

Jacob Turcotte, Nathanaël Amaridon et Eric Gingras

Technologie des systèmes ordinés

Groupe 2317

Comment utiliser l’application python

Compte rendu présenté à

M. Pierre Bergeron

Pour le cours

Gestion de l’information 247-517-SH A2022

Remis

21 décembre 2022

[Partie 1 : Introduction à l’interface graphique 3](#_Toc122555227)

[Introduction 3](#_Toc122555228)

[Colonne 1 4](#_Toc122555229)

[Colonne 2 5](#_Toc122555230)

[Colonne 3 5](#_Toc122555231)

[Partie 2 : Explication du fonctionnement du code 6](#_Toc122555232)

[Introduction aux sections du code. 6](#_Toc122555233)

[Section 1 6](#_Toc122555234)

[Section 2 6](#_Toc122555235)

[Section 3 7](#_Toc122555236)

[Section 4 8](#_Toc122555237)

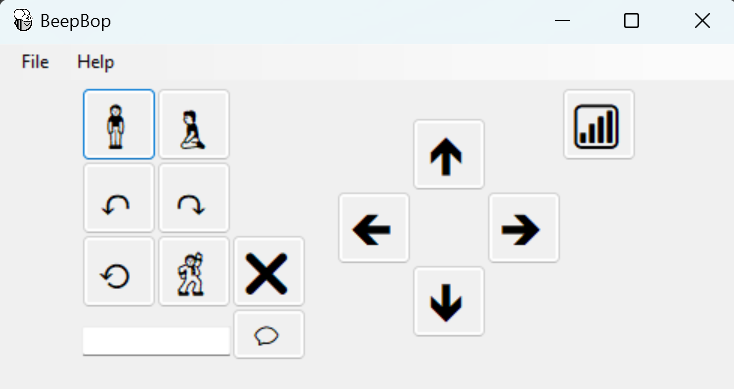
[Partie 4 : Utiliser l’application 10](#_Toc122555238)

# Partie 1 : Introduction à l’interface graphique

## Introduction

L’interface graphique contient ses éléments:

1. Trois colonnes
   1. La première colonne contient:
      1. 8 boutons (Button)
      2. 1 boîte d’entrée texte (Text Input)
   2. La deuxième colonne contient:
      1. 4 boutons (Button)
   3. La troisième colonne contient :
      1. 1 bouton (Button)



a.ii.1

c.i

a.i.1

b.i

Les boutons sont en rouge et la boîte d’entrée de texte est en vert.

## Colonne 1

Dans cette section nous allons voir les différentes fonctions de la première colonne.

A picture containing electronics

Description automatically generated

Les boutons permettent d’envoyer une commande au robot NAO. Ils le font tous bouger (à l’exception du bouton d’envoi de texte en bas à droite).

Le bouton, avec l’image du monsieur debout, envoi une commande au robot lui demandant de se mettre debout.

Le bouton, avec l’image du monsieur assis, envoi une commande au robot lui demandant de se mettre assis.

Le bouton, avec l’image de la flèche qui tourne vers la gauche, envoi une commande au robot lui demandant de tourner vers la gauche d’environs 30 degrés.

Le bouton, avec l’image de la flèche qui tourne vers la droite, envoi une commande au robot lui demandant de tourner vers la droite d’environs 30 degrés.

Le bouton, avec l’image de la flèche qui fait un cercle, envoi une commande au robot lui demandant de tourner 180 degrés.

Le bouton, avec l’image d’un bonhomme qui danse, envoi une commande au robot lui demandant de faire une animation mystère.

Le bouton, avec le X, envoi une commande au robot lui demandant d’arrêter l’animation mystère.

La boîte d’entrée texte permet à l’utilisateur d’écrire un message pour le robot NAO. Le bouton envoi crée une commande au robot lui demandant de dire le message écrit en français. Il est aussi utilisé pour connecter au robot à l’aide de commandes spécifiques que nous allons voir dans la partie 3.

## Colonne 2

Dans cette section nous allons voir les différentes fonctions de la deuxième colonne.

Une image contenant jack

Description générée automatiquement

Les boutons permettent d’envoyer une commande au robot NAO. Ils le font tous marcher dans une direction choisie.

Les boutons envoient tous une commande demandant au robot de bouger selon la direction de la flèche du bouton appuyé.

## Colonne 3

Dans cette section nous allons voir les différentes fonctions de la troisième colonne.

Icon

Description automatically generated

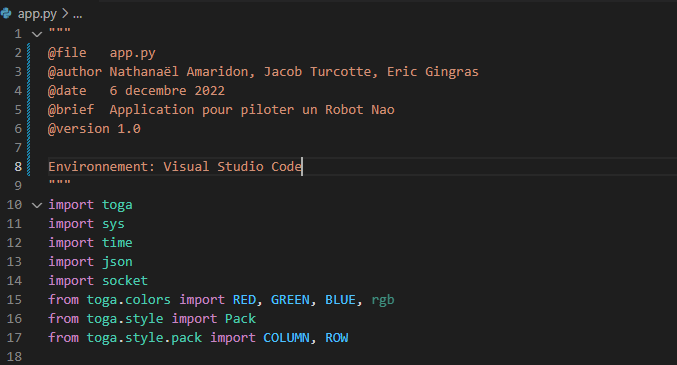
Ce bouton permet d’établir une connexion avec l’ESP8266 pour pouvoir communiquer avec le robot. Notez qu’il faut envoyer des commandes spécifiques par la boîte de texte avant d’utiliser ce bouton pour établir la connexion. Cela sera expliqué dans la partie 3.

# Partie 2 : Explication du fonctionnement du code

## Introduction aux sections du code.

### Section 1

La première section du code contient tous les imports pour inclure les différentes libraires qui rendent l’application fonctionnel.



### Section 2

La deuxième section contient tout ce qui a rapport avec l’affichage de l’application.

Premièrement, il y a la création des boîtes qui serviront à contenir tous les éléments de l’affichage. Btn\_box, joystick\_box et connexion\_box correspondent aux 3 colonnes de boutons de l’affichage que nous avons vu dans la partie 1.

Text

Description automatically generated

Ensuite, il y a la création des éléments de l’affichage comme les boutons et le textbox. Si vous regardez de proche, vous pouvez repérer les différentes icônes des boutons sur l’affichage. Il est aussi possible de voir à quels fonctions les boutons sont reliées.

Graphical user interface, text, application

Description automatically generated

Finalement, pour afficher les boutons et le textbox à l’écran, il faut les ajouter aux différentes boîtes. Cette étape assure que les bons boutons se trouvent dans les bonnes colonnes.

Text

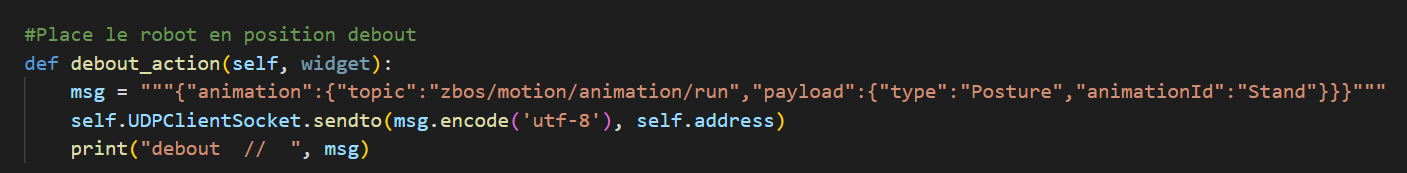
Description automatically generated

### Section 3

La troisième section du code est dédiée aux différentes fonctions pour faire bouger le robot.

Voici la fonction qui est utilisée pour mettre le robot en position debout quand le bouton A picture containing text, screenshot

Description automatically generated est appuyé.



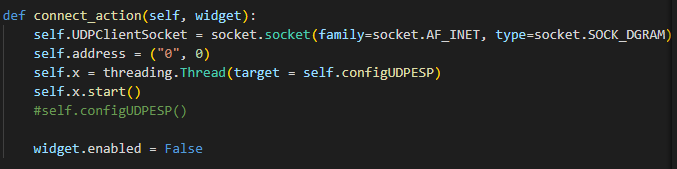
Premièrement, on crée le string « msg » qui contient la commande pour mettre le robot en position debout.

Ensuite, le string est envoyé au module ESP8266 par UDP. L’ESP8266 décortique le string et envoie la bonne commande au robot pour lui mettre en position debout.

### Section 4

La quatrième section du code est dédiée à la connexion UDP.

Voici la fonction qui fait la connexion UDP au ESP.



Cette fonction s’active lorsqu’on appuie sur le boutonIcon

Description automatically generated. On crée le socket UDP et on lui assigne les paramètres nécessaires. On crée le tuple self.address et un crée un thread avec la fonction configUDPESP(). On fait cela pour faire en sorte que notre application soit plus fluide, plusieurs lignes de codes dans la fonction configUDPESP() peuvent faire « bugger » l’application pendant quelques secondes.

Une image contenant texte

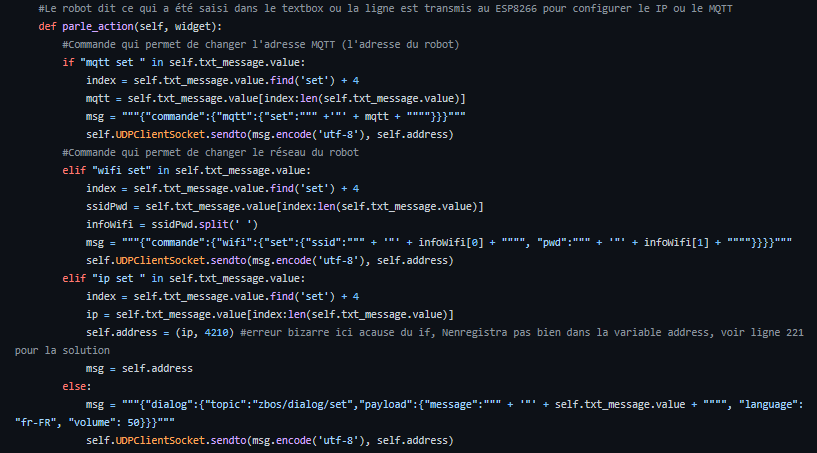
Description générée automatiquement

Cette fonction permet d’attacher (bind) l’adresse de l’appareil qui roule l’application ainsi que le port utiliser au socket UDP. On affiche les informations avant et après afin de confirmer que l’attachement c’est bien fait. Finalement, un envoi un message de confirmation au ESP. Si le message est bien envoyé, ça veut dire que le socket UDP est fonctionnel et nous avons la bonne adresse du ESP.

Une image contenant texte

Description générée automatiquement

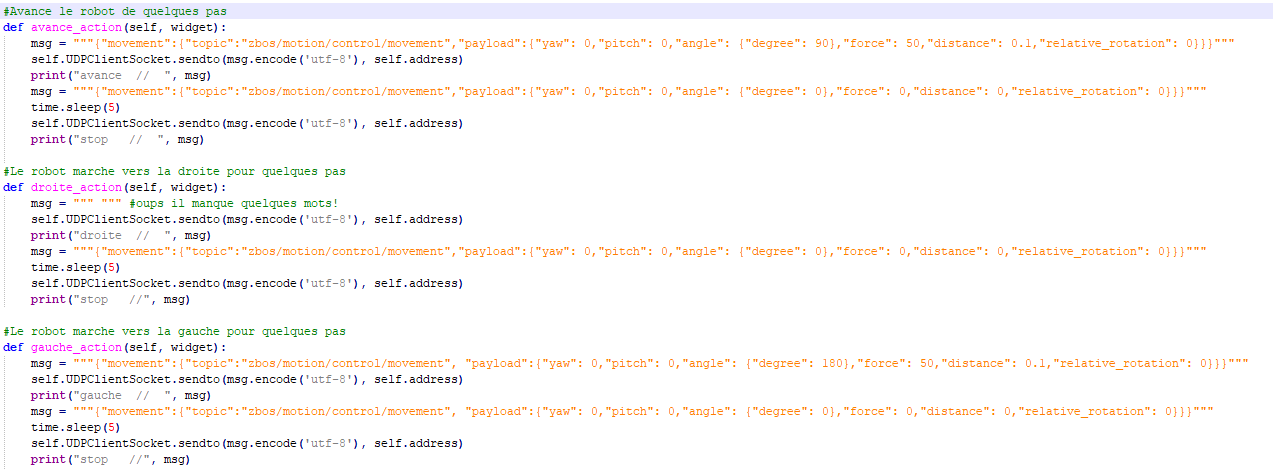
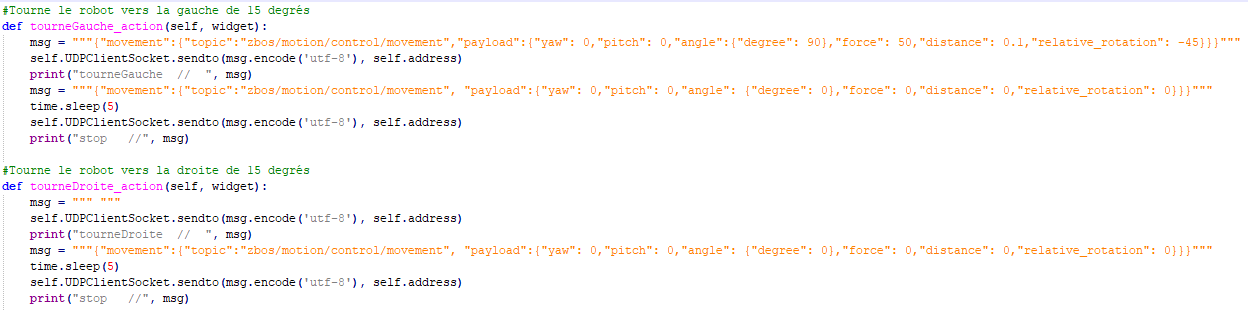
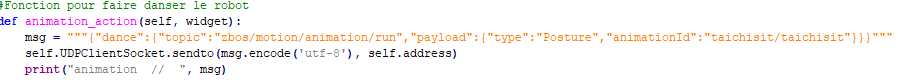
Cette fonction retourne l’IP de l’appareil qui utilise l’application Beepbop. Elle marche sur Android aussi mais pas dans une VM.



Cette fonction permet de faire 4 choses.

1. On peut envoyer l’adresse IP du robot NAO au ESP afin qu’il puisse y envoyer les commandes mqtt. Utiliser la commande « mqtt set ‘IP’»
2. On peut envoyer le SSID du réseau ainsi que du mot de passe au ESP. Ainsi, l’ESP peut être utilisé n’importe où sans à avoir besoin de changer le code. Utiliser la commande « wifi set ‘SSID’ ‘mot de passe’»
3. On peut manuellement mettre l’adresse IP du ESP dans l’application. Utiliser la commande « ip set ‘IP’ »
4. N’importe qu’elle autre commande va faire parler le robot.

# Partie 4 : Utiliser l’application

1. Commencez l’exécutable Beepbop.exe
2. Tapez la commande « ip set » suivi de l’adresse de l’ESP fournie par l’animateur dans la boîte de texte dans la première colonne. Appuyez sur le bouton send.
   1. Exemple : ip set 10.240.9.25
3. Vous pouvez tester la connexion en tapant un texte aléatoire dans la boîte texte. Si la connexion est bonne le robot NAO va dire le texte lorsque vous allez appuyer sur le bouton send.
   1. Exemple : Allo le monde
4. Maintenant c’est à votre tour de jouer! Testez les différents boutons, certains d’entre eux ne marche pas!
5. Les deux boutons pour allez vers la droite ont du texte manquant. Basez vous sur le code des fonctions voisines pour écrire le texte manquant!
6. Modifier la variable msg de la fonction animation\_action afin que le robot puisse performer une autre danse de votre choix. Utilisez ce site pour vous aider [Motion :: Documentation Portal (zorabots.be)](https://docs.zorabots.be/dev-mqtt-docs/latest/motion.html) .