STEMKIT 4SCHOOLS

FEUX DE CIRCULATION AVEC PYTHON ET GPIO

Plan de leçon 2

Cofinancé par le programme Erasmus+ de l'Union européenne

Ce projet a été financé avec le soutien de la Commission européenne.
Cette communication ne reflète que le point de vue de l'auteur et la Commission ne peut être tenue responsable de l'usage qui pourrait être fait des informations qui y sont contenues.

Table des matières

1. Feux de signalisation Python et GPIO

1.1 Informations générales

1.1.1 Description succincte

Dans ce plan de leçon, nous allons apprendre à construire des feux de signalisation dans le monde réel et à les contrôler via le GPIO et le langage de programmation Python. Le plan de cours implique la création du circuit à l'aide des broches GPIO de notre Raspberry Pi et des composants électroniques, et le développement d'un programme en Python qui contrôlera la séquence des feux de signalisation.

1.1.2 Objectifs d'apprentissage

Les principaux objectifs d'apprentissage de ce plan de cours sont:

- Compréhension du concept et du contenu de la théorie des files d'attente, de l'ingénierie du trafic et de la théorie des goulots d'étranglement.
- Se familiariser avec les composants électroniques et créer des circuits et des programmes pour interagir avec les broches GPIO de votre Raspberry Pi.
- Comprendre les structures de base du langage de programmation Python (syntaxe, fonctions, boucles, instructions if, etc.).

1.1.3 Liens avec le curriculum

Les domaines, sous-domaines, sujets / sujets auxquels ce plan de cours peut être lié sont:

- Maths: théorie des files d'attente, files d'attente, files d'attente.
- Ingénierie: goulots d'étranglement, systèmes de service, files d'attente.
- Informatique / Informatique: unité de traitement et périphériques, interfaces, langage de programmation et structures principales, codage.
- Technologie: électronique, matériel et logiciel open source, capteurs, signal numérique, circuits, ordinateurs monocarte.

1.1.4 Matériel requis

Pour ce plan de cours (et pour chaque groupe d'étudiants) en plus de la console STEMKIT et de son Raspberry Pi, nous aurons besoin de:

- 6 x fils de raccordement mâle-femelle
- 1 x fil de raccordement mâle à mâle
- 1 x planche à pain
- 1 x bouton poussoir
- 1 x buzzer

- 3 x LED (rouge, orange et verte)
- 3 x résistances 220 Ohm

1.1.5 Durée

La durée de ce plan de cours est estimée à environ 45 à 60 minutes, soit une heure de classe.

1.2 Plan de leçon

Le plan de leçon est divisé en trois phases, qui sont l'introduction, la préparation et la conclusion.

1.2.1 Introduction

La théorie des files d'attente est l'étude mathématique des files d'attente ou des files d'attente. Une file d'attente est un conteneur d'objets qui sont insérés et supprimés selon le principe du premier entré, premier sorti (FIFO). Nouveaux ajouts à la ligne effectués à l'arrière de la file d'attente, tandis que la suppression se produit à l'avant.

La mise en file d'attente est également l'étude du comportement du trafic à proximité d'une certaine section où la demande dépasse la capacité. Les files d'attente peuvent être observées dans de nombreuses situations: monter à bord d'un bus / avion, goulots d'étranglement d'autoroute, faire les courses, sortir de la salle de classe à la fin d'une conférence ou attendre un hamburger au hamburger local.

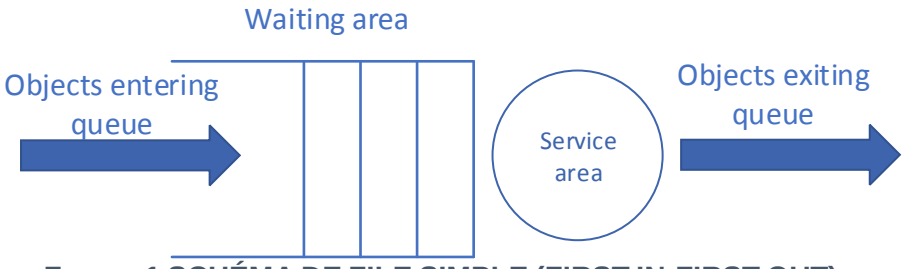

FIGURE 1 SCHÉMA DE FILE SIMPLE (FIRST IN-FIRST OUT)

Il existe différentes distributions statistiques qui expliquent une file d'attente. La distribution des arrivées peut être déterministe (aléatoire) ou aléatoire (par exemple, Poisson). La distribution des services est soit déterministe, soit aléatoire. De plus, il existe différents modèles de services:

- Premier entré, premier sorti (FIFO)
- Dernier entré, premier sorti (LIFO)
- Priorité (contourne la file d'attente)

Dans l'ingénierie du trafic, la mise en file d'attente peut se produire aux feux rouges, aux panneaux d'arrêt, aux goulots d'étranglement ou à toute restriction de flux basée sur la conception ou sur le trafic. Lorsqu'elles ne sont pas traitées correctement, les files d'attente peuvent provoquer de graves congestions du réseau, ce qui en fait quelque chose d'important à étudier et à comprendre. Le modèle de service le plus courant qui explique une file d'attente à un feu de signalisation est FIFO. Cependant, dans certaines circonstances, le modèle de service change (par exemple, lorsque le trafic est contrôlé par un agent de la circulation).

Dans la théorie des files d'attente, nous utilisons la notation suivante:

- Taux d'arrivée = λ
- Taux de départ = μ
- Taux d'utilisation $\rho = \lambda / \mu$

Comprenant le degré de saturation d'une file d'attente, nous considérons ce qui suit:

- Sursaturé: λ> μ
- Sous-saturé: λ <μ
- Saturés: λ = μ

Une file d'attente de trafic est expliquée à l'aide de la formule de Little:
Formule de Little: E (n) = λE (v)
La formule montre que la taille moyenne de la file d'attente (mesurée en véhicules) est égale au taux d'arrivée (véhicules par unité de temps) multiplié par le temps d'attente moyen (à la fois le temps de retard dans la file d'attente et le temps de service) (en unités de temps). Ce résultat est indépendant des distributions d'arrivée particulières et est un principe fondamental important qui n'a été prouvé qu'en 1961.
Dans notre cas, nous utiliserons la théorie des files d'attente pour identifier la signalisation optimale des feux de signalisation (en unités de temps - secondes) afin que le taux d'arrivée λ soit égal ou presque égal au taux de départ μ.
Maintenant que nous avons acquis la compréhension de base des files d'attente, passons à la préparation de notre expérience.

1.2.2 Préparation

1.2.2.1 Création du circuit

La première chose à faire est de créer un circuit et de le connecter aux broches GPIO de notre Raspberry Pi. Éteignez le Raspberry et débranchez tous les câbles qui y sont attachés. Pour notre circuit, nous avons besoin d'une maquette, de LED (rouge, orange, verte), de résistances, de fils de liaison, d'un buzzer et d'un bouton poussoir.
Nous devons placer tous les composants sur la maquette et les connecter aux broches GPIO appropriées sur le Raspberry Pi. Pour comprendre comment chaque composant est connecté, nous devons garder à l'esprit les éléments suivants:

- Un bouton poussoir nécessite 1 broche de masse et 1 broche GPIO
- Une LED nécessite 1 broche de masse et 1 broche GPIO, avec une résistance de limitation de courant.
- Un buzzer nécessitait 1 broche de terre et 1 broche GPIO. Les composants peuvent partager une broche de terre. Nous utilisons la maquette pour activer cela.

Le circuit complet est dans le diagramme schématique qui suit.

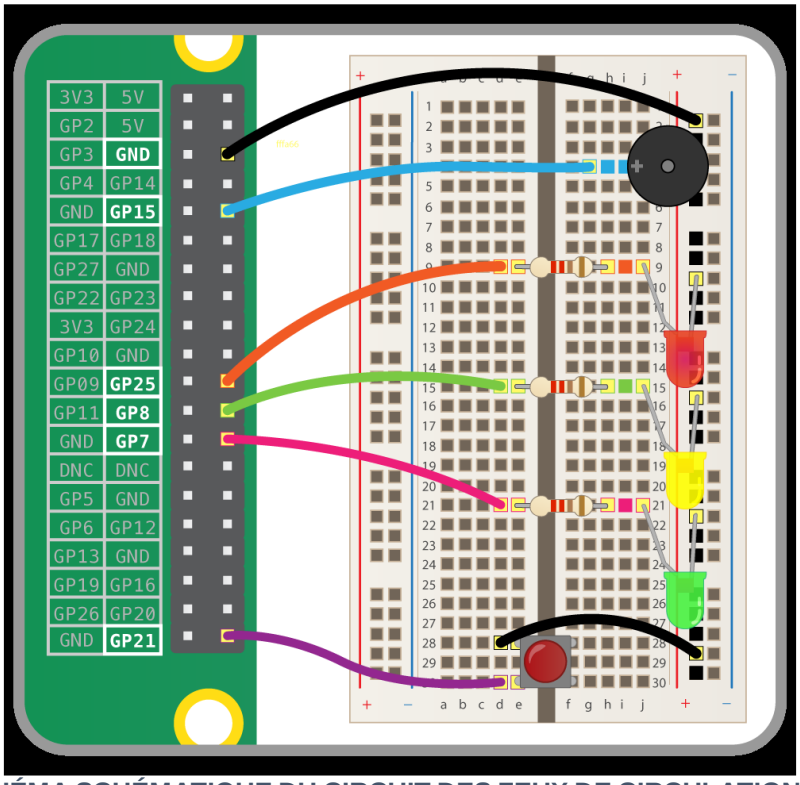

FIGURE 1 SCHÉMA SCHÉMATIQUE DU CIRCUIT DES FEUX DE CIRCULATION CONNECTÉ AUX PINS GPIO. SOURCE: PROJECTS.RASPBERRYPI.ORG

Notez que la rangée le long du côté long de la planche à pain est connectée à une broche de terre sur le Raspberry Pi, de sorte que tous les composants de cette rangée sont connectés à la terre.
En ce qui concerne les broches GPIO, nous utilisons les éléments suivants:

- Bouton à 21
- LED rouge à 25
- LED orange à 8
- LED verte à 7
- Buzzer à 15

Lorsque nous avons terminé le circuit, nous pouvons allumer le Raspberry Pi et commencer à développer notre programme en Python.

1.2.2.2 Programmation Python pour le circuit

Nous devons créer un programme qui contrôle le circuit que nous avons créé à l'étape précédente.
En bref, dans notre programme, nous importons d'abord les modules Python dont nous avons besoin, puis définissons les broches GPIO pour les LED, les boutons et le buzzer. Nous utilisons le module gpiozero car il s'appuie sur RPi.GPIO et a des règles de codage beaucoup plus simples.

```
#########################
# importer des bibliothèques et définir le mode de numérotation gpio
à partir du bouton d'importation gpiozero, TrafficLights, Buzzer
depuis le temps importer le sommeil

# Interface de bouton pour contrôler le bouton poussoir
bouton = Bouton (21)
Interface #TrafficLights pour simuler un feu tricolore réel
feux = TrafficLights (25, 8, 7)
# Interface Buzzer pour contrôler le buzzer
buzzer = Buzzer (15)

tandis que Vrai:
# Séquence pour feu vert
    lights.green.on ()
    button.wait_for_press () #like dans un passage pour piétons
    sleep (3) #delay pour exécuter le code

#Séquence pour la lumière orange
    lights.green.off ()
    lights.amber.on ()
    dormir (3)

#Séquence pour la lumière rouge
    lights.red.on ()
    buzzer.beep ()
    lights.amber.off ()
    dormir (15)

    lights.amber.on ()
    buzzer.off ()
    dormir (1)

    lights.red.off ()
    lights.amber.off ()
```

Dans GPIO Zero, nous créons des objets pour chaque composant utilisé. Chaque interface de composant doit être importée du module gpiozero et une instance créée sur le numéro de broche GPIO auquel elle est connectée.
L'interface TrafficLights prend trois numéros GPIO, un pour chaque broche: rouge, orange, vert (dans cet ordre). Nous pouvons utiliser on, off et clignoter, qui contrôlent tous les trois lumières à la fois.
L'interface Buzzer prend un numéro GPIO et peut être contrôlée par: on, off et bip. En utilisant le nombre entre parenthèses, nous pouvons manipuler le temps de bourdonnement.
Le programme ci-dessus crée la séquence de feux de signalisation suivante:
• Vert allumé

• Ambre allumé
• Rouge allumé
• Rouge et ambre allumés
• Vert allumé

Le bouton simule un passage pour piétons. Lorsqu'un «piéton» appuie sur le bouton, les feux passent au rouge (avec un certain retard) et donnent aux piétons le temps de traverser la rue avant de ramener les feux au vert. L'ajout d'un buzzer améliore la simulation, car dans un vrai passage à niveau il y a un buzzer au profit des piétons malvoyants.

Vérifiez que votre programme fonctionne. Appuyez plusieurs fois sur le bouton et voyez vos feux de signalisation fonctionner. Essayez de changer l'heure du feu vert et du feu rouge.

1.2.3 Conclusion

Nous avons réussi à créer un système de feux de signalisation et un passage pour piétons en utilisant la programmation Python et le GPIO de notre Raspberry. Dans cette phase, nous récapitulons ce que nous avons fait et comment, quelles étaient les principales étapes et discutons des difficultés rencontrées. En tant qu'exercice de suivi, discutez des moyens d'optimiser les scripts utilisés pour cette expérience.

1.3 Références ou Ressources

Voici quelques références utiles et ressources supplémentaires liées à ce plan de cours.

- https://projects.raspberrypi.org/en/projects/physical-computing/10
- https://gpiozero.readthedocs.io/en/stable/
- https://www.cs.cmu.edu/~adamchik/15-121/lectures/Stacks%20and%20Queues/Stacks%20and%20Queues.html
- https://en.wikipedia.org/wiki/Queueing_theory
- https://en.wikibooks.org/wiki/Fundamentals_of_Transportation/Queueing#cite_ref-1
- https://en.wikipedia.org/wiki/Traffic_engineering_(transportation)