Raspberry Pi (7.2)
- commande servomoteur -

Christophe Vardon 2018 – Tous droits réservés

Commande d'un servomoteur

Nom : **Prénom :** **Classe :** **Date :**	**Appréciation** :	**Note** : **/20**
Objectif : **Utilité** :		**durée** : 4h

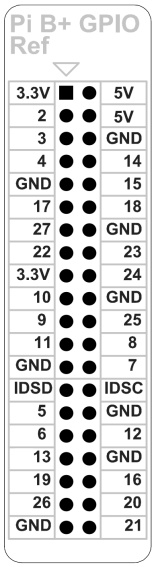
Matériel : plaque labdec – composants électroniques

Prérequis : Connexion à distance avec SSH, commande GPIO

Compétences et savoirs principalement visées :
C2-1, C2-2 (page 3a), C3-2, C3-3 (page 3b à 6)

Travail à réaliser :
-
-
-

Schéma du système :



Le servomoteur

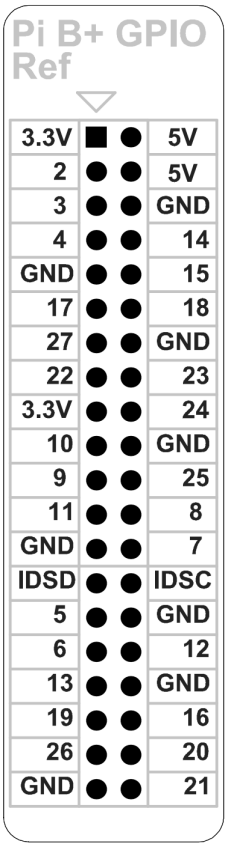
- **Question :** fais une recherche Wikipédia et complètes le cadre ci-dessous, pour expliquer ce qu'est un servomoteur, et quelles sont ses applications.

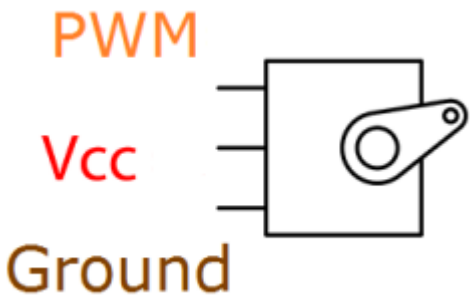
Câblage du système

AVERTISSEMENT
Attention : en cas d'erreur de branchement, ton Raspberry Pi risque d'être **détruit** !!! Ne mets pas le circuit sous tension **avant** que le professeur l'ai vérifié.

- Avant de réaliser le câblage physiquement, **dessine les connexions** (relie par des traits) des éléments ci-dessous en t'aidant du document «ANNEXE – Longruner SG90 5pcs Mini Servo Moteur 9g»

  Note : il y a **3** connexions à réaliser





- Une fois que le schéma est complet, réalise ces connexions à l'aide des câbles fournis, puis fait valider par les professeur

Programme de gestion du servomoteur

Principe de fonctionnement du servomoteur
Un servomoteur est un moteur qui permet d'obtenir une rotation à un angle déterminé. Il se commande avec un signal carré, généralement de faible fréquence (ex : 50Hz), mais de rapport cyclique variable. L' angle obtenu est proportionnel au rapport cyclique (anglais : duty cycle) du signal appliqué au moteur. Très utilisé en robotique et dans l'industrie, il peut servir à orienter un capteur, positionner un bras mécanique, ...

Choix du langage de programmation

Il n'est pas possible de réaliser ce programme avec un langage de scripting (type bash ou python), car ceux-ci ne sont pas assez rapide pour générer ou capter des impulsions de l'ordre de quelques microsecondes, comme c'est le cas ici (voir l'encadré)

Nous utiliserons donc un langage compilé rapide : le langage C ; tu trouveras le code source du logiciel à la page 6.

- Tu vas **analyser ce code** pour essayer de **découvrir** quelles valeurs sont affectées à certains paramètres ; pour cela, complètes le tableau :

Paramètres	Recopie ici la valeur et le code source correspondant
Quel GPIO est utilisé pour contrôler le servomoteur ?	
Quelles sont les valeurs d'angle min. et max. acceptées par le programme ?	
Si la valeur d'angle demandée est supérieure au max. ou inférieure au min., quelle est la valeur retenue par le programme ?	
Combien d'impulsions le programme génère-t-il au total ?	

Info

Dans le fichier programme **servo.c**, les durée de l'état haut et bas de l'impulsion sont représentés respectivement par les paramètres **pulse** et **left**.

- Analyser le code source pour en déduire les formules utilisées pour calculer **pulse** et **left** en fonction de la valeur de l'angle demandé

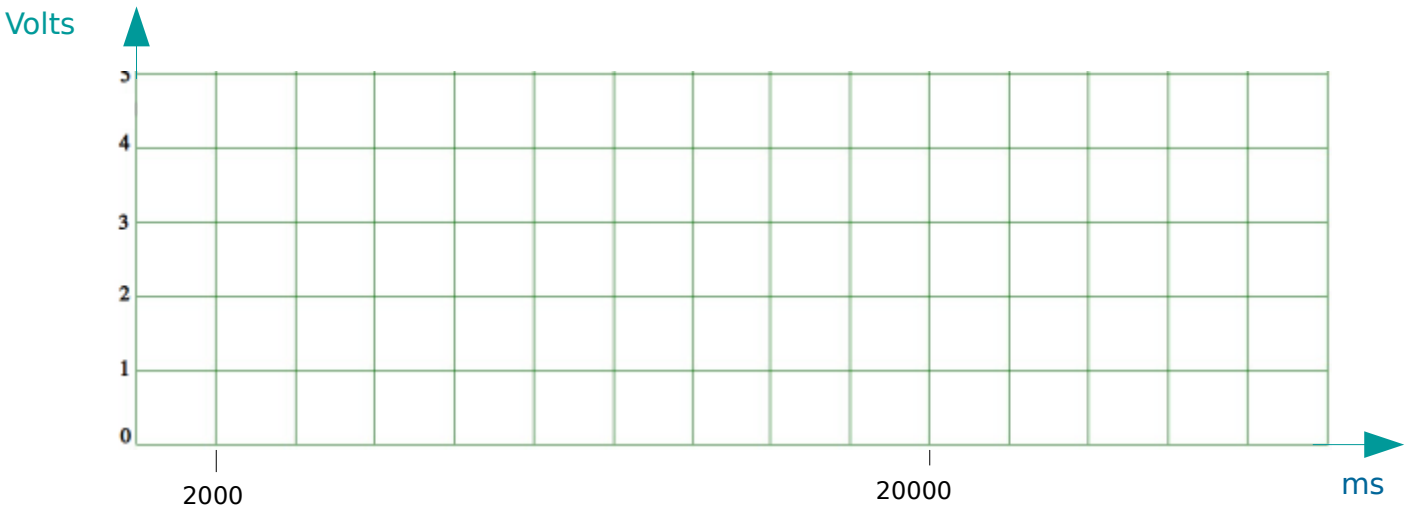
pulse =

left =

- Utilise ces formules pour compléter le tableau

angle	0°	20°	40°	60°	80°	100°	120°	140°	160°	180°
pulse										
left										

- En utilisant les données du tableau, dessine la forme de l'impulsion permettant d'obtenir un angle de 120° *(faire apparaître les unités de mesure utilisées)*



Programme servo.c

```
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
#include <wiringPi.h>


#define SERVO   0       // 17


int main (int argc, char *argv[])
{
  int angle, left, i, pulse;


  if (wiringPiSetup () == -1)
  {
    fprintf (stdout, "oops: %s\n", strerror (errno)) ;
    return 1 ;
  }

  if (argc>1)
  {
      angle=atoi(argv[1]);
      if (angle > 180 || angle<0 ) {angle=0;}
      printf("angle to send : %d\n",angle);

 } else {exit (1);}

 pinMode(SERVO,OUTPUT);
 digitalWrite(SERVO,0);

 // calcul de pulse
 pulse=(angle*9.3)+480;
 printf("pulse to send : %d\n",pulse);
 left=20000-pulse;
 i=0;

 while ( i < 35 ) {
          i++;
          digitalWrite(SERVO,1);
          delayMicroseconds(pulse);
          digitalWrite(SERVO,0);
          delayMicroseconds(left);
 }

  return 0;
}
```

- Crée nouveau fichier nommé **servo.c** et recopie le code source ci-dessus dans ce fichier (utiliser le logiciel notepad++)
- Vérifie que le codage du fichier **servo.c** est bien : **UTF-8** ; préciser ci-dessous le codage du fichier :
- Vérifie que les caractères de fin de ligne sont bien au **format UNIX (LF)** ; préciser ci-dessous le caractères de fin de ligne du fichier **servo**.c :

La compilation d'un programme en C

Compiler un programme, c'est convertir le code source en langage machine.

Le code source est écrit en langage C, lisible par un être humain ; le langage machine est le seul que comprend le microprocesseur ; le code compilé est parfois appelé « binaire ».

La compilation nécessite l'utilisation d'un logiciel appelé « compilateur ». Pour le langage C, le compilateur le plus répandu est **gcc**

- **Compile** le code source du logiciel avec la commande :

gcc -Wall -o servo servo.c -lwiringPi -lm

- Vérifie que le fichier binaire **servo** a bien été créé, sinon recommence l'opération ; s'il y a des messages d'erreur ; ceux-ci doivent t'aider à corriger le code source

Test du programme «servo»

- Pour visualiser le mouvement du servomoteur, dessine un visage sur un bout de papier et fixe-le sur l'axe (voir ci-contre)
- Lances la commande : **./servo 0**

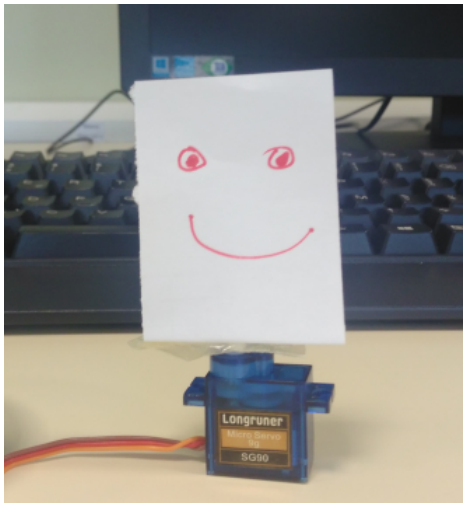
➔ Note tes observations

- Lances la commande : **./servo 90**

➔ Note tes observations

- Lances la commande : **./servo 180**

➔ Note tes observations



Remarque !

Si rien ne se passe, revérifie le câblage et le programme servo.c

- **Exercice** : écris un script «**scan.sh** », qui effectue les opérations suivantes :
  positionnement à 0° puis 30°, puis 60°, puis 90°, puis 120°, puis 150°, puis 180°, puis 120°, puis 90°, puis 60°, puis 30°, puis 0°, avec un arrêt de 1 seconde entre chaque mouvement.
  Remarque 1 *: ton script devra utiliser programme «* ***servo*** *», compilé précédemment, ainsi que la commande «* ***sleep*** *»*
  Remarque 2 *: n'oublie pas de rendre le script* ***scan.sh*** *exécutable, avec la commande appropriée*

- Testes et corrige si nécessaire ce script ; quand il est correct, recopie-le ci-dessous :

```
# !/bin/bash
```

Colle ici une photo de ton servomoteur avec son visage en papier

Script «scan.sh »

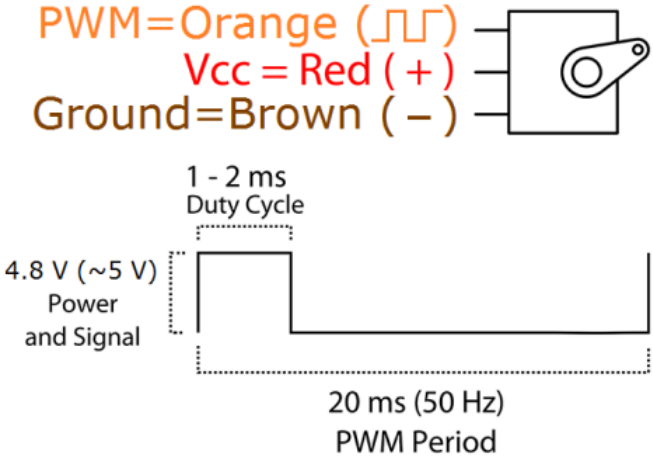
ANNEXE - Longruner SG90 5pcs Mini Servo Moteur 9g

Connexion au Raspberry Pi
Fil rouge = +5V Fil marron = GND Fil orange = **GPIO 17** (WPi : 0)

Vitesse de fonctionnement : 0,12 sec./60 ° (4,8 V sans charge)

- Stall couple (4.8 V) : 496,1 gram/in (1kg/cm)
- Tension de fonctionnement : 3,0 V ~ 7,2 V
- Plage de température : -30 à + 60
- Largeur de bande morte : 7 sec

spécification :
* j-deal sg-90 micro servo
* Tous les engrenages en nylon
* Longueur du câble : 150 mm
* Nom : 9 grammes de gouverne
* Taille : 23 mmx12.2mmx29 mm
* Poids : 9 grammes
* maximum angle : 180 °
* moment de torsion : 1.5 kg/cm
* La tension de fonctionnement : 4.2-6 V
* Plage de température : 0 ? - -55 ?:
* La vitesse de fonctionnement :
0,3 secondes/60 degrés.
* Largeur de bande morte : 10 microsecondes



Commande servo :

signal de période = 20ms (Mark-Space)

le temps Haut de 500 ms à 2200 ms règle l'angle de 0 à 180°

arrêt du signal = le servo reste fixé à sa dernière position

Gaoxing Tech. 4 pièces Arduino roue de pneu en plastique avec DC 3V 5V 6v Moteur de vitesse pour Robot DIY Robot Smart Car

Description:
Voltage: DC3-6V
Reduction ratio: 1:48
Torque: 0.8kg.cm
Tire diameter: 65mm
Tire thickness: 28mm

ANNEXE : PWM

There are four accessible hardware PWM GPIO on the Pis with the 40 pin expansion header.

However there are only two channels so GPIO12 has the same setting as GPIO18 and GPIO13 has the same setting as GPIO19.

See http://abyz.co.uk/rpi/pigpio/python.html#hardware_PWM

In addition DMA timed PWM (i.e. jitter free) may be used on all the other accessible GPIO if that's what you actually need.

in my case, RP 3 B, pwm just work on this pins
WiringPi _____BCM
GPIO.1_____(BCM 18)
GPIO.23_____(BCM 13)
GPIO.24_____(BCM 19)
GPIO.26_____(BCM 12)

```
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
#include <wiringPi.h>


#define SERVO   0       // 17


int main (int argc, char *argv[])
{
  int angle, left, i, pulse;


  if (wiringPiSetup () == -1)
  {
    fprintf (stdout, "oops: %s\n", strerror (errno)) ;
    return 1 ;
  }

  if (argc>1)
 {
          angle=atoi(argv[1]);
          if (angle > 180 || angle<0 ) {angle=0;}
          printf("angle to send : %d\n",angle);

 } else {exit (1);}

 pinMode(SERVO,OUTPUT);
 digitalWrite(SERVO,0);

 // calcul de pulse
 pulse=(angle*9.3)+480;
 printf("pulse to send : %d\n",pulse);
 left=20000-pulse;
 i=0;

while ( i < 35 ) {
          i++;
          digitalWrite(SERVO,1);
          delayMicroseconds(pulse);
          digitalWrite(SERVO,0);
          delayMicroseconds(left);
}

 return 0;
}
```