Hailo AI Kit avec Raspberry Pi 5

Dogan Ibrahim (Royaume-Uni)

Pour ceux qui connaissent déjà le Raspberry Pi 5 mais hésitent à explorer l'*Edge Computing* et le matériel associé, le livre d'Elektor, *Getting Started with the Raspberry Pi 5 AI Kit*, constitue une excellente introduction. Ce guide est riche en exemples pratiques pour débuter efficacement la programmation IA avec le Raspberry Pi 5, s'appuyant sur le kit Raspberry Pi AI d'Elektor. Pour illustrer le potentiel de ce livre et du kit, voici un extrait qui explique comment la reconnaissance de personnes peut être effectuée de deux manières différentes à l'aide d'outils d'IA gratuits, ainsi qu'une introduction concise aux composants logiciels Hailo

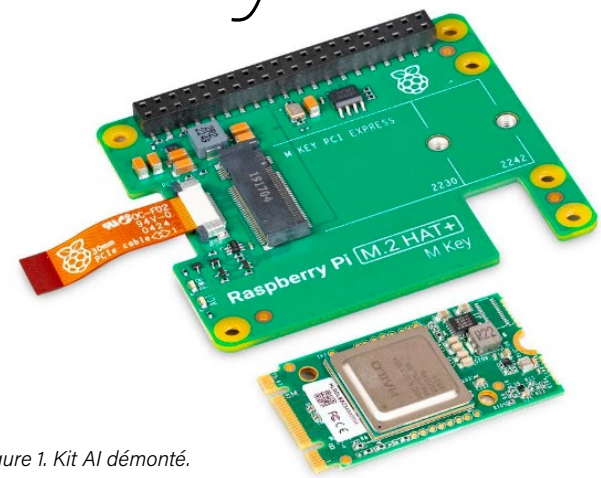

Figure 1. Kit AI démonté.

Note de la rédaction : cet article est un extrait du livre « Getting Started with the Raspberry Pi AI Kit », qui sera publié par Elektor en 2024. Il a été formaté et légèrement modifié pour correspondre aux normes éditoriales et à la mise en page du magazine Elektor. L'auteur et l'éditeur seront heureux de répondre aux questions – pour les contacter, voir l'encadré « **questions ou commentaires ?** »..

Le kit Raspberry Pi 5 AI (**figure 1**) est un accélérateur pour l'inférence de réseaux neuronaux avec une capacité de 13 TOPS (trillions d'opérations par seconde) et est conçu autour de la puce Hailo-8L. Ce kit inclut le Raspberry Pi M.2 HAT+ et un module d'accélération Hailo AI destiné à être utilisé avec le Raspberry Pi 5. Ce module adopte le format M.2 2242 et est préinstallé dans le M.2 HAT+, auquel il se connecte via un connecteur M-key edge. Le M.2 HAT+ facilite la communication entre l'interface M.2 du module AI et l'interface PCIe 2.0 du Raspberry Pi 5. En conséquence, ce kit AI offre une solution abordable, efficace et économe en énergie, adaptée à une multitude d'applications IA, y compris la reconnaissance d'objets, l'automatisation industrielle, la gestion des processus, la sécurité, la robotique, le secteur médical et bien plus encore.

Caractéristiques

- Contient un accélérateur d'inférence de réseau neuronal capable de 13 TOPS
- Entièrement intégré à la pile logicielle de l'appareil photo Raspberry Pi
- Le pad thermique préinstallé entre le module et le HAT+ répartit la chaleur entre les composants, améliorant ainsi les performances.
- Conforme aux spécifications du Raspberry Pi HAT+.

Inclus

Chaque AI Kit est livré avec un module AI préinstallé, un câble ruban, un connecteur GPIO et du matériel de montage. En détail, sont inclus dans le kit :

- Accélérateur Hailo 8L avec unité de traitement neuronal (NPU).
- Raspberry Pi M.2 HAT+
- Pad thermique préinstallé entre le module et le M.2 HAT+.
- Kit de matériel d'assemblage (entretoises, vis)
- Embase d'empilage GPIO de 16 mm
- 4 entretoises filetées
- 8 vis
- 1 vis pour fixer et soutenir le périphérique M.2.

Configuration du matériel

La **figure 2** illustre l'AI Kit monté sur un Raspberry Pi 5. Pour utiliser l'AI Kit, il doit d'abord être installé au-dessus du Raspberry Pi 5. Il est conseillé d'utiliser ce kit avec le refroidisseur actif Raspberry Pi. Si vous disposez d'un tel refroidisseur, assurez-vous de l'installer avant de monter l'AI Kit. De plus, il est préférable d'installer la caméra du Raspberry Pi 5 (par exemple, la caméra V3) avant le kit AI, ce qui facilitera le branchement du câble de la caméra.

Installation du logiciel du AI Kit

Pour commencer, vous devez vous assurer que la dernière version du système d'exploitation Bookworm est installée sur votre Raspberry Pi 5.
Exécutez les commandes suivantes pour mettre à jour le système d'exploitation vers la dernière version

```
pi@raspberrypi:~ $ sudo apt update
pi@raspberrypi:~ $ sudo apt full-upgrade
```

Assurez-vous que le micrologiciel du Raspberry Pi 5 est à jour :

```
pi@raspberrypi:~ $ sudo rpi-eeprom-update
```

Si vous voyez "décembre 2023" ou une date ultérieure, passez à l'étape suivante. Si vous voyez une date antérieure au 6 décembre 2023, exécutez la commande suivante pour ouvrir l'outil de configuration Raspberry Pi :

```
pi@raspberrypi:~ $ sudo raspi-config
```

Sous *Advanced Options Bootloader Version*, chisissez *Latest*. Quittez ensuite `raspi-config` avec *Finish* ou la touche *Echap*. Exécutez la commande suivante :

```
pi@raspberrypi:~ $ sudo rpi-eeprom-update -a
```

Pour obtenir des performances optimales du dispositif Hailo, il est nécessaire de régler *PCIe* sur *Gen3*. Bien qu'il soit possible d'utiliser *Gen2*, les performances seront moindres. Les étapes sont les suivantes :

```
pi@raspberrypi:~ $ sudo raspi-config
```

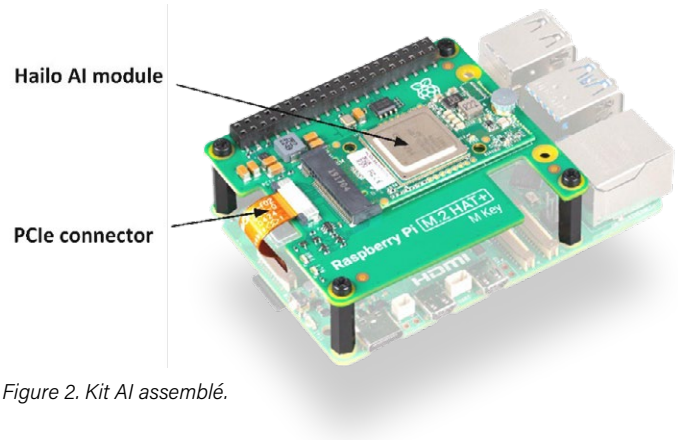

Figure 2. Kit AI assemblé.

Sélectionnez l'option *6. Advanced Options*, puis sélectionnez l'option *A8 PCIe Speed*. Choisissez *Yes* pour activer le mode *PCIe Gen 3*. Cliquez sur *Finish* pour quitter.
Redémarrez votre Raspberry Pi 5 :

```
pi@raspberrypi:~ $  sudo reboot
```

Installation du logiciel du Hailo AI Kit

Exécutez la commande suivante pour installer le logiciel Hailo AI Kit sur Raspberry Pi 5 :

```
pi@raspberrypi:~ $ sudo apt install hailo-all
```

La commande ci-dessus installe les dépendances suivantes :

- Pilote de périphérique et micrologiciel du noyau Hailo
- Logiciel d'exécution HailoRT
- Paquet de base Hailo Tappas
- Les étapes de démonstration du logiciel de post-traitement Hailo rpicam-apps

Redémarrez votre Raspberry Pi 5 pour que les paramètres ci-dessus prennent effet :

```
pi@raspberrypi:~ $  sudo reboot
```

Vérification de l'installation du logiciel

Vous devez vérifier l'installation du logiciel pour vous assurer que tous les composants du logiciel sont en place et que l'AI kit peut être reconnu par le Raspberry Pi 5. Les étapes sont :

- Assurez-vous que le matériel AI kit est installé sur votre Raspberry Pi 5 comme décrit précédemment.
- Entrez la commande suivante :

```
pi@raspberrypi:~ $ hailortcli fw-control identify
```

Si tout est correct, vous devriez voir une sortie similaire à la **figure 3**. Notez ici que la version du logiciel est 4.17.

- Vous devriez également tester l'*installation de TAPPAS Core* en exécutant la commande suivante :

```
pi@raspberrypi:~ $ gst-inspect-1.0 hailotools
```

```
pi@raspberrypi:~ $ hailortcli fw-control identify
Executing on device: 0000:01:00.0
Identifying board
Control Protocol Version: 2
Firmware Version: 4.17.0 (release,app,extended context switch buffer)
Logger Version: 0
Board Name: Hailo-8
Device Architecture: HAILO8L
Serial Number: HLDDLBB241901947
Part Number: HM21LB1C2LAE
Product Name: HAILO-8L AI ACC M.2 B+M KEY MODULE EXT TMP

pi@raspberrypi:~ $
```

Figure 3. Vérification de l'installation du logiciel.

```
pi@raspberrypi:~ $ gst-inspect-1.0 hailotools
Plugin Details:
  Name                     hailotools
  Description              hailo tools plugin
  Filename                 /lib/aarch64-linux-gnu/gstreamer-1.0/libgsthailoto
.so
  Version                  3.28.2
  License                  unknown
  Source module            gst-hailo-tools
  Binary package           gst-hailo-tools
  Origin URL               https://hailo.ai/

  hailoaggregator: hailoaggregator - Cascading
  hailocounter: hailocounter - postprocessing element
  hailocropper: hailocropper
  hailoexportfile: hailoexportfile - export element
  hailoexportzmq: hailoexportzmq - export element
  hailofilter: hailofilter - postprocessing element
  hailogallery: Hailo gallery element
  hailograytonv12: hailograytonv12 - postprocessing element
  hailoimportzmq: hailoimportzmq - import element
  hailomuxer: Muxer pipeline merging
```

Figure 4. Test TAPPAS Core installation (une partie de l'affichage est montrée).

```
pi@raspberrypi:~ $ dmesg | grep -i hailo
[    4.245963] hailo: Init module. driver version 4.18.0
[    4.246076] hailo 0000:01:00.0: Probing on: 1e60:2864...
[    4.246081] hailo 0000:01:00.0: Probing: Allocate memory for device
, 11632
[    4.246099] hailo 0000:01:00.0: enabling device (0000 -> 0002)
[    4.246105] hailo 0000:01:00.0: Probing: Device enabled
[    4.246123] hailo 0000:01:00.0: Probing: mapped bar 0 - 00000000352
4
[    4.246129] hailo 0000:01:00.0: Probing: mapped bar 2 - 00000000b64
[    4.246134] hailo 0000:01:00.0: Probing: mapped bar 4 - 00000000159
```

Figure 5. Vérification des journaux du noyau (une partie de l'affichage est montrée).

Si tout va bien, vous devriez voir quelque chose de similaire à la **figure 4**.
Si `hailo` ou `hailotools` sont introuvables, essayez de supprimer le registre GStreamer :

```
pi@raspberrypi:~ $  rm~/.cache/gstreamer-1.0/registry.aarch64.bin
```

- En outre, vous pouvez exécuter la commande suivante pour vérifier les journaux du noyau :

```
pi@raspberrypi:~ $  dmesg | grep -i hailo
```

Ce qui devrait donner une sortie similaire à **figure 5**.

- Enfin, vous devez vérifier que la caméra fonctionne correctement. Entrez la commande suivante, qui affichera un aperçu de la caméra pendant 10 secondes.

```
pi@raspberrypi:~ $  rpicam-hello-t 10s
```

Si vous utilisez un module V3 camera avec votre Raspberry Pi 5, ajoutez les lignes suivantes à votre fichier *config.txt*:

- Editer le *fichier config.txt*:
  `sudo nano /boot/firmware/config.txt`
- Remplacer la ligne : `camera_auto_detect=1`
  par `camera_auto_detect=0`
- Insérer la ligne : `dtoverlay=imx708`
- Tapez *CntrlX* suivi de *Y pour* sauvegarder et quitter.

Saisissez à nouveau la commande `rpicam-hello -t 10s` pour vérifier à nouveau le fonctionnement de la caméra. À ce stade, vous avez vérifié que le matériel et le logiciel AI Kit ont été correctement installés et que votre caméra fonctionne.

Exécution des programmes de démonstration

Un certain nombre de programmes de démonstration sont disponibles avec l'AI Kit pour le Raspberry Pi 5. Les étapes pour installer certains de ces programmes de démonstration sont :

- Cloner le dépôt

```
git clone https://github.com/hailo-ai/hailo-rpi5-examples.git
```

- Entrer dans le répertoire du dépôt :

```
cd hailo-rpi5-examples
```

- Pour exécuter les exemples, vous devez vous assurer que votre environnement est correctement configuré. Vous pouvez le faire en utilisant le script suivant. Ce script définira les variables d'environnement nécessaires et activera l'environnement virtuel Hailo (s'il n'existe pas, il le créera).

```
source setup_env.sh
```

- Assurez-vous d'être dans l'environnement virtuel et exécutez la commande suivante. Cela peut prendre un certain temps, attendez qu'elle soit terminée.

```
pip install -r requirements.txt
```

- Téléchargement des ressources. Cela peut prendre un certain temps, attendez qu'il soit terminé :

```
./download_resources.sh
```

Ce dépôt contient des exemples de pipelines de base pour la plateforme RPi5 de Hailo. Les exemples démontrent la détection d'objets, l'estimation de la pose humaine et la segmentation d'instances. Il est construit comme des modèles pour vous permettre d'utiliser ces applications comme base pour vos propres projets.
Le dossier *hailo-rpi5-examples* contient les dossiers et les fichiers présentés dans la **figure 6**.
Les programmes d'applications (détection, estimation de la pose, segmentation des instances) sont dans le dossier *basic_pipelines* comme le montre la **figure 7**.

Exemple de démonstration de détection

Cet exemple démontre la détection d'objets. Il utilise le modèle *YOLOv6n*par défaut. Il supporte également les modèles *yolov8s* et *yolox_s_leaky*. Il utilise la couche NMS (Non-Maximum Suppression) de Hailo en tant que partie du fichier HEF, de sorte que tous les réseaux de détection qui sont compilés

```
pi@raspberrypi:~/hailo-rpi5-examples $ ls
basic_pipelines          download_resources.sh  requirements.txt
build.release            hailort.log            resources
compile_postprocess.sh   LICENSE                setup_env.sh
cpp                      meson.build            venv_hailo_rpi5_examples
doc                      README.md
pi@raspberrypi:~/hailo-rpi5-examples $
```

Figure 6. Contenu du dossier des exemples.

```
pi@raspberrypi:~/hailo-rpi5-examples/basic_pipelines $ ls
detection.py          __init__.py                pose_estimation.py
hailo_rpi_common.py   instance_segmentation.py   __pycache__
pi@raspberrypi:~/hailo-rpi5-examples/basic_pipelines $
```

Figure 7. Programmes d'applications

avec NMS peuvent être utilisés avec le même code.
Pour exécuter l'exemple, utilisez (assurez-vous d'être en mode Desktop GUI) :

```
cd hailo-rpi5-examples
source setup_env.sh
python basic_pipelines/detection.py --input resources/
detection0.mp4
```

Cette application lit le fichier vidéo *detection0.mp4* et détecte et indique divers éléments dans le fichier. La **figure 8** montre la sortie du programme affichée sur le moniteur du Raspberry Pi, où les objets identifiés sont représentés par des rectangles.
Pour fermer l'application, appuyez sur *Ctrl+C*. Pour obtenir des options de saisie supplémentaires, entrez la commande suivante :

```
python basic_pipelines/detection.py --help
```

Par exemple, pour lire les images d'une caméra Raspberry Pi 5, entrez `--input rpi`. La commande suivante lira les images de la caméra Raspberry Pi 5 :

```
python basic_pipelines/detection.py --input rpi
```

De même, pour lire les images d'une caméra USB (webcam), entrez `--input /dev/video0`.
Notez que la caméra USB n'est pas garantie d'être `/dev/video0`. Vous pouvez vérifier quels périphériques vidéo sont disponibles en entrant la commande :

```
ls /dev/video*
```

ou en entrant la commande suivante : ls /dev/video*

```
v4l2-ctl -list-devices
```

Vous pouvez tester si la caméra USB fonctionne en entrant (Notez que c'est L minuscule, pas 1) :

```
ffplay -f v4l2 /dev/video0
```

Si vous obtenez une erreur, essayez un autre périphérique (par exemple `/dev/video2`). Dans la configuration de l'auteur, la caméra USB était nommée `/dev/video8`. Votre caméra USB aura probablement un nom de périphérique différent.
Par exemple, pour lire les images de la caméra USB, l'auteur a utilisé l'option `-i /dev/video8`. Dans l'exemple suivant, un clavier est détecté à l'aide d'une caméra USB :

```
python basic_pipelines/detection.py -i /dev/video8
```

La sortie est présentée dans la **figure 9** avec l'étiquette du clavier affichée dans le coin supérieur gauche de l'écran.
Pour afficher la fréquence d'images, ajoutez le flag `--print-fps`. Cela affichera la fréquence d'images sur le terminal et dans la fenêtre de sortie vidéo.

Projet - Détection de présence humaine et signalisation par LED

Description : Ce projet consiste à détecter la présence d'une personne et à allumer une LED pour signaler cette détection. Le dispositif maintient la LED allumée jusqu'à ce que l'utilisateur appuie sur un bouton de réinitialisation confirmant la détection. Le voyant s'éteint ensuite, et le cycle de détection se réinitialise. Cet exemple simple mais pratique peut être adapté pour utiliser un relais ou un buzzer à la place de la LED. Le programme peut également être modifié pour reconnaître des objets autres que des personnes.

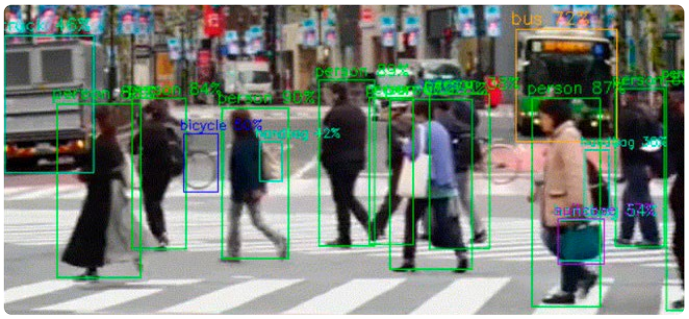

Figure 8. Sortie du programme.

Figure 9. Détection d'un clavier.

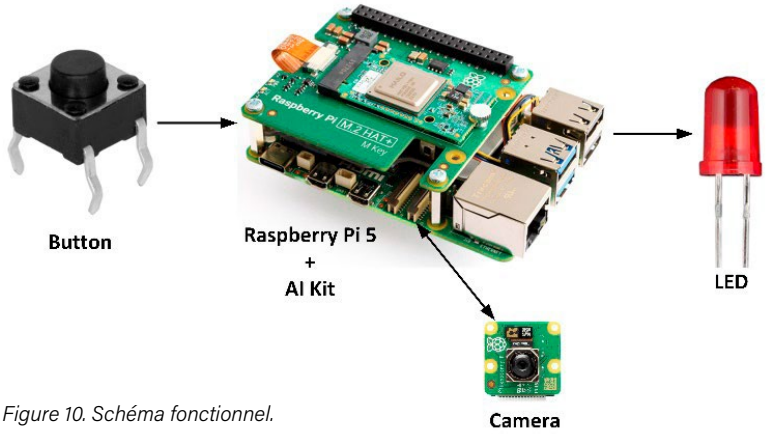

Figure 10. Schéma fonctionnel.

Schéma fonctionnel: La **figure 10** présente le schéma fonctionnel du projet.

Schéma du circuit : Le schéma du circuit, présenté à la **figure 11**, montre la LED connectée au pin 21 du Raspberry Pi 5 via une résistance de 330 Ω pour limiter le courant. Un bouton-poussoir est relié au pin 20. Le bouton est normalement au niveau haut (logic 1) et passe au niveau bas (logic 0) lorsqu'il est pressé.

Listage du programme : Le code Python est présenté dans le **listage 1**. Le programme crée un sous-programme et s'exécute comme un programme shell. Au début du programme, la bibliothèque *gpiozero* est importée et le port GPIO 21 du Raspberry Pi 5 est affecté à LED et le bouton est affecté au port 20.

Le programme d'inférence Hailo de post-traitement *rpicam-apps* est lancé dans une boucle `while` et la sortie de ce programme est comparée si elle contient le mot *person*. c'est-à-dire si une personne a été détectée. Si c'est le cas, le message *Personne détectée* s'affiche sur l'écran, le programme shell se termine et la LED s'allume. Le programme attend ensuite que le bouton soit pressé. L'appui sur le bouton est comme un accusé de réception de la part de l'utilisateur. L'appui sur le bouton éteint la LED. Le programme attend alors 15 secondes, puis le processus de détection reprend depuis le début.

Ce programme peut facilement être modifié en remplaçant la LED par un relais ou un buzzer, et en ajustant le délai d'attente selon les besoins spécifiques de l'application. Il est particulièrement adapté à des applications de sécurité, où il est crucial de détecter la présence humaine. Par ailleurs, il est possible de configurer la détection d'autres types d'objets et de programmer des actions en conséquence.

Formation à la détection d'objets

La détection et l'apprentissage d'objets via l'IA sont des processus complexes nécessitant une compréhension approfondie et une maîtrise pratique de la programmation, notamment en Python. Le chapitre 11 du livre décrit en détail comment mener à bien la formation à la détection d'objets sur un système Ubuntu, avec des résultats appliqués sur un Raspberry Pi 5.

240562-04

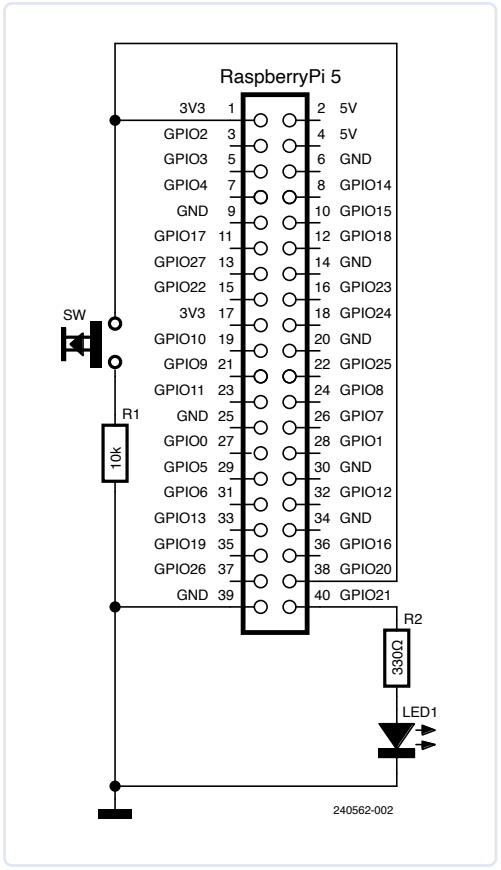

Figure 11. Schéma du circuit.

À propos de l'auteur

Dogan Ibrahim est titulaire d'une licence en ingénierie électronique, d'une Master en ingénierie du contrôle automatique et d'un doctorat en traitement des signaux numériques. Il a travaillé dans de nombreuses organisations industrielles avant de revenir à la vie universitaire. Le professeur Ibrahim est l'auteur de plus de 70 ouvrages techniques et a publié plus de 200 articles techniques sur les microcontrôleurs, les microprocesseurs et les domaines connexes. Il est ingénieur électricien agréé et membre de l'Institution of the Engineering Technology. Il est un professionnel certifié d'Arduino.

Listage 1. Code Python.

```
#-----------------------------------------------------------------
#          DETECT PERSON AND TURN ON LED
#          ---------------------------------------------
#
# File   :  Detect.py
# Date   : October, 2024
# Author: Dogan Ibrahim
#-----------------------------------------------------------------
import os
import subprocess
from gpiozero import LED, Button
from time import sleep
led = LED(21)                                  # LED port
button = Button(20)                            # Button port
led.off()                                      # LED OFF
while(True):
   p=subprocess.call("rpicam-hello -n -v 2 -t 0 \
     --post-process-file /home/pi/rpicam-apps/assets/hailo_yolov8_inference.json\
     --lores-width 640 --lores-height 640 2>&1 | grep -m 1 -c \
     --line-buffered 'person'",shell=True)
     print("Person detected...")
#
# At this point a person has been detected
#
   led.on()
   button.wait_for_press()                    # Wait for button press
   led.off()
   sleep(15)
```

Questions ou commentaires ?

Envoyez un courriel à l'auteur (d.ibrahim@btinternet.com), ou contactez Elektor (redaction@elektor.fr).

Produit

- **Dogan Ibrahim, *Getting Started with the Raspberry Pi AI Kit* (Elektor 2024)**
  www.elektor.fr/21031
- **Dogan Ibrahim, *Getting Started with the Raspberry Pi AI Kit* (Elektor 2024, E-book)**
  www.elektor.fr/21032
- **Raspberry Pi AI Kit**
  www.elektor.fr/20879