

Programmation de la voiture

Le dossier du projet : [dossiermariokartv3.pdf](#)

Présentation

La programmation du déplacement de la voiture est gérée sur le principe d'une **architecture client-serveur** :

- la partie **serveur** va **recevoir** les requêtes de commandes et activer les moteurs. Cette partie est constituée de **deux programmes Python** sur le Raspberry :
- une partie **cliente** depuis laquelle on envoie des requêtes au serveur. Les requêtes sont par exemple faire avancer la voiture, freiner, reculer. Cette partie est gérée avec **une page html associée à un feuille de style CSS**.

Principe de la communication

- le **client** est un **ordinateur portable** qui se connecte en **wifi** sur le Raspberry. Le Raspberry est configuré comme **point d'accès Wifi** avec le logiciel **RaspAP**.
- sur l'ordinateur on lance un **navigateur Internet** et on se connecte au site Web de pilotage de la voiture à l'URL <http://10.3.141.1> Sur le Raspberry, un serveur Web est exécuté et pour publier les pages du projet avec comme page principale la page index.html.
- cette page **index.html** contient l'interface de pilotage de la voiture et crée le **websocket client** pour se connecter au **websocket serveur** du Raspberry à l'adresse `ws://10.3.141.1:5678`
- sur le Raspberry, le programme **serveur.py** doit être lancé pour créer le **websocket serveur** et **attendre les requêtes clientes**. En **permanence** le **serveur** envoie la **distance** du premier obstacle situé devant la voiture ainsi que sa **vitesse**.

Partie serveur

Le fichier Python serveur.py

Ce fichier gère le **websocket** et lance deux tâches :

- une tâche (**gestionreceptionmessage(websocket)**) pour **écouter** les messages reçus. Cette tâche lance elle-même deux autres tâches en **parallèle** :
 - une pour gérer le **déplacement (gestionmoteur(message))**
avancer/reculer/accélérer/ralentir/frein/arrêt
 - une pour gérer la **direction (gestiondirection(message))**
- une tâche (**gestionenvoimessage(websocket)**) pour **envoyer** des messages vers le client (la page web) afin **d'afficher** la distance et pourquoi pas la vitesse.

Le fichier python fonctions.py

Ce fichier contient les instructions pour :

- **initialiser** le **GPIO** du Raspberry utilisé par la carte SB,
- les **fonctions** pour **traiter** les messages reçus.

Partie cliente

Les fichiers sont dans le dossier **/var/www/isn** du serveur Web

Le fichier **index.html**

- définit l'**interface en HTML et CSS**,
- gère l'**interaction** avec l'utilisateur et le **websocket** en **javascript**.
 - **création** du websocket client et **connexion** au serveur à l'adresse `ws://10.3.141.1:5678`
 - **affichage** en permanence de la distance et de la vitesse
 - **affichage** en permanence du **flux vidéo** qui est diffusé par le **logiciel motion** configuré sur le Raspberry à l'adresse <http://10.3.141.1:8081/video>
 - **interaction** avec l'utilisateur :
 - en utilisant soit les **touches du clavier**, soit **la souris**,
 - à **chaque action** de l'utilisateur, un **message est envoyé** au serveur pour indiquer ce qu'il faut faire (avancer, reculer, etc.)

Le fichier de style **style.css** permet de gérer l'affichage côte à côte de la vidéo et des commandes en laissant le **bloc div de la vidéo flotter** à gauche du bloc des commandes.

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 ; dc : {}".format(distance(),dc))
        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, acelerer, 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))
    #gere l'obstacle
    obstacle = asyncio.ensure_future(gestionobstacle())
    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

```

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}
ultrasonarriere={"envoi":4, "echo":18}

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)
    GPIO.setup(ultrasonarriere["envoi"], GPIO.OUT)
    GPIO.setup(ultrasonarriere["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)

def distancearriere():
    GPIO.output(ultrasonarriere["envoi"], GPIO.HIGH)
    time.sleep(0.00001)
    GPIO.output(ultrasonarriere["envoi"], GPIO.LOW)
    start = time.time()
    while GPIO.input(ultrasonarriere["echo"])==0:
        pass
    debutImpulsion= time.time()
    while GPIO.input(ultrasonarriere["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':
        #augmenter la vitesse de 15
        acceleration=15
        GPIO.output(moteur1["Avancer"],GPIO.HIGH)
        GPIO.output(moteur1["Reculer"],GPIO.LOW)
    if message=='Zdeactive':
        #baisser la vitesse de 30
        acceleration=-30
        GPIO.output(moteur1["Avancer"],GPIO.LOW)
        GPIO.output(moteur1["Reculer"],GPIO.LOW)
    #reculer
    if message=='Sactive':
        #augmenter la vitesse de 10
        acceleration=10
        GPIO.output(moteur1["Avancer"],GPIO.LOW)
        GPIO.output(moteur1["Reculer"],GPIO.HIGH)
    if message=='Sdeactive':
        #baisser la vitesse de 30
        acceleration=-30
        GPIO.output(moteur1["Avancer"],GPIO.LOW)
        GPIO.output(moteur1["Reculer"],GPIO.LOW)
    #arret
    if message=='Eactive':
        GPIO.output(moteur1["Avancer"],GPIO.LOW)
        GPIO.output(moteur1["Reculer"],GPIO.LOW)
    #accellerer
    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 (0.1)
        GPIO.output(moteur2["Gauche"],GPIO.LOW)
#droite
if message=='Dactive':
    GPIO.output(moteur2["Droite"],GPIO.HIGH)
    time.sleep (0.1)
    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>
    <div id="camera">
      <fieldset>
        <legend>Vidéo en direct :</legend>
        <section class="streaming">
          
        </section>
      </fieldset>
    </div>
    <fieldset>
      <legend>Commande :</legend>
      <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="frein" id="frein" value="E" />
<label for="frein">Frein</label><br />
<input type="button" name="frein" id="accellerer" value="R" />
<label for="accellerer">Accélérer</label><br />
<input type="button" name="ralentir" id="ralentir" value="F" />
<label for="ralentir">Ralentir</label><br />
<span id="messagerecu">Message</span><br />
</div>
</fieldset>
<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 accellerer = document.getElementById('accellerer');
  accellerer.addEventListener('mousedown', clicSouris);
  accellerer.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
  fonction clicSouris(event) {
    //alert(event.target.value +'active');
    ws.send(event.target.value +'active');
  }

  // gestion du relachement du bouton gauche de la souris : sert
avancer/reculer ; droite/gauche
  // envoi d'un message contenant : le nom de la touche et le mot deactive
  fonction relacheSouris(event) {
    //alert(event.target.value +'deactive');
    ws.send(event.target.value +'deactive');
  }

```

```
}

//ajout des gestionnaires d'evenement 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 E appuyée code 69 -> freiner
    if(event.keyCode == '69') {
        ws.send('E' + 'active');
    }
    // touche R appuyée code 82 -> acclerer
    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.");
};
var messagerecu = document.getElementById('messagerecu');
ws.onmessage = function (event) {
    //affiche le message recu dans la balise <span>
    messagerecu.innerHTML = event.data;
};
</script>
</body>
</html>

```

Le code Javascript isolé pour avoir la coloration syntaxique

```

<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) {
    //alert(event.target.value + 'active');
    ws.send(event.target.value + 'active');
}

// gestion du relachement du bouton gauche de la souris : sert
// avancer/reculer ; droite/gauche
// envoi d'un message contenant : le nom de la touche et le mot deactive
function relacheSouris(event) {
    //alert(event.target.value + 'deactive');
    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 E appuyée code 69 -> freiner
    if(event.keyCode == '69') {
        ws.send('E' + 'active');
    }
}
```

```
// touche R appuyée code 82 -> accélérer
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.");
};
var messagerecu = document.getElementById('messagerecu');
ws.onmessage = function (event) {
    //affiche le message reçu dans la balise <span>
    messagerecu.innerHTML = event.data;
};
</script>
```

Le fichier style.css

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

From:

/ - **APs et stages du BTS SIO du lycée Suzanne Valadon**

Permanent link:

</doku.php/isn/2017/projetmariokart/programmation>

Last update: **2018/06/10 22:17**

