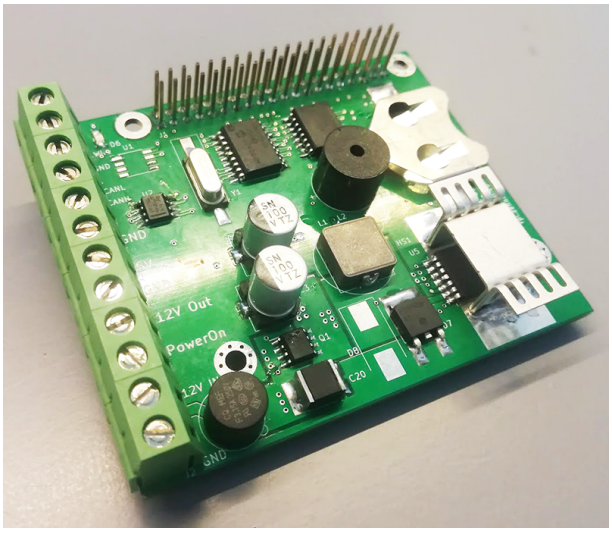
Carte mezzanine pour raspberry Pi C

1) Présentation

Cette carte est destinée à embarquer un Raspberry Pi dans un environnement autonome, et communiquant via un BUS CAN. Le cas typique d'utilisation est dans une voiture.
Les points forts que le mets en avant sont :

- Alimentation robuste 9V – 28 V en entrée, 5V en sortie
- Filtrage de la tension d'entrée
- Gestion intelligente du ON/OFF
- Sortie 12V commutée
- Horloge temps réelle (RTC)
- BUS CAN
- Buzzer
- OneWire



2) Historique

Date	Ver. Doc	Ver. carte	Changements
21 06 17	1.0	A	Version initiale
01 06 18	1.1	A	Reprise du document (modèle)
01 01 19	2.0	B	Nouvelle version de carte
01 01 2024	3.0	C	Nouvelle version de carte

3) Contact

Moi directement : J. Zéhnné ; jihzed@gmail.com

Site web : http://rpi.zehnne.fr

FaceBook : cherchez mon adresse mail, ça doit marcher !

4) Table des matières

5) Fonctions

a) Filtrage de l'alimentation

L'alimentation est filtrée par 2 condensateurs 220nF 63V auto cicatrisants PET, un fusible 3,15A et une diode TVS 1500W 24V protégeant la carte des surtensions et inversions de polarité.

b) Power Management :

- Alimentation +12V commutée par un signal sur l'entrée APC ou BP.
- L'alimentation 5V démarre dès que le +12V est commuté.

- Possibilité d'utiliser la RTC pour programmer un démarrage à une date souhaitée.
- 2 entrées de démarrage sur 12 V nommées « PowerON » sur la carte, pour brancher par exemple un bouton (BP) et le +12V après contact (APC).
- Relecture de l'état des entrées de démarrage « PowerON » via la broche 37 du GPIO.
- Le raspberry maintient son alimentation avec le signal UART TXD (broche 8) ou avec la GPIO 21 (broche 40) (configurable via la goutte GS5).
- Lorsque le raspberry s'éteint, l'alimentation commutée se coupe entrainant la coupure du 12V commuté et du 5V.
- Sortie 12V commutée disponible sur le connecteur pour alimenter des sous-ensembles avec le Raspberry en 12V. Ces sorties ne consomment pas sur l'alimentation 5V du Raspberry, mais sont néanmoins protégées par le fusible.
- Fusible 3,15 A rapide (par défaut, peut être remplacé par un fusible 5A max).

c) DC/DC converter 12V vers 5V, 4A :

- technologie Buck basé sur un ROHM BD90640HFP
- tension d'entrée : 9 à 28V
- tension de sortie : 5V
- courant de sortie : 4A max permanent, protection thermique.
- présence d'un connecteur 5V et GND pour alimenter un périphérique (par exemple écran). La consommation totale Raspberry + périphériques ne doit pas excéder 4A continu.

d) Can Bus Controleur :

- Can Contrôleur basé sur le MCP2515 de Microship.
- La mise à niveau matérielle est assurée par un SN65HVD230D de T.I.
- Si les fils sont longs, il est possible d'activer la résistance de terminaison et 120Ω en fermant GS1 avec une goutte de soudure (fermée par défaut).

e) RTC (horloge temps réel) :

- RTC basée sur DS3231SN# de Dallas.
- RTC alimentée par une pile au format 2032. La durée estimée de la pile est 3 ans.
- Possibilité de mettre une batterie 3V en fermant la goutte GS2 avec une goutte de soudure.
- Possibilité de programmer la RTC pour démarrer l'alimentation principale. Voir doc du Dallas DS3231 ou me contacter. Il faut fermer la goutte GS4 avec une goutte de soudure pour utiliser cette fonction.

f) Buzzer :

- un buzzer est commandé par la GPIO 20 (broche 38) du Raspberry Pi (pour une alarme par exemple)

g) OneWire

- Une sortie OneWire est disponible sur le connecteur J4-4. La résistance de pullup est déjà mise en place (il est donc inutile de l'ajouter).

6) Raspberrys compatibles

À priori tous les modèles avec port GPIO 40 points : Pi B, B+, 2B, 3B, 4B.

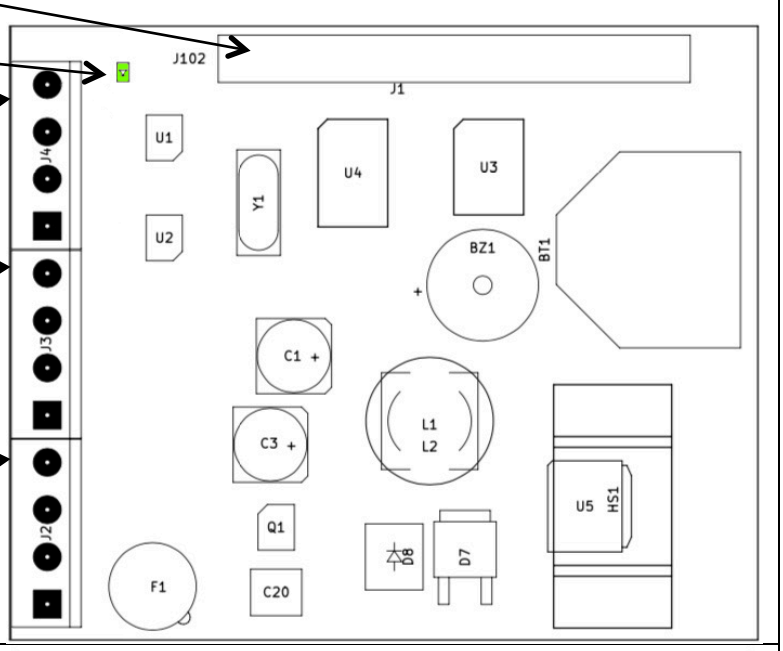
7) Branchement :

J1 : à connecter sur le Raspberry Pi
Led verte de présence 5V

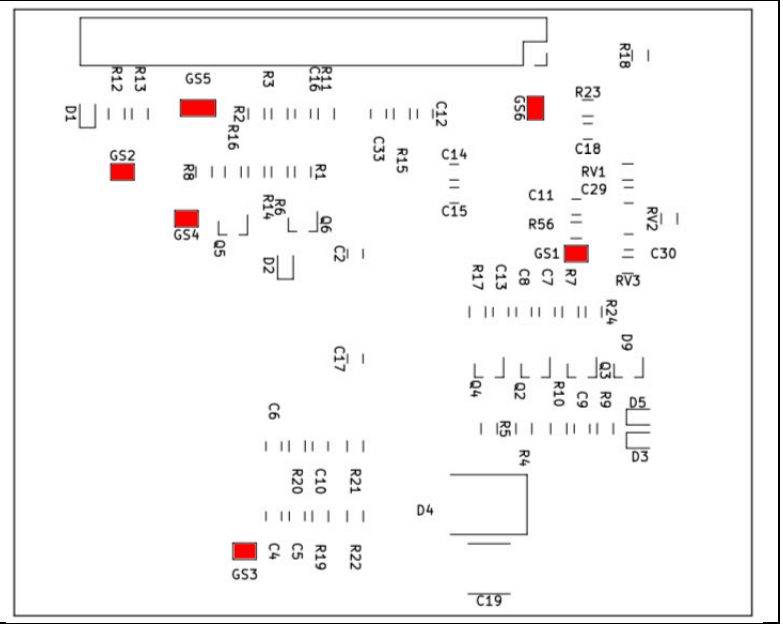
J4-4 : OneWire
J4-3 : GND
J4-2 : CANL
J4-1 : CANH

J3-4 : GND
J3-3 : 5V
J3-2 : GND
J3-1 : sortie 12V commutée

J2-4 : clé de commutation
J2-3 : clé de commutation
J2-2 : +12V batterie
J2-1 : GND



GS1 : résistance de terminaison CAN de 120Ω.
GS2 : charge d'un accumulateur à la place de la pile.
GS3 : doit rester fermé.
GS4 : démarrage programmé par RTC.
GS5 : Choix maintient de l'alimentation (signal à droite par défaut, TXD)
GS6 : Pull up de One Wire (4,7kΩ)



a) Exemple :

Branchement dans un véhicule :

- ➔ Débranchez la batterie du véhicule.
- ➔ récupérez le +12V permanent derrière un fusible, par exemple les Warning et branchez-le sur le J2-2.
- ➔ branchez GND sur une reprise de masse (vis, fixation quelconque) sur J2-1.
- ➔ Récupérez le +12 après contact (APC) sur un accessoire qui s'allume avec le contacte (par exemple l'autoradio, les essuie-glaces, …) sur J2-3.
- ➔ Branchez un bouton (BP) entre J2-2 et J2-3 (de manière à ce qu'il y a +12V lorsque l'on appuie sur le BP).
- ➔ Rebranchez la batterie du véhicule.

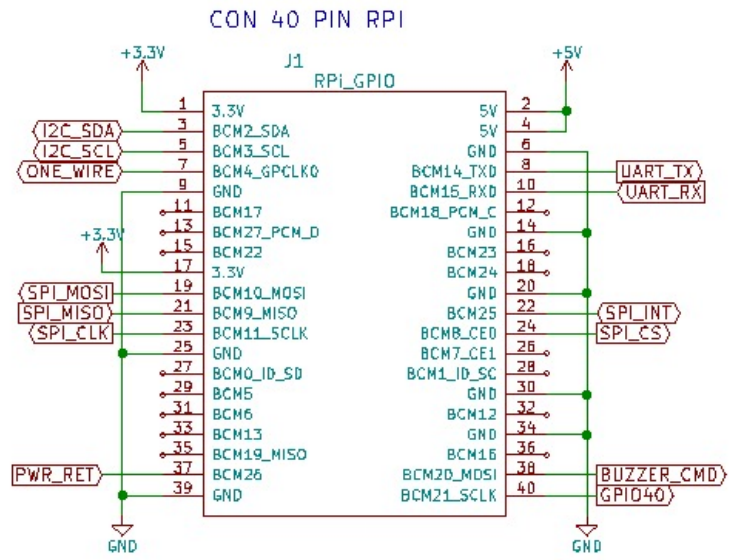
Voilà, votre Raspberry Pi est prêt à démarrer.

8) Configuration

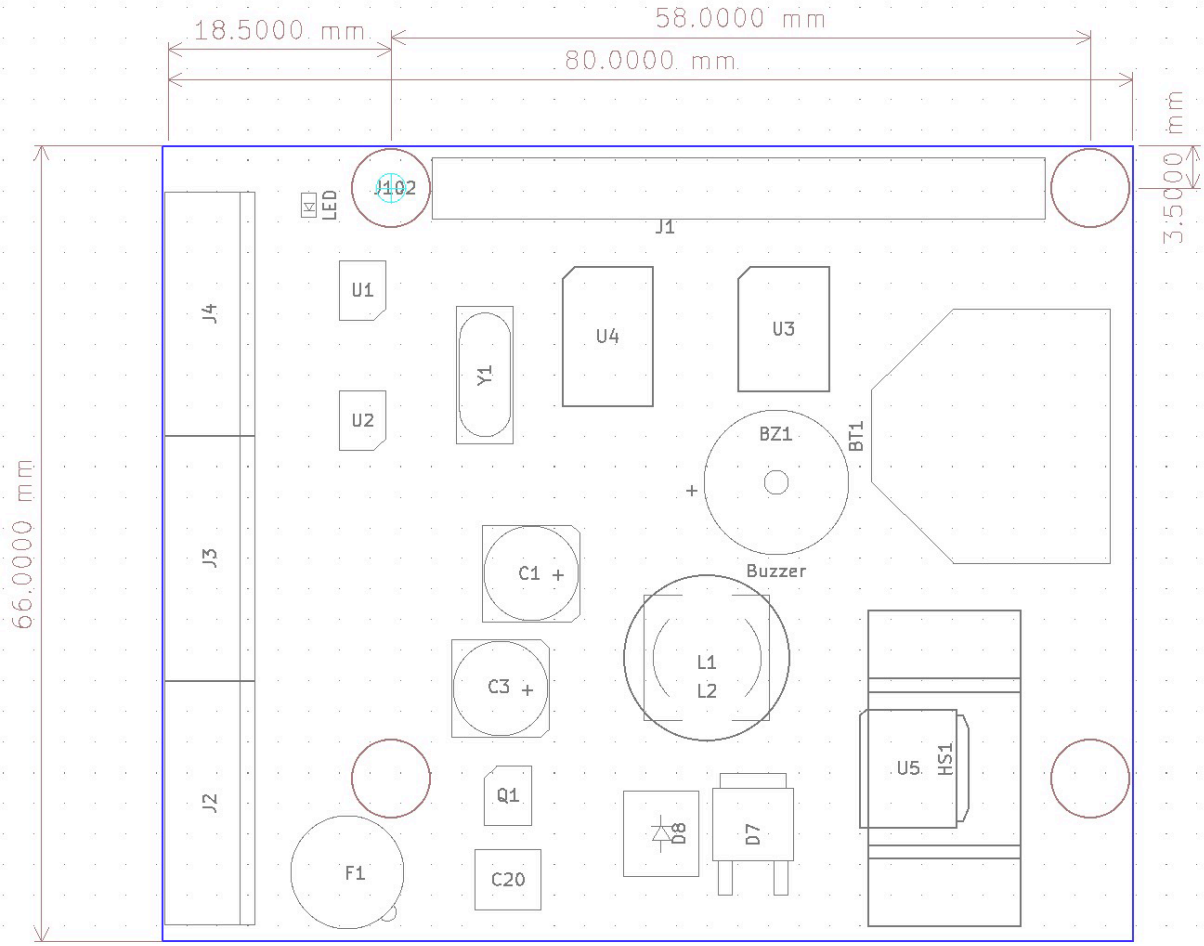
a) Changement de commande de l'auto-maintient

Par défaut, le maintien de l'alimentation est réalisé avec le signal TX du RPI. Il est possible de le réaliser avec le GPIO 21 (broche 40) du RPI. Pour cela, retirez la goutte de soudure sur GS5 et réalisez une goutte de soudure sur de l'autre côté.

9) Broches disponibles pour d'autres fonctions



10) Dimensions mécaniques



11) Annexes

a) Relecture de l'état du PowerON :

Voici le code python pour lire l'état de l'entrée PowerON. Elle peut par exemple servir à décider l'extinction du Pi et donc de l'alimentation :

```
>>> import RPi.GPIO as GPIO
>>> GPIO.setmode(GPIO.BOARD)
>>> GPIO.setup(37, GPIO.IN)
>>> GPIO.input(37)                    # bouton ON sur position ON
1
>>> GPIO.input(37)                    # bouton ON sur position OFF
0
>>>
```

b) Programmer un redémarrage sur RTC via Python

Le Linux ne sait pas (enfin je n'ai pas trouvé) programmer la sortie interruption du DS3231. Il faut donc taper directement sur le BUS I2C.
En 2 étapes : la première est de programmer et s'éteindre. La seconde, au redémarrage, et de bien penser à redescendre le flag pour ne pas vider la pile !
Ce programme nécessite la petite lib que j'ai développé, disponible sur mon site.

```
>>> import ds3231
```

```
>>> import time
>>> rtc = ds3231.ds3231()
>>> rtc.set_alarm1(1,12,0,0) # redémarrage au 1er du mois en cours, à 12h00
et 0 secondes
```

Et au redémarrage :

```
>>> import ds3231
>>> import time
>>> rtc = ds3231.ds3231()
>>> rtc.reset_alarm1()      # reset du flag pour ne pas vider la pile.
```

Bien sur, ces mêmes actions peuvent être réalisées directement sous Linux Raspbian !