Raspberry : utiliser la carte d'extension Motor Shield de SB Components

Ressources

- Site de SB Component sur le Motot Shield : http://sb-components.co.uk/motor-shield.html
- schéma de la carte Motor Shield : motor_shield_schematic.pdf
- Information sur le Pont-H L293D : https://wiki.mchobby.be/index.php?title=Pont-H_L293D

Présentation

La carte d'extension **Motor Shield** pour Raspberry Pi permet de contrôler :

- 4 **moteurs à courant continu** (DC) ou 2 **moteurs pas à pas** en utilisant le **Pont-H** de puissance moyenne (600mA) L293D qui a les caractéristiques et les fonctionnalités suivantes :
  - fournir un **courant** de sortie jusqu'à **1A** et une tension maximale de **24 V**,
  - réaliser **l'inversion** de la polarisation aux bornes des moteurs,
  - contrôler la **vitesse** du moteur à l'aide d'un **signal PWM**.
- 2 **capteurs infrarouge** (IR),
- un capteur à **ultrasons**.

Branchement des moteurs

GPIO utilisés pour les moteurs

Moteur	PWM	Avancer	Reculer
Moteur 1	17	GPIO 27	GPIO 22
Moteur 2	25	GPIO 24	GPIO 23
Moteur 3	10	GPIO 11	GPIO 09
Moteur 4	12	GPIO 07	GPIO 08

GPIO utilisés pour les flèches

Flèche	GPIO
Avancer	16
Reculer	19
Droite	13
Gauche	26

GPIO utilisés pour les capteurs

Capteur	Réception	Emission
IR 1	GPIO 04 (echo)	-
IR 2	GPIO 18 (echo)	-
Ultrasons	GPIO 6 (echo)	GPIO 5 (trigger)

Gérer les moteurs

La **gestion** d'un moteur nécessite :

- de créer un **objet PWM** pour gérer la puissance du moteur
- d'**utiliser conjointement** les deux GPIO pour définir le **sens de rotation** du moteur.

Exemple pour le moteur 1 :

```
import RPi.GPIO as GPIO
import time

# Utiliser la numérotation électronique du GPIO
```

```
GPIO.setmode(GPIO.BCM)

# définir les broches du GPIO à utiliser en sortie dans un tableau associatif
moteur1 = {"PWM":17, "Avancer":27, "Reculer":22}

# Configurer les broches en sortie
GPIO.setup(moteur1["PWM"], GPIO.OUT)
GPIO.setup(moteur1["Avancer"], GPIO.OUT)
GPIO.setup(moteur1["Reculer"], GPIO.OUT)

# creation d'un objet PWM appelé moteurPWM en précisant le numero de broche (moteur1["PWM"]) et la
frequence (50Hz)
moteurPWM = GPIO.PWM(moteur1["PWM"], 50)

# demarrage du PWM avec un cycle a 0 : moteur arrêté off
moteurPWM.start(0)

# définir le rapport cyclique à 20 pour faire tourner le moteur 1 à 20% de sa puissance
moteurPWM.ChangeDutyCycle(50)

# faire tourner le moteur dans un sens pendant 2 secondes
GPIO.output(moteur1["Avancer"],GPIO.HIGH)
GPIO.output(moteur1["Reculer"],GPIO.LOW)
time.sleep(2)
GPIO.output(moteur1["Avancer"],GPIO.LOW)
GPIO.output(moteur1["Reculer"],GPIO.LOW)

# Arreter le PWM
moteurPWM.stop()

# libérer le port du GPIO utilisé
GPIO.cleanup()
```

Gérer les flèches à LED

Pour **activer** l'éclairage d'une flèche, il suffit :

- de **définir** les broches concernées en **sortie**,
- de mettre la sortie à l'**état haut** pour activer l'éclairage.

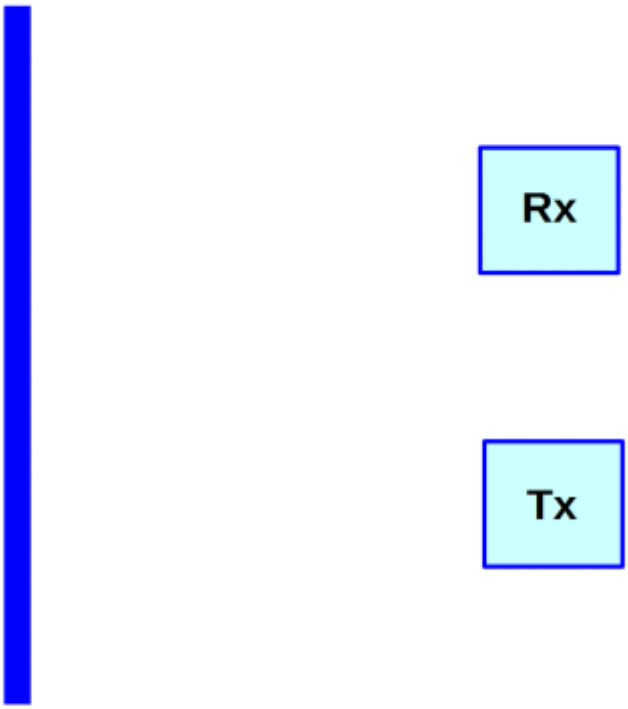
```
import RPi.GPIO as GPIO
import time

# Utiliser la numerotation electronique du GPIO
GPIO.setmode(GPIO.BCM)

# définir les broches du GPIO a utiliser en sortie dans un tableau associatif
fleche={"avancer":16, "reculer":19, "droite":13, "gauche":26}

# Configurer les broches en sortie
GPIO.setup(fleche["avancer"],GPIO.OUT)
GPIO.setup(fleche["reculer"],GPIO.OUT)
GPIO.setup(fleche["droite"],GPIO.OUT)
GPIO.setup(fleche["gauche"],GPIO.OUT)

print("Activer la flèche avancer pendant 1 seconde :")
GPIO.output(fleche["avancer"],GPIO.HIGH)
time.sleep(1)
GPIO.output(fleche["avancer"],GPIO.LOW)
time.sleep(1)

print("Activer la flèche reculer pendant 1 seconde :")
GPIO.output(fleche["reculer"],GPIO.HIGH)
time.sleep(1)
GPIO.output(fleche["reculer"],GPIO.LOW)
time.sleep(1)

print("Activer la flèche droite pendant 1 seconde :")
GPIO.output(fleche["droite"],GPIO.HIGH)
```

```
time.sleep(1)
GPIO.output(fleche["droite"],GPIO.LOW)
time.sleep(1)

print("Activer la flèche gauche pendant 1 seconde :")
GPIO.output(fleche["gauche"],GPIO.HIGH)
time.sleep(1)
GPIO.output(fleche["gauche"],GPIO.LOW)
time.sleep(1)

# libérer les ports du GPIO utilises
GPIO.cleanup()
```

Gérer le capteur à ultrasons

principe

Un émetteur d'ultrasons (Tx) envoie un train d'ondes sonores (8 impulsions à 40kHz) qui se réfléchissent sur un obstacle et reviennent vers un récepteur (Rx). Connaissant la vitesse du son dans l'air (environ 340 m/s) il suffit de diviser par 2 le temps mis par les ondes pour faire l'aller-retour et calcule alors la distance de l'obstacle.



Pour en savoir plus :

- https://www.framboise314.fr/mesure-de-distance-par-ultrasons-avec-le-raspberry-pi/
- http://espace-raspberry-francais.fr/Composants/Mesure-de-distance-avec-HC-SR04-Raspberry-Francais/

Pour **mesurer** la distance d'un obstacle on procède de la manière suivante :

- on envoie sur l'entrée **Trig** du capteur HC-SR04 un train d'onde pendant un très bref instant de 10 micro secondes (0.00001 s),
- DES QUE LE TRAIN EST EMIS, l'entrée **Echo** délivre une tensions de 5 v
- dès que l'entrée **Echo** détecte le retour du train d'onde, l'entrée n'est plus à 5V.

```
import RPi.GPIO as GPIO
import time

# Utiliser la numerotation electronique du GPIO
GPIO.setmode(GPIO.BCM)

# définir les broches du GPIO a utiliser en sortie pour envoyer le train d'onde et en entrée pour la
réception
ultrason={"envoi":5, "echo":6}

# Configurer les broches
GPIO.setup(ultrason["envoi"],GPIO.OUT)
GPIO.setup(ultrason["echo"],GPIO.IN)

# fonction qui retourne la distance d'un obtacle
def distance():
    # generation du train d'ondes ultrasonores
    GPIO.output(ultrason["envoi"], GPIO.HIGH)
    time.sleep(0.00001)
    GPIO.output(ultrason["envoi"], GPIO.LOW)
    start = time.time()

    # boucler tant que l'entree n'est pas à l'etat haut
    while GPIO.input(ultrason["echo"])==0:
       pass

    # enregistrement du temps de départ
    debutImpulsion= time.time()

    # boucler tant que l'entree n'est pas revenue à un etat bas
    while GPIO.input(ultrason["echo"])==1:
       pass

    # enregistre le temps quand l'entree n'est plus à l'état haut
    finImpulsion  = time.time()

    # calcul de la distance en cm arrondie à l'entier
    distance = round((finImpulsion - debutImpulsion) * 343*100/2,1)

    # renvoyer la valeur de la distance
    return distance

# lancer la fonction distance() jusqu'à l'appui d'une touche
while True:
   try:
      # lancement de la fonction distance() et affichage du résultat obtenu
      print(distance())

      # attendre 1 seconde avant de relancer la détermination de la distance
      time.sleep(1)

   except KeyboardInterrupt:
      # arreter le programme
      pass

# libérer les ports du GPIO utilises
GPIO.cleanup()
```

Les activités ...

Je reviens à la liste des activités.