous prenez une route différente, l'itinéraire sera recalculé.



15



16

L'icône **Itinéraire** permet de revenir à l'affichage complet du trajet, ou d'afficher une description de l'itinéraire en mode texte.
Navit utilise les cartes openStreetMap (OSM) et donne accès aux POI (points d'intérêts) figurant dans OSM, comme les parkings, stations essence, pharmacies etc.

Conclusion

La mise en œuvre d'un logiciel GPS sur le Raspberry Pi est relativement facile et les paquets sont disponibles sans avoir à compiler quoi que ce soit. Sans être aussi performant qu'un GPS du commerce, Navit est une solution GPS maker. On privilégiera un Raspberry Pi 4 si la vitesse de réaction est un critère important. Mais cette solution fonctionne également sur des Raspberry Pi plus anciens… avec un peu de patience.
Pour les Raspberry Pi isolés, non connectés à un réseau, le GPS peut se révéler intéressant pour fournir une heure précise (à quelques restrictions près) au système [9].
D'autres solutions existent, en particulier pour les marins. Vous en aurez un aperçu dans l'article [8] sur le blog framboise314.fr.

Sources

[1] https://fr.aliexpress.com/item/32951942245.html
[2] https://fr.wikipedia.org/wiki/NMEA_0183
[3] https://www.navit-project.org/
[4] https://wiki.openstreetmap.org/wiki/FR:Navit
[5] http://maps9.navit-project.org/
[6] https://navit.readthedocs.io/en/latest/
[7] https://sebsauvage.net/wiki/doku.php?id=navit
[8] https://www.framboise314.fr/un-gps-usb-pour-le-raspberry-pi/
[9] https://www.framboise314.fr/lheure-gps-sur-votre-raspberry-pi/

1 an de Programmez!

ABONNEMENT PDF :
35 €

Abonnez-vous sur :
www.programmez.com