Thème : informatique embarquée et objets connectés

Classe enseignée : SNT

Park Assist

L.DIDIER

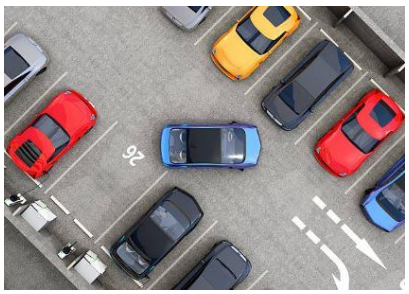
Compétences visées

Un des objectifs de l'enseignement de SNT est de développer et de coder des scripts PYTHON afin d'apporter une réponse à une problématique précise. A travers le thème " informatique embarqué et objets connectés" , nous pouvons notamment travailler les compétences suivantes dans l'activité proposée :

- Coder des scripts simples d'acquisition de données.
- Gérer des entrées/sorties à travers les ports utilisés par le système.
- Écrire et développer des algorithmes pour résoudre une problématique.
- Identifier des algorithmes de contrôle des comportements physiques à travers les données des capteurs.

Le développement des logiciels embarqués est délicat, car il pose souvent des questions de temps-réel, c'est-à-dire de respect de temps de réponse imposé. Ceci conduit à des méthodes de programmation spécifiques dans certains cas.

Situation déclenchante



Aujourd'hui, de nombreuses voitures sont équipées de système de Park Assist. Comment fonctionne ce système et comment la technologie permet de nous assister lors de situations compliquées ?

Problématique

Comment concevoir une solution technologique
pour détecter une place et simuler un Park Assist ?

TI-83 Premium CE
EDITION PYTHON

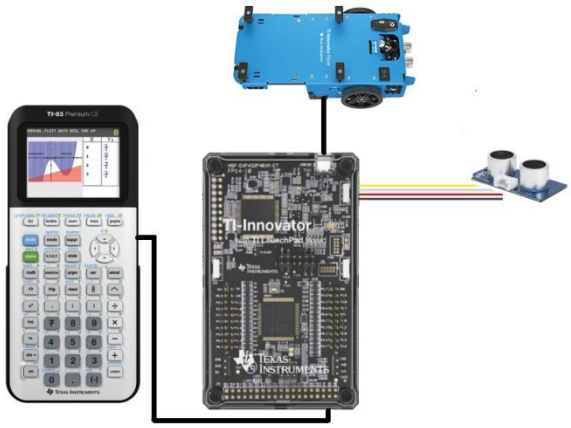
Thème : informatique embarquée et objets connectés

Classe enseignée : SNT

Fiche méthode

L.DIDIER

Matériel nécessaire



- Calculatrice TI-83 Premium CE
- Câble (calculatrice/Hub)
- TI-Innovator Hub
- Rover
- Capteur de distance (ranger)
- Câbles

Déroulement possible du projet

Travail de groupe possible. Chaque groupe disposera du matériel ci-dessus et devra concevoir une réponse à la problématique sous forme d'une démonstration munie d'une documentation qui explique les scripts PYTHON pilotant le Rover. A la fin du projet, les groupes pourront voter pour élire la production qui répond le mieux à la problématique.

Les scripts développés devront répondre à la problématique uniquement dans la situation ci-contre et répondre aux critères suivants :

Critère 1 : Le rover devra circuler sur la route, à moins de 20 cm latéralement des voitures garées.

Critère 2 : Détecter une place suffisante pour garer le rover.

Critère 3 : Mener une manœuvre pour garer le rover .

Critère 4 : Allumer la diode du rover lorsque la manœuvre est engagée.

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

Thème : informatique embarquée et objets connectés

TI-83 Premium CE
EDITION PYTHON

Classe enseignée : SNT

python™

Fiche méthode

L.DIDIER

Proposition de résolution

Il faudra fixer le ranger sur le côté du rover de manière à pouvoir détecter une place latérale lors du déplacement du rover (voir vidéo).

Importation des bibliothèques (voir le paragraphe importation des bibliothèques)

L'instruction **rv.forward(50)** permet d'avancer de 50 unités de mesure (le décimètre, par défaut).

Tant que le rover ne détecte pas de place latérale il continue à avancer.

L'instruction **a=ranger("IN1")** permet d'associer le ranger à la variable a.

L'instruction **b=a.measurement( )** permet de lire la distance latérale et permet de la stocker dans b. Cette distance est en mètres.

Détection d'une place et calcul de la longueur de la place.

L'instruction **d=rv.waypoint_distance( )** permet de

connaitre la distance parcourue depuis l'instruction **rv.path_clear( )**.

Vérification si la distance est suffisante pour garer le rover (30 cm).

Manœuvre pour garer le rover et allumer la diode RGB du rover.

L'instruction **rv.color_rgb(0,255,0)** permet d'allumer la diode RGB du rover avec la couleur verte. L'intensité est ici de 255 (et peut varier entre 0 et 255).

L'instruction **rv.left(50)** permet au rover de tourner à gauche de 50 degrés.

L'instruction **rv.backward(2.8)** permet au rover de reculer de 2.8 unités de mesure (le décimètre).

ÉDITEUR : PLACE
LIGNE DU SCRIPT 0001

```
TI-Rover
from time import *
from ti_system import *
import ti_rover as rv
from ranger import *

b=0
a=ranger("IN 1")
p=0
while p==0:
  rv.forward(50)
  while b<=0.2:
    b=a.measurement()
    print(b)
  rv.stop()
  rv.path_clear()
  rv.forward(50)
  while b>0.2:
    b=a.measurement()
    print(b)
  d=rv.waypoint_distance()
  print("d=",d)
  rv.stop()
  if d>0.3:
    p=1
rv.color_rgb(0,255,0)
wait(1)
rv.backward(0.5)
wait(1)
rv.left(50)
wait(1)
rv.backward(2.8)
wait(1)
rv.right(50)
wait(1)
rv.forward(0.2)
wait(2)
rv.color_rgb(0,0,0)
```

Fns… | a A # | Outils | Exéc | Script

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus !

TI-83 Premium CE
EDITION PYTHON

Thème : informatique embarquée et objets connectés

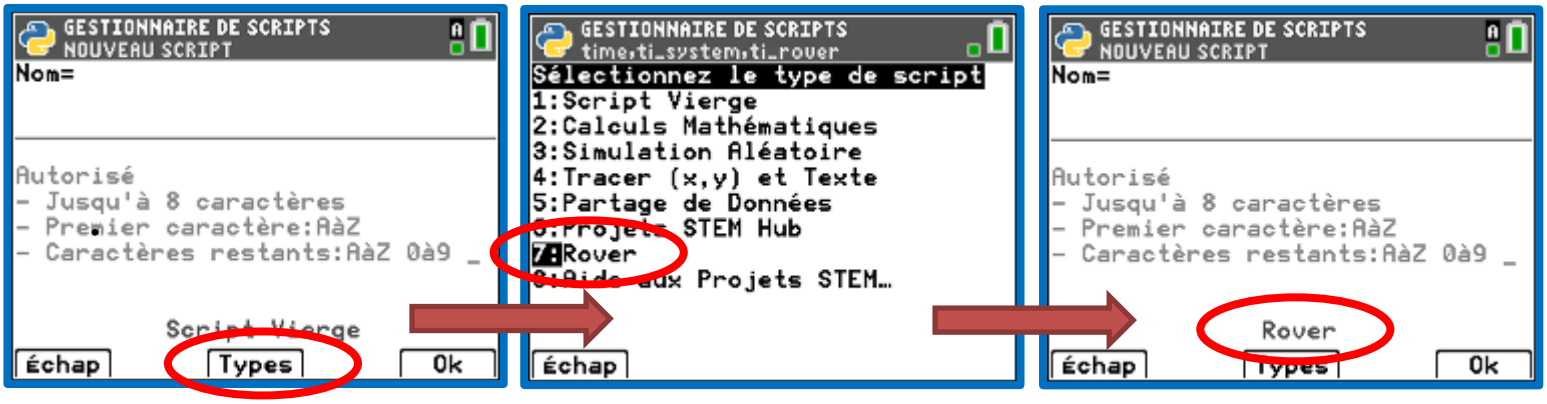
Classe enseignée : SNT

Fiche méthode

L.DIDIER

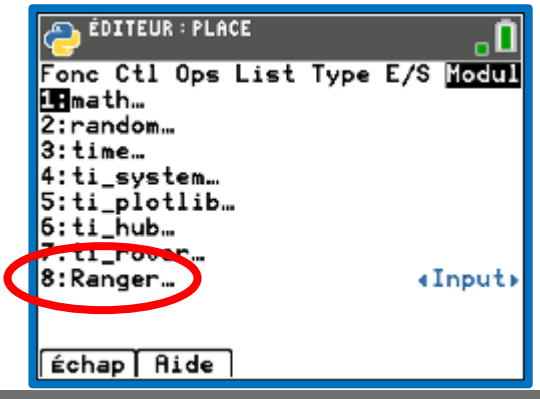
Importation des bibliothèques

- Lors de la création du script, sélectionner la rubrique Types, puis Rover. Cela permettra de charger automatiquement les trois bibliothèques suivantes : *time* , *ti_systeme* et *ti_rover*.



- Une fois le nouveau script créé, pour importer la bibliothèque *Ranger* appuyer sur la touche f(x), sélectionner la rubrique Modul, puis suivre les copies d'écran suivantes :

- Pour utiliser la bibliothèque *Ranger* importée, appuyer sur la touche f(x), sélectionner la rubrique Modul, puis sélectionner **8 : Ranger…**
- Au fur et à mesure que les bibliothèques sont importées, elles s'ajoutent aux bibliothèques déjà présentes (*math*, *random*, …etc.) et permettent d'accéder aux instructions qu'elles contiennent.



Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus !