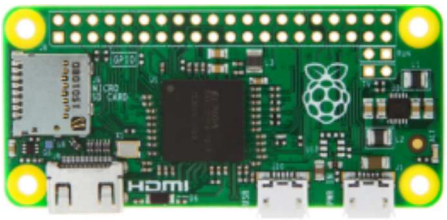
raspberry-pi.developpez.com

Raspberry Pi - Réalisation d'un traceur GPS

TheMagPi

12-15 minutes



Réalisation d'un traceur GPS

NDLR : le Raspberry Pi Zero est un nano-ordinateur monocarte de dimension 65 mm x 30 mm conçu dans le cadre de la Fondation Raspberry Pi. Cet ordinateur est destiné à encourager l'apprentissage de la programmation, mais ses performances (processeur ARM cadencé à 1 GHz, 512 Mo de mémoire et un lecteur de carte microSD pour l'espace de stockage) au regard de sa taille minuscule le destinent aussi à des applications embarquées et connectées. À l'annonce officielle de sa disponibilité le 26 novembre 2015, son prix était fixé à 5 $.

Grâce à quelques composants du commerce et un Raspberry Pi Zero, vous pouvez créer à moindre coût un petit système de journalisation de données de localisation GPS à emporter lors de vos randonnées, vos sorties en kayak ou en voiture. Et comme le Raspberry Pi est un ordinateur complet, vous pouvez même le connecter à un écran, un clavier et une souris de façon à visualiser vos déplacements sur une carte et analyser les données que vous avez collectées.

Dans ce projet, vous apprendrez comment les dispositifs GPS communiquent les informations de localisation à travers une liaison série, et comment on peut traiter ces données en Python et les sauvegarder dans un fichier.

12 commentaires ⭐⭐⭐⭐⭐

Article lu 26825 fois.

Matériel nécessaire

- Récepteur GPS USB (NDLR : le modèle GlobalSat BU-353 testé par l'auteur de l'article peut être trouvé aux alentours de 35-40 €)
- Batterie USB (NDLR : alimentation 5 V @ 1 ou 2 A, de 30 à 60 € selon la capacité de la batterie)
- Boîtier étanche (NDLR : le boîtier Pelican 1010 de l'article est à 30 € environ, mais pour ce prix-là, vous avez un boîtier réputé solide, étanche à la poussière et à l'eau)
- Adaptateur microUSB - USB (NDLR : ce genre de connectique est facile à trouver, on trouve aussi cet adaptateur dans les kits pour débuter avec le Pi Zero)
- Logiciel GpsPrune

Comme vous le savez sans aucun doute, la technologie GPS utilise des satellites en orbite autour de la Terre afin de déterminer votre localisation géographique avec les coordonnées en longitude et latitude. Donc, pour travailler sur un tel projet, il faut que le récepteur GPS ait un accès dégagé au-dessus de lui. À l'intérieur de chez vous, vous devrez sans doute vous mettre près d'une fenêtre et positionner le récepteur GPS à l'extérieur derrière la vitre.

Le boîtier Pelican1010 comprend un mousqueton bien utile pour ce projet.

La grande majorité des récepteurs GPS devraient convenir pour ce projet, mais nous conseillons d'en choisir un avec une documentation claire comprenant les caractéristiques de la liaison série comme la vitesse de transmission en bauds (la vitesse de transmission des bits). Quand vous recherchez un récepteur en particulier sur le Net, notez sa référence et en particulier la mention « baud rate », et s'assurer que d'autres utilisateurs ont pu récupérer les données via une liaison série.

Le modèle que nous allons utiliser est un GlobalSat BU-353 transmettant ses données à un débit de 4800 bauds. La première chose à faire avec le récepteur GPS est de jeter un œil aux données qu'il transmet et de configurer correctement le Pi Zero. Aussitôt après avoir connecté le récepteur GPS au Pi Zero, lancez la commande :

Recherchez une nouvelle entrée vers la fin du fichier log, qui devrait ressembler à quelque chose comme « pl2303 converter now attached to ttyUSB0 ». Cela signifie que le dispositif GPS est maintenant reconnu et peut être accédé via le chemin /dev/ttyUSB0. Pour configurer la vitesse de transmission, exécutez la commande stty en suivant le modèle de la ligne de commande suivante pour un débit de 4800 bauds sur le port /dev/ttyUSB0.

```
stty -F /dev/ttyUSB0 4800
```

Pour visualiser les données en provenance du récepteur GPS, exécutez la commande suivante :

Vous apercevrez une grande quantité de données, chaque ligne débutant avec un entête indiquant le type de trame comme $GPRMC, suivie de données délimitées par des virgules. Ces trames au format ASCII respectent le protocole standard américain NMEA (National Marine Electronics Association) relatif à certains types de données, y compris la position géographique. Le code devra donc lire ces trames de type RMC comprenant toutes les données dont nous avons besoin et même plus.

Voici un exemple de trame RMC reçu par mon dispositif :

```
$GPRMC,204311.602,A,3747.3392,N,12223.8954,W,0.50,324.18,061115,,,A*7A
```

Les champs de données qui nous concernent sont décrits dans le tableau ci-dessous. Les autres champs comprenant les données tels la vitesse au sol, le cap, la déclinaison

magnétique, ainsi que le dernier champ correspondant à une donnée de contrôle checksum ne seront pas utilisés dans ce projet.

Les champs-clés d'une trame NMEA RMC

Donnée	Description	Format
$GPRMC	Type de trame	-
204311.602	Heure UTC	hhmmss.ss
A	État	A (valide) ou V (non valide)
3747.3392	Latitude	ddmm.mmm
N	Indicateur de latitude	N (nord) ou S (sud)
12223.8954	Longitude	dddmm.mmmm
W	Indicateur de longitude	E (est) ou W (ouest)
061115	Date	jjmmaa

Après avoir constaté la réception des données, et ainsi confirmé la valeur de la vitesse de transmission ou le chemin d'accès au port série, appuyez sur CTRL+C pour stopper le processus.

Alors qu'il existe des bibliothèques toutes faites pour s'interfacer avec des dispositifs GPS, nous avons préféré coder le traitement des trames (parsing) NMEA par nous-même en Python, qui est particulièrement bien adapté à ce genre de traitement sur des données texte. Le traitement principal consiste donc à créer un fichier journal au format texte comprenant une succession de coordonnées en latitude et longitude séparées par une virgule (voir la boucle principale dans les dernières lignes du codeLe code Python). Vous noterez que seules les données valides sont écrites dans le journal, celles avec le statut retournant la lettre « A » (A=Active).

Le code comporte deux fonctions auxiliaires. La première prend une ligne brute de la trame NMEA en paramètre et retourne les données dans un dictionnaire Python. Cette fonction rend juste les données plus faciles à traiter au cas où vous voudriez améliorer le script de votre projet.

L'autre fonction auxiliaire, à partir des coordonnées exprimées en degrés-minutes et l'hémisphère que le récepteur GPS délivre en sortie, retourne ces mêmes coordonnées converties en degrés décimaux. Il sera plus simple de travailler avec le système décimal si vous souhaitez plus tard améliorer le projet en déterminant par exemple si les coordonnées sont situées ou non à l'intérieur d'une certaine zone. Par ailleurs, de nombreux outils de cartographie comme Google Maps utilisent le système décimal.

Éditez le fichier GPS.pyLe code Python puis adaptez éventuellement le chemin d'accès au port et le débit de transmission de votre équipement GPS.

Pour exécuter le programme au démarrage du Pi Zero, éditez le fichier rc.local :

Ajoutez et adaptez cette ligne, à placer juste avant le exit 0 :

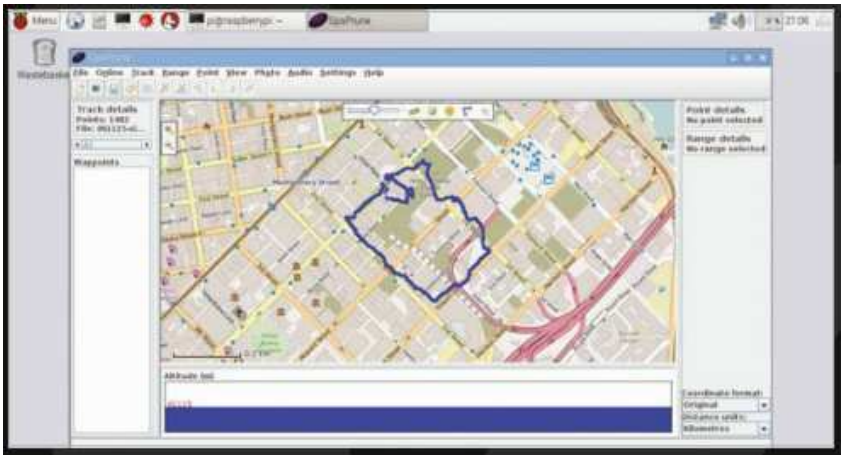
```
python /home/pi/gps_experimentation/gps.py &
```

Dès lors, quand vous mettrez en service votre Raspberry Pi Zero, le programme démarrera automatiquement et enregistrera toute position GPS valide dans le journal.

Pour visualiser les données enregistrées dans le journal, connectez le Raspberry Pi à un clavier, un écran et une souris, puis démarrez-le. Installez GpsPrune en saisissant la ligne de commande suivante :

```
sudo apt-get install gpsprune
```

Lancez le logiciel avec la commande gpsprune. Dans le menu File > Open, choisissez le fichier journal. Les options sélectionnées par défaut devraient convenir pour ce projet. Vous devriez maintenant visualiser le trajet parcouru sur une carte !



GpsPrune s'exécute sur Raspberry Pi et permet de visualiser vos données de localisation GPS sur une carte.

Le projet présenté ici ne vous montre que les principes de base, mais nous espérons qu'il vous fournira une idée plus précise sur l'exploitation de données GPS.

Vous pouvez maintenant poursuivre le projet et même l'améliorer. Par exemple, vous pouvez créer une application de « géocaching inverse », c'est-à-dire une boîte qui se déverrouillera uniquement si elle est localisée dans une zone géographique précise.

```
import serial
import os

firstFixFlag = False
firstFixDate = ""


ser = serial.Serial(
    port='/dev/ttyUSB0',\
    baudrate=4800,\
    parity=serial.PARITY_NONE,\
    stopbits=serial.STOPBITS_ONE,\
    bytesize=serial.EIGHTBITS,\
        timeout=1)


def degrees_to_decimal(data, hemisphere):
    try:
        decimalPointPosition = data.index('.')
        degrees = float(data[:decimalPointPosition-2])
        minutes = float(data[decimalPointPosition-2:])/60
        output = degrees + minutes
        if hemisphere is 'N' or hemisphere is 'E':
            return output
        if hemisphere is 'S' or hemisphere is 'W':
            return -output
    except:
        return ""



def parse_GPRMC(data):
    data = data.split(',')
    dict = {
            'fix_time': data[1],
            'validity': data[2],
```

```
            'latitude': data[3],
            'latitude_hemisphere' : data[4],
            'longitude' : data[5],
            'longitude_hemisphere' : data[6],
            'speed': data[7],
            'true_course': data[8],
            'fix_date': data[9],
            'variation': data[10],
            'variation_e_w' : data[11],
            'checksum' : data[12]
    }
    dict['decimal_latitude'] =
degrees_to_decimal(dict['latitude'],
dict['latitude_hemisphere'])
    dict['decimal_longitude'] =
degrees_to_decimal(dict['longitude'],
dict['longitude_hemisphere'])
    return dict


while True:
    line = ser.readline()
    if "$GPRMC" in line:
        gpsData = parse_GPRMC(line)
        if gpsData['validity'] == "A":
            if firstFixFlag is False:
                firstFixDate = gpsData['fix_date'] + "-"
gpsData['fix_time']
                firstFixFlag = True
            else:
                with open("/home/pi/gps_experimentation/" +
firstFixDate +"-simple-log.txt", "a") as myfile:
                    myfile.write(gpsData['fix_date'] + "," +
gpsData['fix_time'] + "," + str(gpsData['decimal_latitude']) +
"," + str(gpsData['decimal_longitude']) +"\n")
                with open("/home/pi/gps_experimentation/" +
firstFixDate +"-gprmc-raw-log.txt", "a") as myfile:
                    myfile.write(line)
```

Cet article est une traduction de l'article paru dans le n° 40 du magazine TheMagPi, sous le titre Zero GPS Logger.

Nous remercions les membres de la Rédaction de Developpez pour le travail de

traduction et de relecture qu'ils ont effectué, en particulier :

- f-leb
- Lolo78

Merci également à Claude Leloup pour sa relecture orthographique.

Vous avez aimé ce tutoriel ? Alors partagez-le en cliquant sur les boutons suivants :