Projet Mariokart

Membres du groupe : Gentreau Amélie et Averland Martin

Fonctionnement

1. Allumer la voiture et le Raspberry (faire attention au bon branchement des piles sinon elle ne roulera pas)

2. Lancer putty sur l'adresse 10.3.141.1 (identifiant : pi, mdp : raspberry)

3. lancer un explorateur et entrer l'adresse ip du Raspberry

4. Prenez connaissances des commandes

5. Amusez vous

Nous avons décidé de faire ce projet car en cours nous avions déjà commencé à travailler sur le Raspberry. C'était aussi une occasion pour nous de découvrir un peu le monde de l'ingénierie mécatronique. De plus, les voitures autonomes sont au cœur des défis actuels d'un point de vue économique, social et environnemental. En parallèle, cela constituait aussi un défi de programmation, de mécanique et de mise en réseau. C'est ce qui nous a poussé à choisir ce projet.

Objectif *:* créer une voiture contrôlable à distance avec du wifi, mais capable d'intervenir seule s'il y a un obstacle en s'arrêtent avant de heurter ce dernier. Pour y parvenir, il faut que la voiture puisse aller dans toutes les directions, qu'elle s'arrête sans intervention humaine si elle est trop proche d'un obstacle et enfin que l'on puisse la contrôler à distance.

Cahier des charges :

Pilotage :

- touches utilisées :
  - Z : avancer
  - S : reculer,
  - D : aller à droite
  - Q : aller à gauche
  - E : freiner
  - R : accélérer
  - F : décélérer
- distance d'arrêt de 10 cm

Langages utilisés : le langage Python, le langage Javascript

Matériels utilisés :

- un Raspberry PI (Le Raspberry Pi 3 est un nano-ordinateur monocarte à processeur ARM)
- une carte d'extension SB Motor Shield pour Raspberry Pi
- une Pi camera
- 1 capteur à ultrasons
- OS Raspbian

Répartition des tâches

Noms	Planning	Tâches
Amélie	Février	-Créer un point d'accès wifi avec RaspAP
	Février/Mars	-Créer un serveur de fichier
	Mars/Avril	-Créer un intranet avec le serveur Web Lighttpd
	Avril/Mai	-Diffuser le flux vidéo de la Pi Camera avec Motion
	Mai	-Faire la page d'accueil du site, l'organiser, créer l'interaction avec la page
Martin	Février	-découverte du fonctionnement du GPIO
	Mars	-codage de la voiture (marche avant/arrière)
	Avril	-fin du codage de la voiture (direction et capteur)
	Mai	-finition du codage et mise en commun des parties (partie client, gestion de l'envoi et de la réception des messages), plus finitions de l'interface

Démarche collaborative :

Nous nous envoyons des messages et des mails pour pouvoir modifier les dossiers si besoin. Nous utilisons également le cahier numérique du projet sur le site http://cours.btssio.ac-limoges.fr.

Partie application (Martin) :

```
import RPi.GPIO as GPIO
```

(On importe une bibliothèque)

```
GPIO.setmode(GPIO.BCM)
```
(On règle le GPIO sur les numéros de ports)

```
moteur1 = {"PWM":17, "Avancer":27, "Reculer":22}
```
(On attribue un port pour les fonctions recherchées)

```
GPIO.setup(moteur1["PWM"], GPIO.OUT) GPIO.setup(moteur1["Avancer"], GPIO.OUT)
GPIO.setup(moteur1["Reculer"], GPIO.OUT)
```
(On dit que nos ports doivent être en sortie)

```
moteurPWM = GPIO.PWM(moteur1["PWM"], 50) moteurPWM.start(0)
moteurPWM.ChangeDutyCycle(dc)
```
(Le moteur tournera à 50%)

```
GPIO.output(moteur1["Avancer"],GPIO.HIGH/LOW)
```
(avancer ou non)

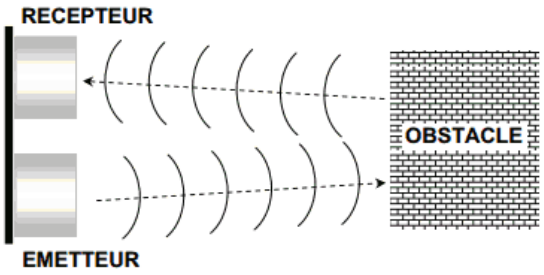
```
GPIO.output(moteur1["Reculer"],GPIO.HIGH/LOW)
```
(reculer ou non)

En adaptant ces différentes fonctions Martin a pu coder pour avancer, reculer, tourner à gauche et à droite ainsi que l'accélération/décélération, le freinage et enfin l'arrêt la voiture (fin du programme)

Bilan : Cette partie n'a pas posé beaucoup de problèmes car une fois trouvé le fonctionnement pour activer une partie il suffisait de réadapter pour autre et donc pas de nouvelles difficultés. Le seul problème a été la détection de clic non présenté car elle a été remplacée.

Distance de freinage :

En utilisant la fonction distance () fournie en annexe, nous avons pu empêcher la voiture d'avancer si la distance d'un obstacle était inférieure à 10cm :




```
async def echange(websocket,path):
```
(définir la fonction comme asynchrone)

```
envoyer=asyncio.ensure_future(gestion_envoi_message(websocket))
recevoir=asyncio.ensure_future(gestion_reception_message(websocket))
```
(Envoyer/recevoir un message)

Bilan : Bien qu'il a été difficile d'actualiser la distance sur la première version, une fonction asynchrone a permis de palier à ce problème en actualisant en permanence la distance et donc permettre un freinage automatique.

Partie serveur (Amélie) :

Point d'accès wifi sur le Raspberry :

Pour créer le point d'accès wifi Amélie a lancé le terminal de commande et a visualisé la configuration du wifi du Raspberry avant modification puis l'a éditée.

```
pi@raspberrypi:~ $ cat /etc/wpa_supplicant/wpa_supplicant.conf
(on ouvre le fichier /etc/wpa_supplicant/wpa_supplicant.conf)
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=FR

network={
        ssid="ssid_actuel"
        psk="motdepassen"
        key_mgmt=WPA-PSK
}
pi@raspberrypi:~ $
```

Avant modification

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=FR
```

Après modification, on ne garde que les trois première lignes

Pour installer le point d'accès wifi on utilise le logiciel RasAP, puis on redémarrer le Raspberry. Le logiciel RasAP permet aux clients qui se connectent sur le point d'accès Wifi d'obtenir une configuration IP.

Partie cliente (Amélie) :

Page d'accueil de la page de contrôle :

Pour créer un bouton avec le code suivant :

```
<div id="codage">
    <input type="button" name="flecheAvancer" id="flecheAvancer" value="Z" />
```

et on fait de même pour les autres en changeant ce qui se trouve après name, id et value. Pour activer celui-ci par un clic de la souris on rajoute après dans le codage :

```
var flecheAvancer = document.getElementById('flecheAvancer');
   flecheAvancer.addEventListener('mousedown',clicSouris);
   flecheAvancer.addEventListener('mouseup',relacheSouris);
```

ici aussi on répètera l'opération pour les autres boutons … .

Pour insérer la caméra dans la page dans le programme Amélie a rajouté :

```
<div id="camera">
   <fieldset>
   <legend>Vidéo en direct :</legend>
    <section class="streaming">
      <img name="PiCamera" src="http://10.3.141.1:8081/video"
      width="320" height="240" alt="Caméra Voiture"
      style="background-color: #009999" />
```

Intégration du travail

En appuyant sur la même touche (ici Z) la voiture réalise également la demande.

Bilan et perspectives

Bilan : nous avons atteint notre objectif de départ qui était l'arrêt automatique de la voiture à proximité d'un obstacle.

Amélioration possible : Au-delà de l'aspect physique de notre voiture le véritable cœur du projet était le contrôle à distance. Plus qu'une simple voiture télécommandée notre projet pourrait s'étendre à tout un réseau de voitures autonomes ou pourquoi pas quelque chose d'autre qu'une voiture puisque le contrôle à distance est applicable dans de nombreux domaines comme la médecine. Sinon il pourra rester tel quel et servir de démonstration de ce que l'on peut faire en ISN et en ICN au lycée Suzanne Valadon.

Ce que nous ont apporté ce projet et le travail en équipe :

Amélie : Ce projet m'a permis de découvrir beaucoup de chose notamment comment crée un point d'accès wifi sur un Raspberry Pi, d'utiliser Putty pour créer tout ce que j'ai fait.

Martin : Le projet en lui-même m'a beaucoup appris sur la complexité d'un programme. En commençant l'ISN, j'espérais bien créer ce genre de code ; j'ai de même pu apprendre beaucoup sur l'importance des différentes parties de l'élaboration d'un produit. Il est compliqué pour deux personnes de créer un produit aussi simple qu'une petite voiture télécommandée alors pour un projet plus élaboré il est évident que le travail d'équipe est un point clé du développement d'un produit. Nous avons aussi pu nous rendre compte de la difficulté de mettre en commun le travail de chacun. J'ai notamment développé ma capacité à travailler en autonomie à travers mes différentes recherches pour parvenir à créer notre voiture. Enfin j'ai aussi étendu mes connaissances en codage. Pour conclure, ce projet a été extrêmement enrichissant.

Annexe

Le fichier serveur.py

```
#!/usr/bin/env python3
import asyncio
import websockets
#import fonctions
from fonctions import *
 fonctions.init()
 # Gestion de l'envoi des messages du serveur au client
async def gestion_envoi_message(websocket):
    # boucle infinie pour envoyer toutes les secondes un message ;
    while True:
       await websocket.send("distance {} cm ".format(distance()))
       await asyncio.sleep(1)
 # Gestion des messages recus du client
async def gestion_reception_message(websocket):
    while True:
        # reception du message d'un client
        message = await websocket.recv()
        # lancement de deux taches en parallele
        # tache qui gere avancer, reculer, accelerer,
        # ralentir, le freinage et l'arret

asyncio.get_event_loop().create_task(gestionmoteur(message))
        # tache qui gere tourner a droite et a gauche
    asyncio.get_event_loop().create_task(gestiondirection(message))
        #print(message)
 # fonction lancee a chaque connexion d'un client
async def echange(websocket,path): # definir la fonction comme
asynchrone
    #envoyer des messages en parallele
    envoyer =
asyncio.ensure_future(gestion_envoi_message(websocket))
    #recevoir message en parallele
    recevoir =
asyncio.ensure_future(gestion_reception_message(websocket))
#recevoir message en parallele
    obstacle = asyncio.ensure_future(gestion_obstacle())
    termine, attente = await asyncio.wait(
            [envoyer, recevoir, obstacle],
            return_when = asyncio.FIRST_COMPLETED,
    )
# Definir la fonction qui sera appelee par le serveur a la
connexion d'un client
lancement_serveur = websockets.serve(echange,'10.3.141.1', 5678)
# Creation de la boucle d'evenement (event loop)
loop = asyncio.get_event_loop()
loop.run_until_complete(lancement_serveur)
loop.run_forever()
loop.close()
```

Le fichier fonctions.py

```python
import RPi.GPIO as GPIO
import time
import fonctions
import asyncio

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

moteur1 = {"PWM":17, "Avancer":27, "Reculer":22}
moteur2 = {"PWM":25 ,"Gauche": 23, "Droite":24}
ultrason={"envoi":5, "echo":6}

global dc, acceleration,  moteurPWM, moteurDIRPWM
dc=50
acceleration=15

def init():
    global dc, acceleration,  moteurPWM, moteurDIRPWM
    GPIO.setup(moteur1["PWM"], GPIO.OUT)
    GPIO.setup(moteur1["Avancer"], GPIO.OUT)
    GPIO.setup(moteur1["Reculer"], GPIO.OUT)
    GPIO.setup(moteur2["PWM"], GPIO.OUT)
    GPIO.setup(moteur2["Gauche"], GPIO.OUT)
    GPIO.setup(moteur2["Droite"], GPIO.OUT)
    GPIO.setup(ultrason["envoi"],GPIO.OUT)
    GPIO.setup(ultrason["echo"],GPIO.IN)

    moteurPWM = GPIO.PWM(moteur1["PWM"], 50)
    moteurPWM.start(0)
    moteurPWM.ChangeDutyCycle(dc)
    moteurDIRPWM = GPIO.PWM(moteur2["PWM"], 50)
    moteurDIRPWM.start(50)

def distance():
    GPIO.output(ultrason["envoi"], GPIO.HIGH)
    time.sleep(0.00001)
    GPIO.output(ultrason["envoi"], GPIO.LOW)
    start = time.time()
    while GPIO.input(ultrason["echo"])==0:
        pass

    debutImpulsion= time.time()

    while GPIO.input(ultrason["echo"])==1:
        pass

    finImpulsion  = time.time()

    distance = round((finImpulsion - debutImpulsion) * 343*100/2,1)
    return round(distance)

async def gestionobstacle () :
    while True:
        if distance()<= 50 :
            print("ALERTE OBSTACLE !")
            GPIO.output(moteur1["Avancer"],GPIO.LOW)
            GPIO.output(moteur1["Reculer"],GPIO.LOW)
        await asyncio.sleep(0.2)

async def gestionmoteur(message):
    global dc, acceleration,  moteurPWM, moteurDIRPWM

    #avancer
    if message=='Zactive':
        print("avance")
        #augmenter vitesse de 10
        acceleration=10
        GPIO.output(moteur1["Avancer"],GPIO.HIGH)
        GPIO.output(moteur1["Reculer"],GPIO.LOW)
    if message=='Zdeactive':
        #baisser la vitesse de 10
        acceleration=-10
        GPIO.output(moteur1["Avancer"],GPIO.LOW)
        GPIO.output(moteur1["Reculer"],GPIO.LOW)
    #reculer
    if message=='Sactive':
        #augmenter vitesse de 10
        acceleration=10
        GPIO.output(moteur1["Avancer"],GPIO.LOW)
```

```
        GPIO.output(moteur1["Reculer"],GPIO.HIGH)
    if message=='Sdeactive':
        #baisser la vitesse de 10
        acceleration=-10
        GPIO.output(moteur1["Avancer"],GPIO.LOW)
        GPIO.output(moteur1["Reculer"],GPIO.LOW)
    #arrêt
    if message=='Eactive':
        GPIO.output(moteur1["Avancer"],GPIO.LOW)
        GPIO.output(moteur1["Reculer"],GPIO.LOW)
    #accelerer
    if message=='Ractive':
        if dc<=95:
           dc=dc+5
           moteurPWM.ChangeDutyCycle(dc)
    #ralentir
    if message=='Factive':
        if dc>=30:
           dc=dc-5
           moteurPWM.ChangeDutyCycle(dc)
```

```
async def gestiondirection(message):
    print("message : ",message)

    #gauche
    if message=='Qactive':
        GPIO.output(moteur2["Gauche"],GPIO.HIGH)
        time.sleep (0.2)
        GPIO.output(moteur2["Gauche"],GPIO.LOW)

    #droite
    if message=='Dactive':
        GPIO.output(moteur2["Droite"],GPIO.HIGH)
        time.sleep (0.2)
        GPIO.output(moteur2["Droite"],GPIO.LOW)

def fin():
    moteurPWM.stop()
    moteurDIRPWM.stop()
    GPIO.cleanup()
```

Le fichier index.html

```
<!DOCTYPE html>
<html>
 <head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="style.css">
         <title>
         Projet Mariokart
         </title>
 </head>
 <body>
  <h1>Projet Mariokart</h1>
  <h2 id="titre">Interface</h2>
  <div id="camera">
   <fieldset>
   <legend>Vidéo en direct :</legend>
    <section class="streaming">
      <img name="PiCamera" src="http://10.3.141.1:8081/video"
      width="320" height="240" alt="Caméra Voiture"
      style="background-color: #009999" />
    </section>
   </fieldset>
  </div>
  <div id="codage">
    <input type="button" name="flecheAvancer" id="flecheAvancer"
value="Z" />
         <label for="flecheAvancer">Avancer</label><br />
    <input type="button" name="flecheReculer" id="flecheReculer"
value="S" />
         <label for="flecheReculer">Reculer<label><br />
    <input type="button" name="flecheDroite" id="flecheDroite"
value="D" />
         <label for="flecheDroite">Droite</label><br />
    <input type="button" name="flecheGauche" id="flecheGauche"
value="Q" />
         <label for="flecheGauche">Gauche</label><br />
    <input type="button" name="down" id="down" value="A" />
         <label for="down">Down</label><br />
    <input type="button" name="frein" id="frein" value="E" />
         <label for="frein">Frein</label><br />
  </div>
```

```
  <script>
   <script>
   // Gestion de la souris
   //  * ajout des gestionnaire d'evenement sur les boutons activés
par la souris
   //  * au clic gauche de la souris
   //  * et au relachement du bouton gauche de la souris
   var flecheAvancer = document.getElementById('flecheAvancer');
   flecheAvancer.addEventListener('mousedown',clicSouris);
   flecheAvancer.addEventListener('mouseup',relacheSouris);
   var flecheReculer = document.getElementById('flecheReculer');
   flecheReculer.addEventListener('mousedown',clicSouris);
   flecheReculer.addEventListener('mouseup',relacheSouris);
   var flecheDroite = document.getElementById('flecheDroite');
   flecheDroite.addEventListener('mousedown',clicSouris);
   flecheDroite.addEventListener('mouseup',relacheSouris);
   var flecheGauche = document.getElementById('flecheGauche');
   flecheGauche.addEventListener('mousedown',clicSouris);
   flecheGauche.addEventListener('mouseup',relacheSouris);
   var frein = document.getElementById('frein');
   frein.addEventListener('mousedown',clicSouris);
   frein.addEventListener('mouseup',relacheSouris);
   var accelerer = document.getElementById('accelerer');
   accelerer.addEventListener('mousedown',clicSouris);
   accelerer.addEventListener('mouseup',relacheSouris);
   var ralentir = document.getElementById('ralentir');
   ralentir.addEventListener('mousedown',clicSouris);
   ralentir.addEventListener('mouseup',relacheSouris);

   // gestion du clic gauche de la souris
   // envoi d'un message contenant : le nom de la touche et le mot
active
   function clicSouris(event) {
         ws.send(event.target.value +'active');
   }

   // gestion du relachement du bouton gauche de la souris et qui
   // sert pour avancer/reculer ; droite/gauche
   // envoi d'un message contenant : le nom de la touche et le mot
deactive
   function relacheSouris(event) {
```

```
            ws.send(event.target.value +'deactive');
    }

 //ajout des gestionnaotes d'evenemebt pour le clavier
  document.addEventListener('keydown', toucheAppuyee);
  document.addEventListener('keyup', toucheRelachee );

  // Gestion du clavier
  // Gestion de l'appui d'une touche
  // envoi d'un message contenant : le nom de la touche et le mot
active
  function toucheAppuyee(event) {
    // touche Z appuyée code 90 -> avancer
    if(event.keyCode == '90') {
       ws.send('Z' + 'active');
    }
    // touche S appuyée code 83 -> reculer
    if(event.keyCode == '83') {
       ws.send('S' + 'active');
    }
    // touche Q appuyée code 81 -> tourner a gauche
    if(event.keyCode == '81') {
       ws.send('Q' + 'active');
    }
    // touche D appuyée code 68 -> tourner a droite
    if(event.keyCode == '68') {
      ws.send('D' + 'active');
    }
    // touche A appuyée code 65 -> arret
    if(event.keyCode == '65') {
       ws.send('A' + 'active');
    }
    // touche E appuyée code 69 -> freiner
    if(event.keyCode == '69') {
       ws.send('E' + 'active');
    }
    // touche R appuyée code 82 -> accelerer
    if(event.keyCode == '82') {
       ws.send('R' + 'active');
    }
    // touche F appuyée code 70 -> ralentir
    if(event.keyCode == '70') {
       ws.send('F' + 'active');
    }
  }

  // Gestion du relachement d'une touche uniquement pour
  //  * avancer et reculer
  //  * droite et gauche pour ramener les roues au centre
  // et envoi d'un message contenant : le nom de la touche et le
mot deactive
  function toucheRelachee(event) {
    // touche Z relachée code 90 -> desactiver avancer
    if(event.keyCode == '90') {
       ws.send('Z' + 'deactive');
    }
    // touche S relachée code 83 -> desactiver reculer
    if(event.keyCode == '83') {
       ws.send('S' + 'deactive');
    }
    // touche Q relachée code 81 -> tourner a gauche
    if(event.keyCode == '81') {
        ws.send('Q' + 'deactive');
    }
    // touche D relachée code 68 -> tourner a droite
    if(event.keyCode == '68') {
       ws.send('D' + 'deactive');
    }
  }

  // Création du Websocket client vers le serveur du Raspberry Pi
  var ws = new WebSocket("ws://10.3.141.1:5678/");
  // envoi d'un premier message lors de la connexion au serveur
   ws.onopen = function (event) {
      ws.send("J'envoie un premier message au serveur.");
  };
 </script>
```

Le fichier style.css

```
#camera{
float:left;
}
```

```
# Definir la fonction qui sera appelee par le serveur a la connexion d'un client
lancement_serveur = websockets.serve(echange,'10.3.141.1', 5678)
# Creation de la boucle d'evenement (event loop)
loop = asyncio.get_event_loop()
loop.run_until_complete(lancement_serveur)
loop.run_forever()
loop.close()
```