

 kunalyselne / Nhận dạng khuôn mặt sử dụng Raspberry Pi

Cộng cộng

A small project which does face detection using OpenCV library.

☆ 76 stars

 38 forks

 Branches

 Tags


 Activity

☆ Star

 Notifications

<> Mã số

 Vấn đề

 Yêu cầu kéo

 Hành động

 Dự án

 Bảo vệ

 Thông tin chi tiết



bậc thầy

 1 Chi nhánh

 0 Thẻ






 Đi đến tập tin

Go to file


Mã số













 kunalyselne

Xóa kết quả.jpg

c7db8c9 · 6 năm trước



 .gitignore	Cam kết ban đầu	7 năm trước
 01_face_dataset.py	Thêm tệp thông qua tải lên	7 năm trước
 02_face_training.py	Thêm tệp thông qua tải lên	7 năm trước
 03_nhận diện khuôn mặt.py	Thêm tệp thông qua tải lên	7 năm trước
 CamTest.py	Thêm tệp thông qua tải lên	7 năm trước
 FaceDetection.py	Thêm tệp thông qua tải lên	7 năm trước
 FaceEyeDetection.py	Thêm tệp thông qua tải lên	7 năm trước
 FaceSmileDetection.py	Thêm tệp thông qua tải lên	7 năm trước
 README.md	Cập nhật README.md	7 năm trước
 haarcascade_frontalface_default.xml	Thêm tệp thông qua tải lên	7 năm trước


Nhận dạng khuôn mặt sử dụng Raspberry Pi


Sự miêu tả

Một dự án nhỏ thực hiện phát hiện khuôn mặt bằng thư viện OpenCV trên RaspberryPi.

Nội dung

1. [Raspberry pi là gì?](#)
2. [Nhận dạng khuôn mặt?](#)
3. [Thư viện OpenCV?](#)
4. [Dự án](#)

 [Yêu cầu](#)

 [Thủ tục](#)

Raspberry pi là gì?

Raspberry pi là một máy tính mini (kích thước bằng thẻ tín dụng), có khả năng thực hiện nhiều tác vụ khác nhau nhưng với sức mạnh tính toán kém hơn vì không có tài nguyên mạnh. Raspberry Pi chậm hơn máy tính xách tay hoặc máy tính để bàn hiện đại nhưng vẫn là một máy tính Linux hoàn chỉnh và có thể cung cấp tất cả các khả năng mong đợi ngụ ý ở mức tiêu thụ điện năng thấp.

[Khám phá thêm](#)

Nhận diện khuôn mặt là gì?

Nhận dạng khuôn mặt/Nhận dạng khuôn mặt là một loại phần mềm sinh trắc học nhận dạng mọi người bằng khuôn mặt của họ. Khuôn mặt được chụp bằng máy ảnh kỹ thuật số và hệ thống được đào tạo và sau đó có khả năng nhận dạng người đó.

Thư viện OpenCV

OpenCV (Open Source Computer Vision Library) là một thư viện phần mềm máy tính và thị giác máy tính nguồn mở. OpenCV được xây dựng để cung cấp một cơ sở hạ tầng chung cho các ứng dụng thị giác máy tính và để đẩy nhanh việc sử dụng nhận thức máy trong các sản phẩm thương mại. Nó có giao diện C++, Python và Java và hỗ trợ Windows, Linux, Mac OS, iOS và Android.

[Đọc thêm](#)

Dự án

Trong dự án này chúng ta sẽ tìm hiểu cách phát hiện khuôn mặt theo thời gian thực trên Picam.

Những thứ được sử dụng trong dự án này

1. [Raspberry Pi 3 Model-B](#)
2. [Mô-đun máy ảnh Raspberry Pi](#)

Phần mềm

1. [Thư viện OpenCV](#)
2. [Python3](#)
3. [Trình soạn thảo văn bản](#)

1] Giới thiệu:

Dự án này được thực hiện với "Open Source Computer Vision Library", OpenCV. Trong dự án này, chúng tôi sẽ sử dụng Raspberry Pi (tức là Raspbian làm hệ điều hành) và Python. OpenCV được thiết kế để đạt hiệu quả tính toán và tập trung mạnh vào các ứng dụng thời gian thực. Vì vậy, nó hoàn hảo để nhận dạng khuôn mặt thời gian thực bằng camera.

2] Cấu trúc dự án:

1. Phát hiện khuôn mặt và thu thập dữ liệu.
2. Đào tạo hệ thống.
3. Nhận dạng khuôn mặt.

3] Thủ tục:

Bước 1: Cài đặt thư viện OpenCV

Tôi đang sử dụng Raspberry Pi V3 được cập nhật lên phiên bản Raspbian mới nhất (Stretch), vì vậy cách tốt nhất để cài đặt OpenCV là làm theo hướng dẫn tuyệt vời do Adrian Rosebrock phát triển: [Raspbian Stretch: Cài đặt OpenCV 3 + Python trên Raspberry Pi của bạn](#)

Sau khi hoàn tất hướng dẫn, bạn sẽ có một môi trường ảo OpenCV sẵn sàng để chạy thử nghiệm của chúng tôi trên Pi của bạn. Hãy vào môi trường ảo của chúng tôi và xác nhận rằng OpenCV 3 đã được cài đặt đúng cách.

Chạy lệnh `source` mỗi khi bạn mở một thiết bị đầu cuối mới để đảm bảo các biến hệ thống đã được thiết lập chính xác.

```
source ~/.profile
```

Tiếp theo, chúng ta hãy vào môi trường ảo của mình:

```
workon cv
```

Nếu bạn thấy văn bản (cv) đứng trước lời nhắc của mình thì bạn đang ở trong môi trường ảo cv:

```
(cv) pi@raspberrypi:~$
```

Bây giờ, hãy nhập trình thông dịch Python của bạn:

```
python
```

sau đó nhập thư viện OpenCV:

```
>>import cv2
```

Nếu không có thông báo lỗi nào xuất hiện thì OpenCV đã được cài đặt đúng cách TRÊN MÔI TRƯỜNG ẢO PYTHON CỦA BẠN.

Bạn cũng có thể kiểm tra phiên bản OpenCV đã cài đặt:

```
cv2.__version__
```

Bước 2: Kiểm tra Camera

Sau khi bạn đã cài đặt OpenCV trong Raspberry Pi, hãy kiểm tra để xác nhận camera của bạn hoạt động bình thường. Tôi cho rằng bạn đã cài đặt PiCam trên Raspberry Pi.

Bạn phải bật camera khi thực hiện hướng dẫn của Adrian, nếu không, trình điều khiển sẽ không được cài đặt đúng cách.

Trong trường hợp bạn gặp lỗi như: OpenCV Error: Assertion failed , bạn có thể thử giải quyết vấn đề bằng lệnh:

```
sudo modprobe bcm2835-v4l2
```

Sau khi đã cài đặt đúng tất cả trình điều khiển, hãy nhập mã Python bên dưới vào IDE hoặc bất kỳ Trình soạn thảo văn bản nào:

```
import numpy as np
import cv2
cap = cv2.VideoCapture(0)
cap.set(3,640) # set Width
cap.set(4,480) # set Height
while(True):
    ret, frame = cap.read()
    if ret:
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        cv2.imshow('frame', frame)
        cv2.imshow('gray', gray)

        k = cv2.waitKey(30) & 0xff
        if k == 27: # press 'ESC' to quit
            break
    cap.release()
cv2.destroyAllWindows()
```



Mã ở trên sẽ ghi lại luồng video sẽ được tạo ra bởi PiCam của bạn, hiển thị cả hai chế độ, màu BGR và chế độ Xám. Bạn cũng có thể tải xuống mã [CamTest.py từ trên. Bây giờ, hãy chạy chương trình:](#)

```
python CamTest.py
```

Một số người có thể gặp sự cố khi cố gắng mở camera (thông báo lỗi "Assertion failed"). Điều đó có thể xảy ra nếu camera không được bật trong quá trình cài đặt OpenCv và do đó, trình điều khiển camera không được cài đặt đúng cách. Để sửa lỗi, hãy sử dụng lệnh:

```
sudo modprobe bcm2835-v4l2
```

Bước 3: Phát hiện khuôn mặt

Nhiệm vụ cơ bản nhất của Nhận dạng khuôn mặt tất nhiên là "Phát hiện khuôn mặt". Trước hết, bạn phải "chụp" một khuôn mặt để nhận dạng nó, khi so sánh với một khuôn mặt mới được chụp trong tương lai.

TẬP ĐỌC TÔI

Phát hiện đối tượng đang bộ phận loại tăng dựa trên đặc điểm Haar là một phương pháp phát hiện đối tượng hiệu quả được Paul Viola và Michael Jones đề xuất trong bài báo "Phát hiện đối tượng nhanh bằng cách sử dụng tầng Boosted Cascade of Simple Features" vào năm 2001. Đây là phương pháp tiếp cận dựa trên máy học, trong đó một hàm tăng được đào tạo từ nhiều hình ảnh tích cực và tiêu cực. Sau đó, nó được sử dụng để phát hiện đối tượng trong các hình ảnh khác.

Ở đây chúng ta sẽ làm việc với phát hiện khuôn mặt. Ban đầu, thuật toán cần rất nhiều hình ảnh dương (hình ảnh khuôn mặt) và hình ảnh âm (hình ảnh không có khuôn mặt) để đào tạo bộ phân loại. Sau đó, chúng ta cần trích xuất các đặc điểm từ nó. Tin tốt là OpenCV đi kèm với một trình đào tạo cũng như một trình phát hiện. Nếu bạn muốn đào tạo bộ phân loại của riêng mình cho bất kỳ đối tượng nào như ô tô, máy bay, v.v., bạn có thể sử dụng OpenCV để tạo một bộ phân loại. Chi tiết đầy đủ của nó được cung cấp tại đây: [Đào tạo bộ phân loại theo tầng](#).

Đủ lý thuyết rồi, chúng ta hãy cùng tạo một trình phát hiện khuôn mặt bằng OpenCV!

Tải xuống tệp: [FaceDetection.py](#)

```
import numpy as np
import cv2
faceCascade = cv2.CascadeClassifier('/home/pi/opencv-3.4.1/data/haarcascades/haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)
cap.set(3,640) # set Width
cap.set(4,480) # set Height
while True:
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.2,
        minNeighbors=5,
        minSize=(20,20)
    )
    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]
        cv2.imshow('video',img)
        k = cv2.waitKey(30) & 0xff
        if k == 27: # press 'ESC' to quit
            break
    cap.release()
    cv2.destroyAllWindows()
```

Khi bạn so sánh với mã cuối cùng được sử dụng để kiểm tra máy ảnh, bạn sẽ nhận ra rằng có một vài phần được thêm vào. Lưu ý dòng bên dưới:

```
faceCascade = cv2.CascadeClassifier('Cascades/haarcascade_frontalface_default.xml')
```

Đây là dòng tải "phân loại" (phải nằm trong thư mục có tên "Cascades/", trong thư mục dự án của bạn).

Sau đó, chúng ta sẽ thiết lập camera và bên trong vòng lặp, tải video đầu vào ở chế độ thang độ xám (giống như chúng ta đã thấy trước đó).

Bây giờ chúng ta phải gọi hàm phân loại của mình, truyền cho nó một số tham số rất quan trọng, như hệ số tỷ lệ, số lượng hàng xóm và kích thước tối thiểu của khuôn mặt được phát hiện.

```
faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.2,
    minNeighbors=5,
    minSize=(20, 20)
)
```



Ở đây,

- * `gray` là hình ảnh thang độ xám đầu vào.
- * `scaleFactor` là tham số chỉ định kích thước hình ảnh được giảm bao nhiêu ở mỗi tỷ lệ hình ảnh. Nó được sử dụng để tạo kim tự tháp tỷ lệ.
- * `minNeighbors` là tham số chỉ định số lượng hàng xóm mà mỗi hình chữ nhật ứng viên nên có, để giữ nguyên nó. Số cao hơn cho kết quả dương tính giả thấp hơn.
- * `minSize` là kích thước hình chữ nhật tối thiểu để được coi là một khuôn mặt.

Chức năng này sẽ phát hiện khuôn mặt trên hình ảnh. Tiếp theo, chúng ta phải "đánh dấu" các khuôn mặt trong hình ảnh, ví dụ, sử dụng hình chữ nhật màu xanh. Điều này được thực hiện bằng phần mã này:

```
for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]
```



Nếu tìm thấy khuôn mặt, nó sẽ trả về vị trí của các khuôn mặt được phát hiện dưới dạng hình chữ nhật có góc trên bên trái (x,y) và có "w" là Chiều rộng và "h" là Chiều cao ==> (x,y,w,h).

Bây giờ, hãy chạy tập lệnh python ở trên trên môi trường python của bạn bằng cách sử dụng Raspberry Pi Terminal:

```
python FaceDetection.py
```

Sau khi thực thi đoạn mã trên, bạn sẽ thấy một cửa sổ bật lên có chứa khuôn mặt của bạn.

Bạn cũng có thể bao gồm các bộ phân loại cho "phát hiện mắt" hoặc thậm chí "phát hiện nụ cười". Trong những trường hợp đó, bạn sẽ bao gồm hàm phân loại và vẽ hình chữ nhật bên trong vòng lặp khuôn mặt, vì sẽ không có ý nghĩa gì khi phát hiện mắt hoặc nụ cười bên ngoài khuôn mặt.

[*.FaceEyeDetection.py](#) * [.FaceSmileDetection.py](#) * [.FaceEyeSmileDetection.py](#)

Bước 4: Thu thập dữ liệu

Hãy bắt đầu giai đoạn đầu tiên của dự án. Những gì chúng ta sẽ làm ở đây là bắt đầu từ Phát hiện khuôn mặt, chúng ta sẽ chỉ cần tạo một tập dữ liệu, nơi chúng ta sẽ lưu trữ cho mỗi id, một nhóm ảnh màu xám với phần được sử dụng để phát hiện khuôn mặt.

Đầu tiên, hãy tạo một thư mục nơi bạn phát triển dự án của mình, ví dụ: FaceRecognition:

```
mkdir FaceRecognition
```

Trong thư mục này, ngoài 3 tập lệnh python mà chúng ta sẽ tạo cho dự án của mình, chúng ta phải lưu Facial Classifier vào đó. Bạn có thể tải xuống từ phía trên: [haarcascade_frontalface_default.xml](#)

Tiếp theo, tạo một thư mục con nơi chúng ta sẽ lưu trữ các mẫu khuôn mặt và đặt tên là "dataset":

```
mkdir dataset
```

Tải xuống [01_face_dataset.py](#)

```
import cv2
import os
```



```

cam = cv2.VideoCapture(0)
cam.set(3, 640) # set video width
cam.set(4, 480) # set video height
face_detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# For each person, enter one numeric face id
face_id = input('\n Enter user ID end press <Enter> ==> ')
print("\n Initializing face capture. Look the camera and wait ...")
# Initialize individual sampling face count
count = 0
while(True):
    ret, img = cam.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_detector.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
        count += 1
        # Save the captured image into the datasets folder
        cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) + ".jpg", gray[y:y+h,x:x+w])
        cv2.imshow('image', img)
    k = cv2.waitKey(100) & 0xff # Press 'ESC' for exiting video
    if k == 27:
        break
    elif count >= 30: # Take 30 face sample and stop video
        break
# Do a bit of cleanup
print("\n [INFO] Exiting Program and cleanup stuff")
cam.release()
cv2.destroyAllWindows()

```

Mã này rất giống với mã mà chúng ta thấy để phát hiện khuôn mặt. Những gì chúng ta thêm vào là một "lệnh nhập" để nắm bắt ID người dùng, đó phải là một số nguyên (1, 2, 3, v.v.)

```
face_id = input('\n enter user id end press ==> ')
```

Và đối với mỗi khung hình đã chụp, chúng ta nên lưu nó dưới dạng một tệp trong thư mục "bộ dữ liệu":

```
cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) + ".jpg", gray[y:y+h,x:x+w])
```

Lưu ý rằng để lưu tệp trên, bạn phải nhập thư viện "os". Tên của mỗi tệp sẽ theo cấu trúc sau: User.face_id.count.jpg

Ví dụ, đối với người dùng có face_id = 1, tệp mẫu thứ 4 trên dataset/thư mục sẽ giống như sau:

```
User.1.4.jpg
```

Trong code, các bạn có thể thấy có một count biến đếm số lượng hình ảnh được chụp. Biến này được đặt thành 30, chụp càng nhiều thì hệ thống sẽ hoạt động càng chính xác.

Bước 6: Huấn luyện viên

Ở giai đoạn thứ hai này, chúng ta phải lấy tất cả dữ liệu người dùng từ tập dữ liệu của mình và "huấn luyện" OpenCV Recognizer. Điều này được thực hiện trực tiếp bởi một hàm OpenCV cụ thể. Kết quả sẽ là một .xml tệp được lưu trong thư mục "trainer/".

Vậy thì trước tiên chúng ta hãy tạo thư mục trainer:

```
mkdir trainer
```

Tải xuống [02_face_training.py](#)

```

import cv2
import numpy as np
from PIL import Image
import os
# Path for face image database
path = 'dataset'

os.chdir("/home/pi/opencv-3.4.1/data/haarcascades")
recognizer = cv2.face.LBPHFaceRecognizer_create()

```



```

detector = cv2.CascadeClassifier("/home/pi/opencv-3.4.1/data/haarcascades/haarcascade_frontalface_default.xml");
# function to get the images and label data
def getImagesAndLabels(path):
    imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
    faceSamples=[]
    ids = []
    for imagePath in imagePaths:
        PIL_img = Image.open(imagePath).convert('L') # convert it to grayscale
        img_numpy = np.array(PIL_img,'uint8')
        id = int(os.path.split(imagePath)[-1].split(".")[1])
        faces = detector.detectMultiScale(img_numpy)
        for (x,y,w,h) in faces:
            faceSamples.append(img_numpy[y:y+h,x:x+w])
            ids.append(id)
    return faceSamples,ids
print ("\nTraining faces. It will take few seconds. Wait ...")
faces,ids = getImagesAndLabels(path)
recognizer.train(faces, np.array(ids))
# Save the model into trainer/trainer.yml
recognizer.write('/home/pi/FaceRecognition/trainer/trainer.yml')
# Print the number of faces trained and end program
print("\n [INFO] {0} faces trained. Exiting Program".format(len(np.unique(ids))))

```

Xác nhận xem bạn đã cài đặt thư viện PIL trên Raspberry Pi chưa. Nếu chưa, hãy chạy lệnh bên dưới trong Terminal:

```
pip install pillow
```

Chúng tôi sẽ sử dụng LBPH (LOCAL BINARY PATTERNS HISTOGRAMS) Face Recognizer làm trình nhận dạng, có trong gói OpenCV. Có thể thực hiện bằng cách sau:

```
recognizer = cv2.face.LBPHFaceRecognizer_create()
```

Hàm "getImagesAndLabels(path)", sẽ lấy tất cả ảnh trong thư mục: "dataset/", trả về 2 mảng: "Ids" và "faces". Với các mảng đó làm đầu vào, chúng ta sẽ "huấn luyện trình nhận dạng":

```
recognizer.train(faces, ids)
```

Kết quả là, một tệp có tên "trainer.yml" sẽ được lưu trong thư mục trainer mà chúng tôi đã tạo trước đó.

Lưu ý: Đảm bảo rằng bất cứ khi nào bạn thu thập dữ liệu, tức là chạy chương trình 1, bạn cũng phải chạy chương trình 2 để đào tạo Rpi.

Bước 6: Nhận dạng

Bây giờ, chúng ta đã đến giai đoạn cuối cùng của dự án. Ở đây, chúng ta sẽ chụp một khuôn mặt mới trên máy ảnh của mình và nếu người này đã chụp và đào tạo khuôn mặt trước đó, trình nhận dạng của chúng ta sẽ đưa ra "dự đoán" trả về id và chỉ mục, cho thấy trình nhận dạng tự tin như thế nào với sự trùng khớp này.

Tải xuống [03_face_recognition.py](#)

```

import cv2
import numpy as np
import os

os.chdir("/home/pi/opencv-3.4.1/data/haarcascades")
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('/home/pi/FaceRecognition/trainer/trainer.yml')
cascadePath = "/home/pi/opencv-3.4.1/data/haarcascades/haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);

font = cv2.FONT_HERSHEY_SIMPLEX

#iniciate id counter
id = 0

# names related to ids: example ==> KUNAL: id=1, etc
names = ['None', 'Kunal', 'Kaushik', 'Atharv', 'Z', 'W']

```

```

# Initialize and start realtime video capture
cam = cv2.VideoCapture(0)
cam.set(3, 640) # set video width
cam.set(4, 480) # set video height

# Define min window size to be recognized as a face
minW = 0.1*cam.get(3)
minH = 0.1*cam.get(4)

while True:
    ret, img = cam.read()
    #img = cv2.flip(img, -1) # Flip vertically
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor = 1.2,
        minNeighbors = 5,
        minSize = (int(minW), int(minH)),
    )

    for(x,y,w,h) in faces:
        cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
        id, confidence = recognizer.predict(gray[y:y+h,x:x+w])

        # Check if confidence is less than 100 ==> "0" is perfect match
        if (confidence < 100):
            id = names[id]
            confidence = " {0}%".format(round(100 - confidence))
        else:
            id = "unknown"
            confidence = " {0}%".format(round(100 - confidence))

        cv2.putText(img, str(id), (x+5,y-5), font, 1, (255,255,255), 2)
        cv2.putText(img, str(confidence), (x+5,y+h-5), font, 1, (255,255,0), 1)

    cv2.imshow('camera',img)

    k = cv2.waitKey(10) & 0xff # Press 'ESC' for exiting video
    if k == 27:
        break

# Do a bit of cleanup
print("\n [INFO] Exiting Program and cleanup stuff")
cam.release()
cv2.destroyAllWindows()

```

Chúng tôi sẽ đưa vào đây một mảng mới, vì vậy chúng tôi sẽ hiển thị "tên", thay vì các id được đánh số:

names = ['None', 'Kunal', 'Kaushik', 'Tushar', 'X', 'Y', 'Z'] Vì vậy, ví dụ: Kunal sẽ là người dùng có id = 1; Kaushik: id=2, v.v.

Tiếp theo, chúng ta sẽ phát hiện khuôn mặt, giống như chúng ta đã làm trước đó với bộ phân loại haarCascade. Khi đã phát hiện được khuôn mặt, chúng ta có thể gọi hàm quan trọng nhất trong đoạn mã trên:

```
id, confidence = recognizer.predict(gray portion of the face)
```

, recognizer.predict () sẽ lấy một phần khuôn mặt được chụp làm tham số để phân tích và sẽ trả về chủ sở hữu có thể có của khuôn mặt đó, cho biết ID của chủ sở hữu và mức độ tin cậy của bộ nhận dạng đối với sự trùng khớp này.

Lưu ý rằng chỉ số tin cậy sẽ trả về "không" nếu nó được coi là khớp hoàn hảo

Dưới đây là hình ảnh kết quả cuối cùng, bạn cũng sẽ nhận được kết quả tương tự



Phần kết luận:

Tôi hy vọng dự án này có thể giúp mọi người tìm được cách bước vào thế giới IoT thú vị!

Phát hành

Không có bản phát hành nào được công bố

Gói hàng

Không có gói nào được công bố

Ngôn ngữ

● Trăn 100,0%