VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Phát triển Ứng dụng Internet Of Things

Lab 02

Lab2

Advisor(s): Th.S Vũ Trọng Thiên
Student(s): Huỳnh Nguyễn Hiếu Nhân 2013961

HO CHI MINH CITY, FEBRUARY 2024

Mục lục

1 Tích hợp Google Teachable Machine **4**
1.1 Hiện thực trên Windows với VS code 4
1.1.1 Sử dụng Google teachable machine 4
1.1.2 Code 6
1.1.3 Kết quả 11
1.2 Hiện thực trên Linux với môi trường WSL tích hợp trong VS Code 12
1.2.1 Kết quả 12

2 Đọc dữ liệu từ cảm biến **16**
2.1 Hiện thực trên Windows với VS code 16
2.1.1 Tạo cổng com ảo bằng COM0COM 16
2.1.2 Code và giải thích 17
2.1.3 Kết quả 21
2.2 Hiện thực trên Linux với môi trường WSL tích hợp trong VS Code 22
2.2.1 Cài đặt và sử dụng socat và minicom trên WSL để thực hiện mô phỏng 22
2.2.2 Kết quả 23

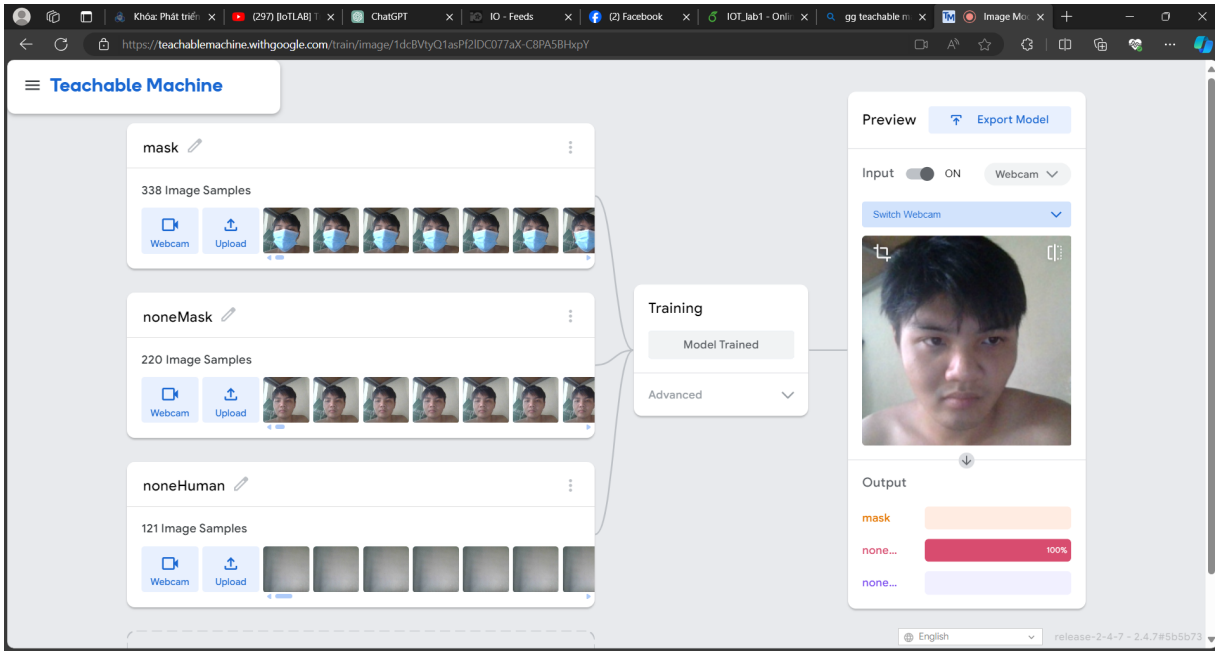
3 Link Github source code **25**

1 Tích hợp Google Teachable Machine

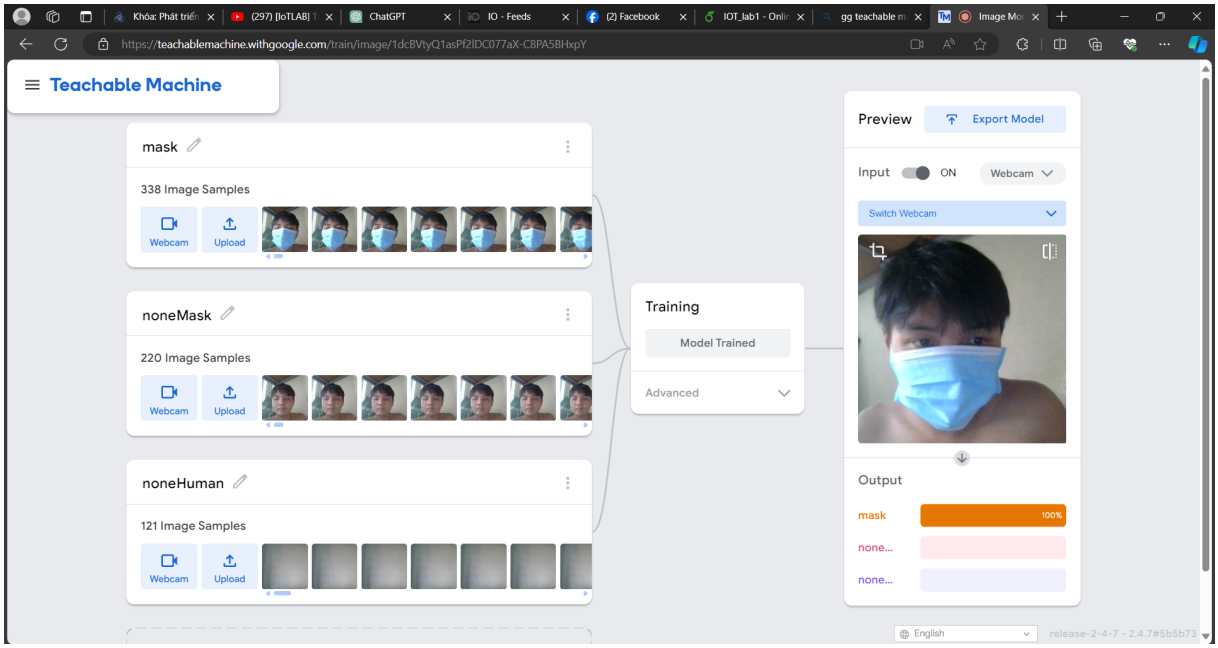
1.1 Hiện thực trên Windows với VS code

1.1.1 Sử dụng Google teachable machine

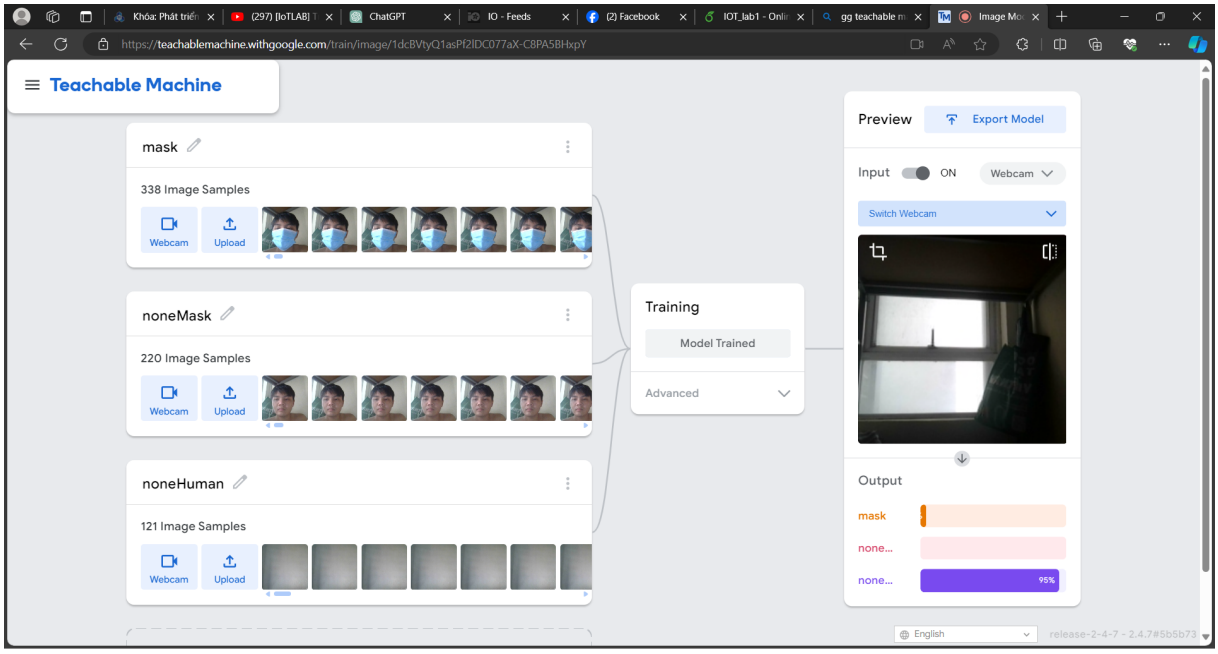
Model này sẽ được train với 3 class là mask (có đeo khẩu trang), noMask (không đeo khẩu trang) và noneHuman (không có người)



Hình 1.1: Nhận diện không đeo khẩu trang



Hình 1.2: Nhận diện đeo khẩu trang



Hình 1.3: nhận diện không có người

1.1.2 Code

File AI_model.py:

```
from keras.models import load_model  # TensorFlow is required for Keras to work
import cv2  # Install opencv-python
import numpy as np
import queue
import threading

class Camera:
    def __init__(self,source):
        self.camera = self.get_camera(source)
        # self.frame_buffer = queue.Queue()
        self.stream_thread = threading.Thread(target=self.run)
        self.stream_thread.daemon = True
        self.stream_thread.start()
        self.last_frame = None

    def get_last_image(self):
        #return bool, img
        return self.last_frame

    def get_camera(self,source):
        return cv2.VideoCapture(source)

    def release_camera(self):
        self.camera.release()

    def run(self):
        while True:
            ret, self.last_frame = self.camera.read()
            if not ret:
                break
        # self.frame_buffer.put(self.last_frame)
```

```python



class AI_model:
    def __init__(self,model_file,label_file,source = 0):
        self.model_file = model_file
        self.label_file = label_file
        self.camera = Camera(source)
        # Load the model
        self.model = load_model(model_file, compile=False)
        # Load the labels
        self.class_names = open(label_file, "r").readlines()

    def predict(self,image):
        # Resize the raw image into (224-height,224-width) pixels
        image = cv2.resize(image, (224, 224), interpolation=cv2.INTER_AREA)
        # Make the image a numpy array and reshape it to the models input shape.
        image = np.asarray(image, dtype=np.float32).reshape(1, 224, 224, 3)

        # Normalize the image array
        image = (image / 127.5) - 1

        # Predicts the model
        prediction = self.model.predict(image)
        index = np.argmax(prediction)

        # Print prediction and confidence score
        class_name = self.class_names[index].strip("\n")
        confidence_score = prediction[0][index]

        print("Class:", class_name[2:], end=",")
```

```
        print("Confidence Score:", str(np.round(
            confidence_score * 100))[:-2], "%")
        return class_name[2:], confidence_score


    def image_detector(self):
    # Grab the webcamera's image.
        image = self.camera.get_last_image()
        cv2.imshow("my camera",image)
        return self.predict(image)
```

Class Camera sẽ thực hiện các nhiệm vụ liên quan đến đọc videos từ thiết bị hoặc từ một nguồn stream bằng link url video stream. Khi khởi tạo class Camera thì class này sẽ tạo luồng và đọc frame liên tục từ nguồn từ **method run()**. Để lấy ảnh của Camera thì ta chỉ cần dùng **method get_last_image()**

Class AI_model khi khởi tạo sẽ cần cung cấp đường dẫn đến file model AI và file label tương ứng, ngoài sẽ còn phải cung cấp thêm nguồn input ảnh là từ thiết bị của PC hay là từ link url stream khác để khởi tạo 1 class Camera. Ở đây, nếu source = 0 tức là ta dùng camera của chình máy tính. Ngoài ra ta còn có thể dùng link url của một nguồn stream nào đó.

Quan trọng nhất ở class này là **method predict** sẽ nhận input là 1 frame ảnh và trả về kết quả nhận diện có xác suất cao nhất và xác suất của kết quả. **Method image_detector()** sẽ đơn giản là lấy frame ảnh của camera và sử dụng lại **method predict** để dự đoán kết quả.

File main.py:

```
from keras.models import load_model  # TensorFlow is required
    for Keras to work
import cv2  # Install opencv-python
import numpy as np
import time
import threading
#import my module
from AI_model import AI_model
from mqtt import MQTT_Module
from uart import Serial_module
```

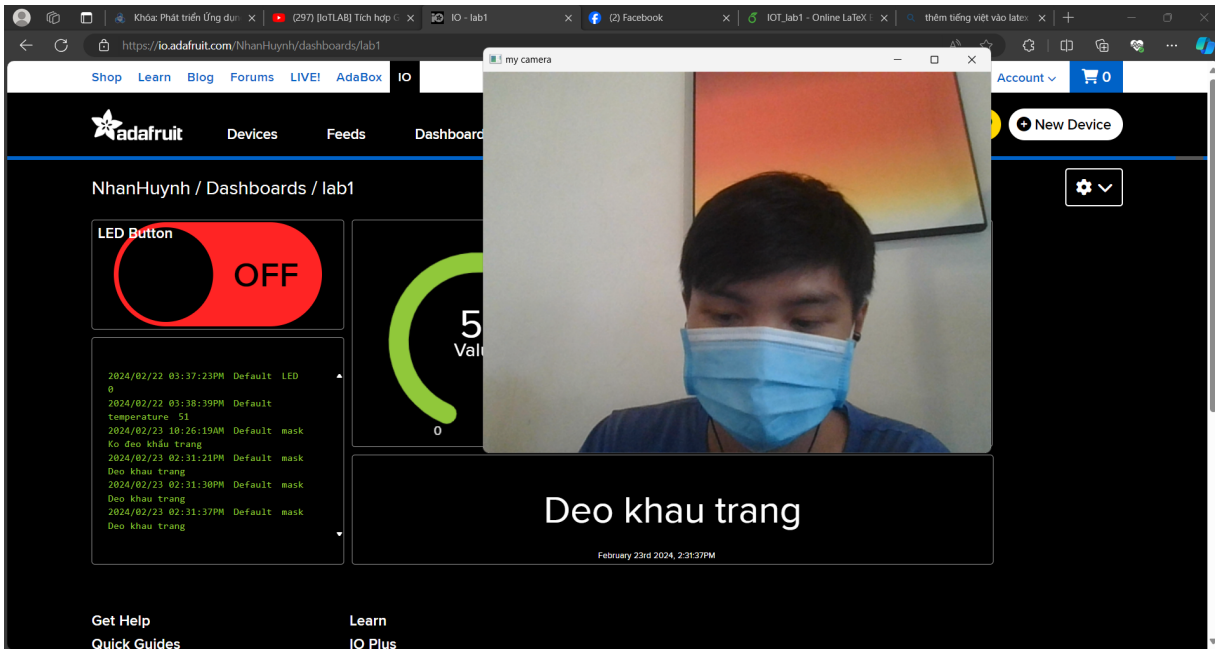
```

AIO_FEED_ID  = [
    "NhanHuynh/feeds/led",
    "NhanHuynh/feeds/temperature" ,
    "NhanHuynh/feeds/mask"
]
AIO_USERNAME     = "NhanHuynh"
AIO_KEY          = "aio_cjdw41TQxexd6tdbOQ2mW7bYdc3r"
BROKER_ADDRESS   = "io.adafruit.com"
PORT             = 1883

# URL              = "http://192.168.9.67:8080/video"



result_in_Vietnamese = {
    "mask"       : "Deo khau trang",
    "noneMask"   : "Ko deo khau trang",
    "noneHuman"  : "Ko co nguoi"
}
def main():
    # Disable scientific notation for clarity
    np.set_printoptions(suppress=True)
    my_ai = AI_model("model/keras_model.h5","model/labels.txt
        ",0)
    mqtt_handler = MQTT_Module(AIO_USERNAME, AIO_KEY,
        AIO_FEED_ID, BROKER_ADDRESS, PORT)
    #create a counter
    count_ai = 0
    while True:

        if count_ai > 5:
            count_ai = 0
            result, _ = my_ai.image_detector()
            mqtt_handler.publish("NhanHuynh/feeds/mask",
```

```
                result_in_Vietnamese[result])
        else:
            count_ai += 1
        # Listen to the keyboard for presses.
        keyboard_input = cv2.waitKey(1)
        # 27 is the ASCII for the esc key on your keyboard.
        if keyboard_input == 27:
            break
        time.sleep(1)

    my_ai.camera.release_camera()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()
```

mqtt_handler được hiện thực giống với lab1 nhưng được chỉnh sửa, đóng gói để tạo thành 1 class. Khi khởi tạo sẽ truyền vào tài khoản Adafruit IO, access token, danh sách feed cần subcribe nếu có, địa chỉ Broker và port.
count_ai được dùng để xác định thời gian detect ở đây thì sau mỗi 5s thì sẽ thực hiện detect ảnh và gửi kết quả lên feed "NhanHuynh/feeds/mask" tương ứng với danh sách trong biến **result_in_Vietnamese**

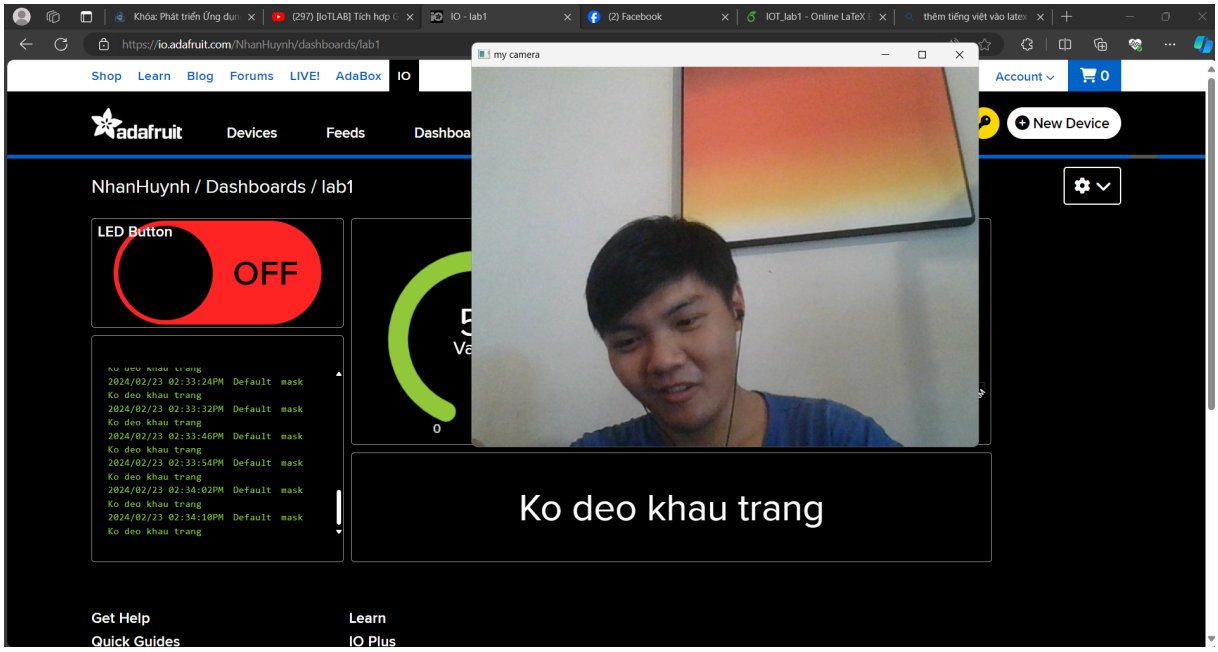
1.1.3 Kết quả

Kết quả nhận diện khi đeo khẩu trang



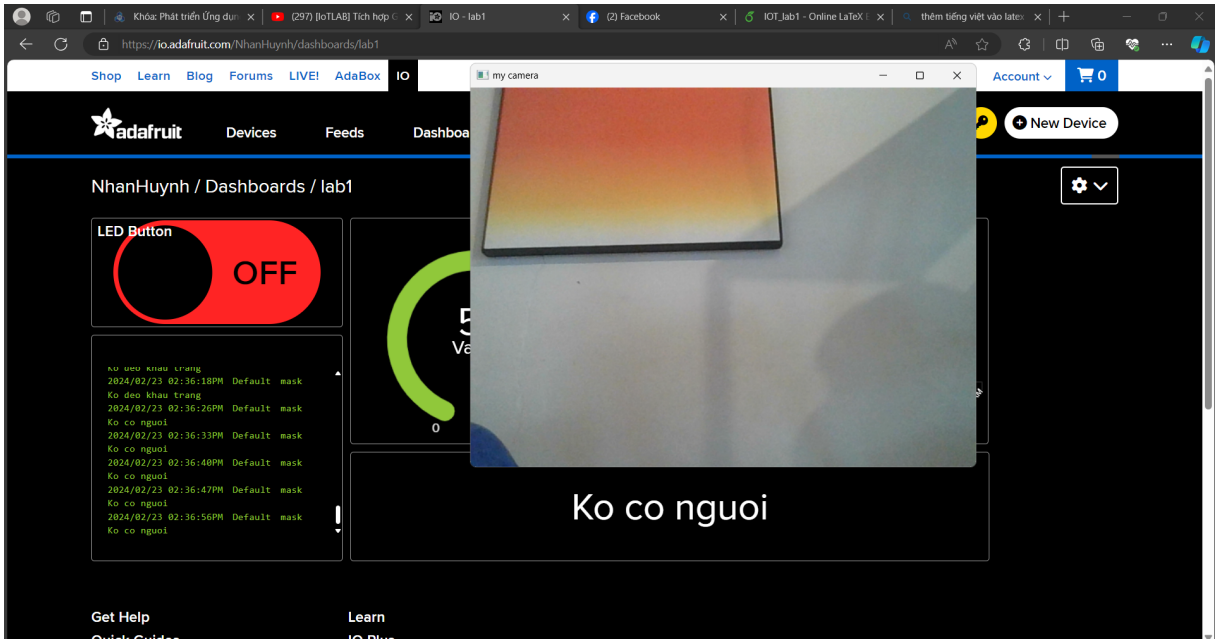
Hình 1.4: Kết quả model nhận diện khi đeo khẩu trang

Kết quả nhận diện khi không đeo khẩu trang



Hình 1.5: Kết quả model nhận diện khi không đeo khẩu trang

Kết quả nhận diện khi không có người



Hình 1.6: Kết quả model nhận diện khi không có người

1.2 Hiện thực trên Linux với môi trường WSL tích hợp trong VS Code

Vì WSL không thể sử dụng Camera của máy do không có drive trong kernel linux, do đó để sử dụng được camera thì sẽ dùng app chạy trên android là IP Webcam hoặc DroidCam. Hai app này sẽ host điện thoại như một server và sẽ cung cấp 1 link url để cho chúng ta có thể sử dụng link này như một nguồn stream.Lưu ý là máy tính và điện thoại android phải cùng một mạng với nhau.

Ta chỉ cần thay đổi từ source = 0 thành source=url ở lúc khởi tạo Class AI_model:

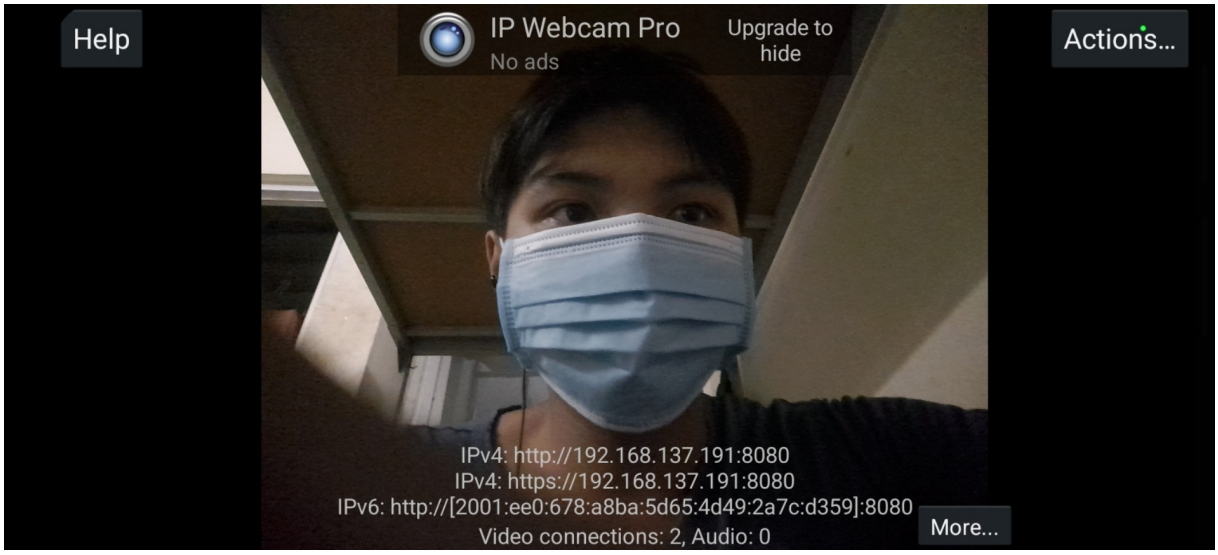
```
url = "http://192.168.137.4:8080/video"
my_ai = AI_model("model/keras_model.h5","model/labels.txt",
    url)
```

1.2.1 Kết quả

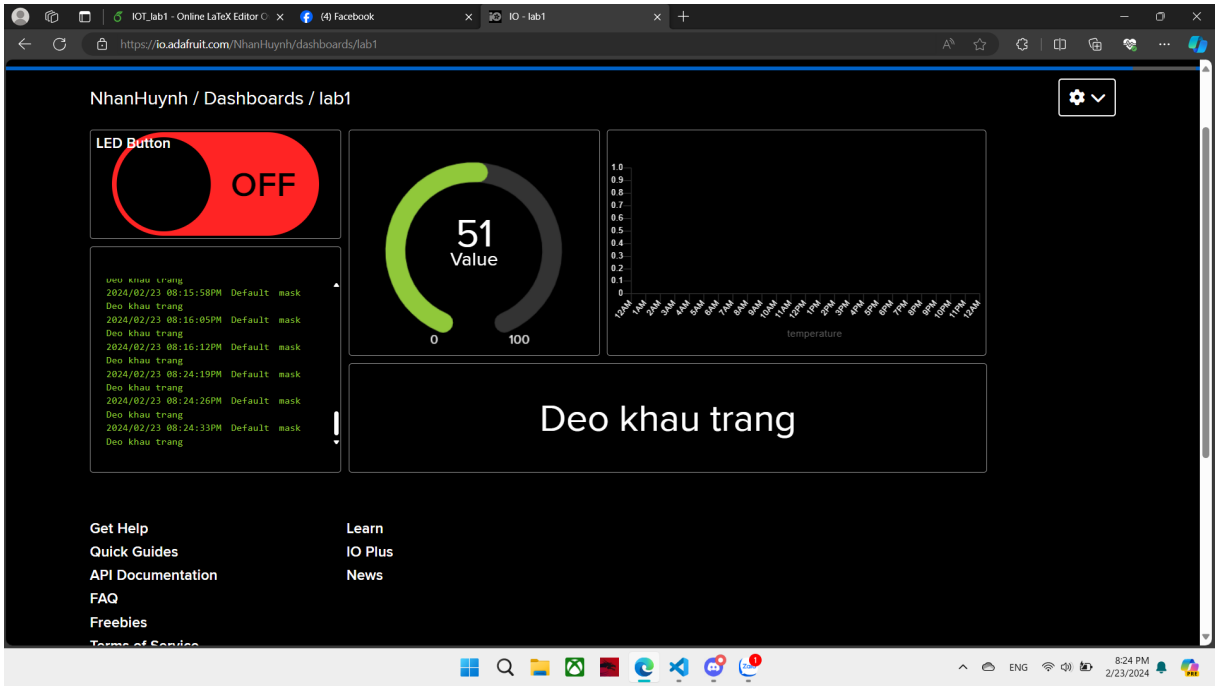
Trường hợp có đeo khẩu trang

Input

Output



Hình 1.7: ảnh đầu vào camera điện thoại có đeo khẩu trang



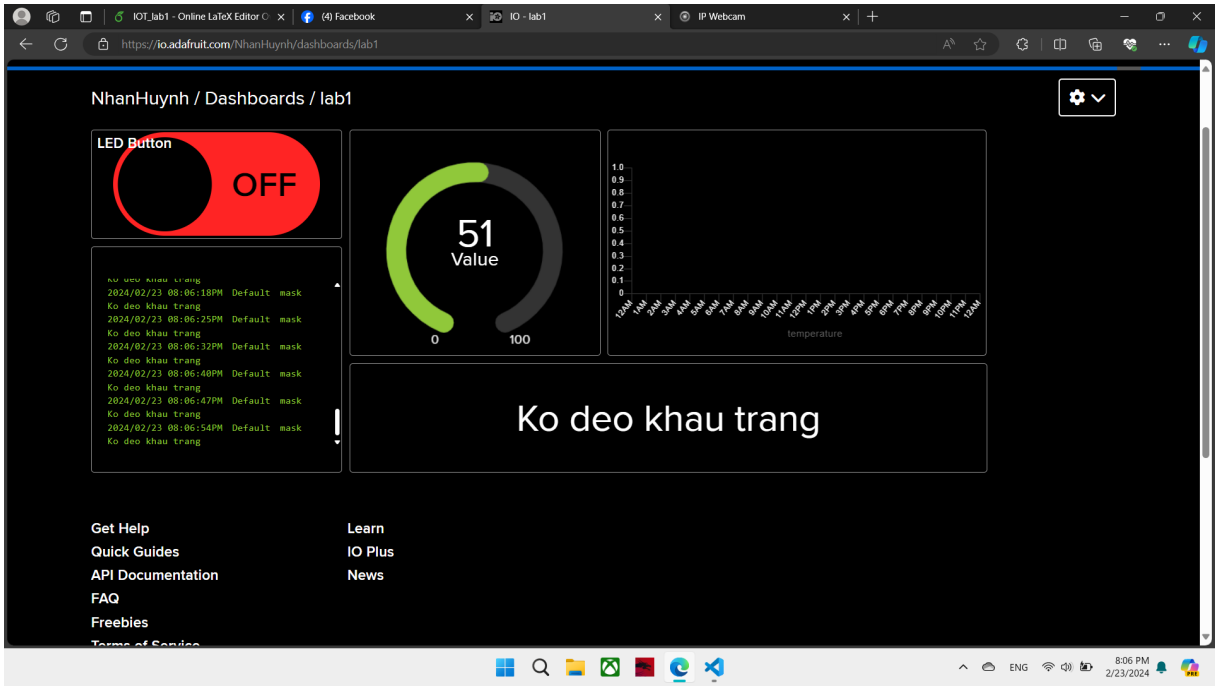
Hình 1.8: Kết quả detect trên camera điện thoại đeo khẩu trang

Trường hợp không đeo khẩu trang

Input

Hình 1.9: ảnh đầu vào camera điện thoại không đeo khẩu trang

Output

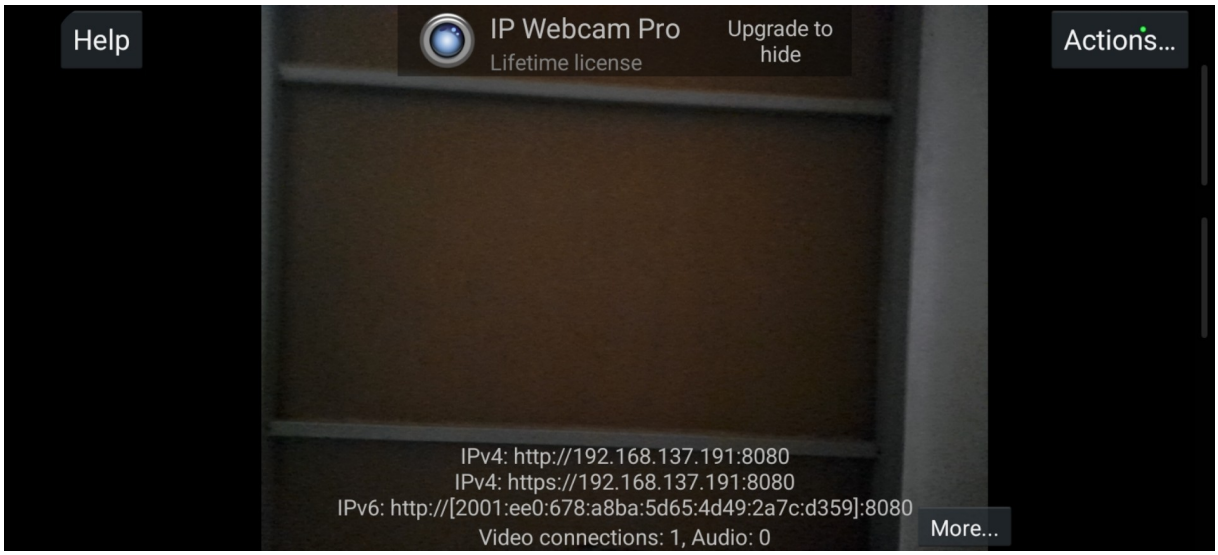


Hình 1.10: Kết quả detect trên camera điện thoại không đeo khẩu trang

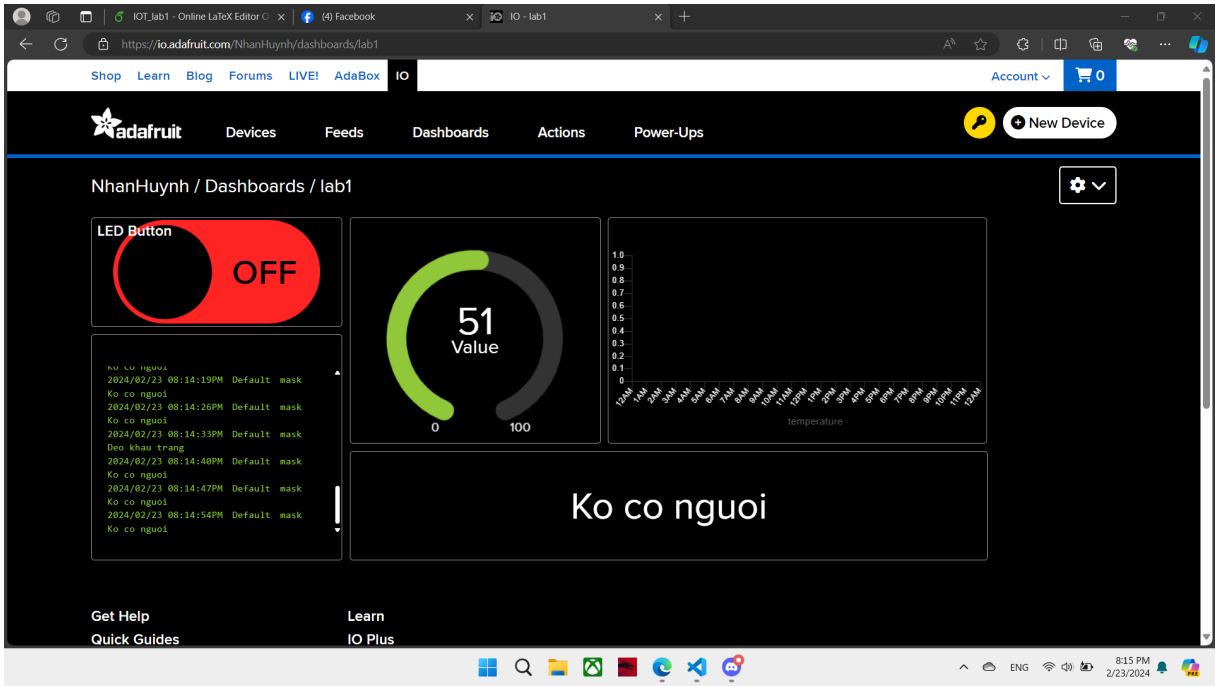
Trường hợp không có người

Input:

Output:



Hình 1.11: ảnh đầu vào detect trên camera điện thoại đeo khẩu trang

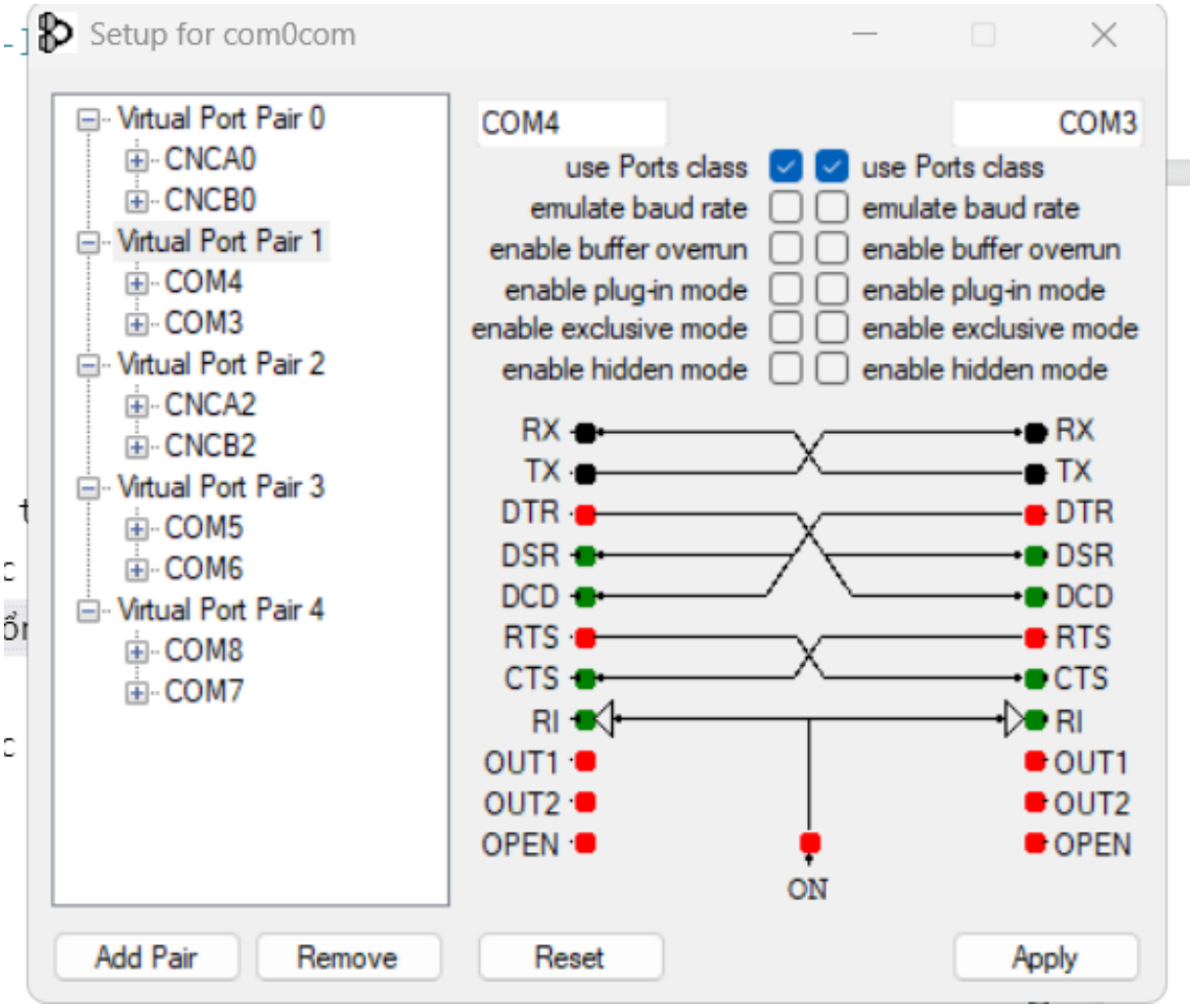


Hình 1.12: Kết quả detect trên camera điện thoại không có người

2 Đọc dữ liệu từ cảm biến

2.1 Hiện thực trên Windows với VS code

2.1.1 Tạo cổng com ảo bằng COM0COM



Hình 2.1: Com0Com setup

Tại đây ta sẽ sử dụng cặp COM3(code python) - COM4(hercules) để mô phỏng quá trình đọc dữ liệu từ cảm biến.

2.1.2 Code và giải thích

File uart.py:

```
import serial.tools.list_ports
import random
import time
import  sys

class Serial_module:
    serial_handler = None
    mess = None
    processData = None
    port = None
    baudrate = None

    def getPort():
            ports = serial.tools.list_ports.comports()
            N = len(ports)
            commPort = "None"
            for i in range(0, N):
                port = ports[i]
                strPort = str(port)
                if "USB Serial Device" in strPort:
                    splitPort = strPort.split(" ")
                    commPort = (splitPort[0])
            return commPort

    def __init__(self, port, baudrate):
        self.port = port if port is not None else self.getPort()
        self.baudrate = baudrate if baudrate is not None else 115200
        self.serial_handler = serial.Serial( self.port, self.baudrate)
        self.mess = ""
```

```
        self.processData = None

    def set_processData(self,func):
        self.processData = func

    def call_processData(self,*arg):
        if(self.processData is not None):
            self.processData(*arg)
        else:
            print("processData hasnot been setted yet")

    def readSerial(self,mqtt_handler):
        bytesToRead = self.serial_handler.inWaiting()
        if (bytesToRead > 0):

            self.mess = self.mess + self.serial_handler.read(
                bytesToRead).decode("UTF-8")
            while ("#" in self.mess) and ("!" in self.mess):
                start = self.mess.find("!")
                end = self.mess.find("#")
                print(self.mess[start:end + 1])
                self.call_processData(self.mess[start:end +
                    1],mqtt_handler)
                if (end == len(self.mess)):
                    self.mess = ""
                else:
                    self.mess = self.mess[end + 1:]
```

Class Serial_module được giống như hướng dẫn trên kênh youtube chỉ khác là được đóng gói thành 1 class.Tuy nhiên có điểm đặc biệt là **thuộc tính processData**, thuộc tính này được tạo ra nhằm để gán giá trị là một hàm số được định nghĩa ngoài tầm vực của class. Tương ứng với **thuộc tính processData** là 2 method **def set_processData(self,func)** và **def call_processData(self,*arg)**
Method **set_processData** là setter của thuộc tính processData, còn **call_processData** sẽ dùng để thực thi hàm thuộc tính processData đang mang.Lưu ý là các tham số truyền vào phải đúng với thức tự

Nguyên nhân phải thực thi như vậy là vì nhằm giúp linh động hơn khi tái sử dụng do mỗi project khác nhau sẽ cần hiện thực đoạn code processData khác nhau.

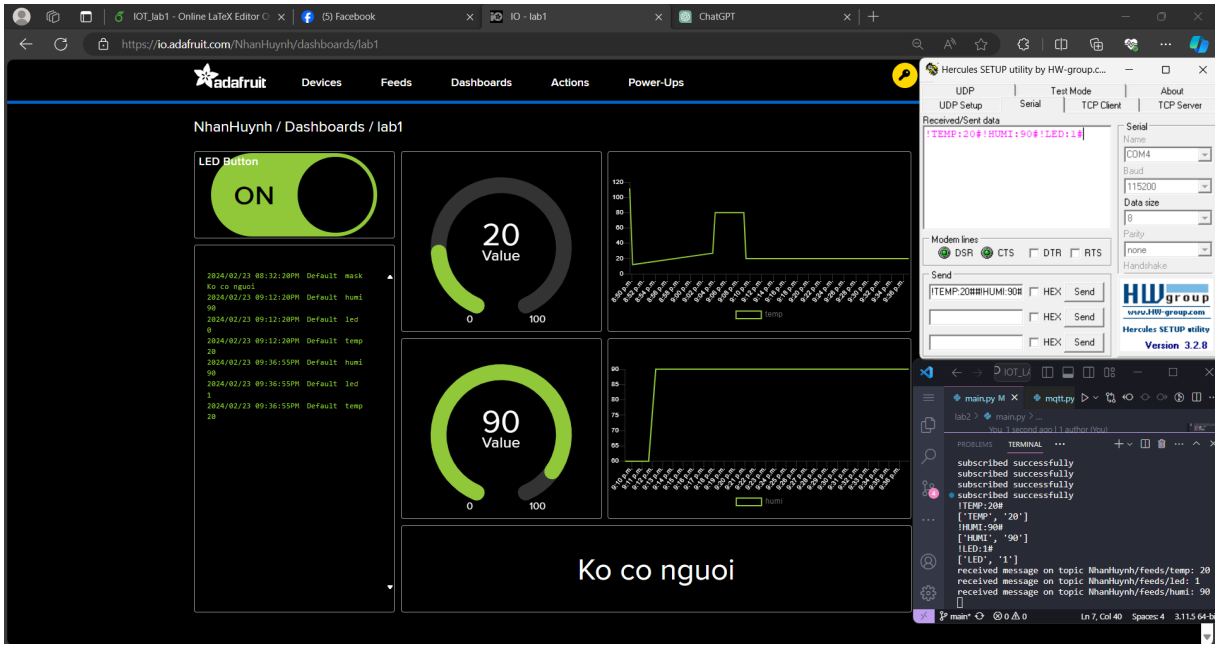
main.py:

```
import time
import threading
#import my module
from AI_model import AI_model, Camera
from mqtt import MQTT_Module
from uart import Serial_module

AIO_FEED_ID  = [
    "NhanHuynh/feeds/led",
    "NhanHuynh/feeds/temp" ,
    "NhanHuynh/feeds/mask",
    "NhanHuynh/feeds/humi"
]
AIO_USERNAME     = "NhanHuynh"
AIO_KEY          = "aio_zsWp90MrYtJoK2UOoiFU6XJ2csDK"
BROKER_ADDRESS   = "io.adafruit.com"
PORT             = 1883

def processData_external_define(data,mqtt_handler):
    data = data.replace("!", "")
    data = data.replace("#", "")
    splitData = data.split(":")
    print(splitData)
    if splitData[0] == "TEMP":
        mqtt_handler.publish("NhanHuynh/feeds/temp",splitData
            [1])
    elif splitData[0] == "LED":
        mqtt_handler.publish("NhanHuynh/feeds/led",splitData
            [1])
    elif splitData[0] == "HUMI":
```

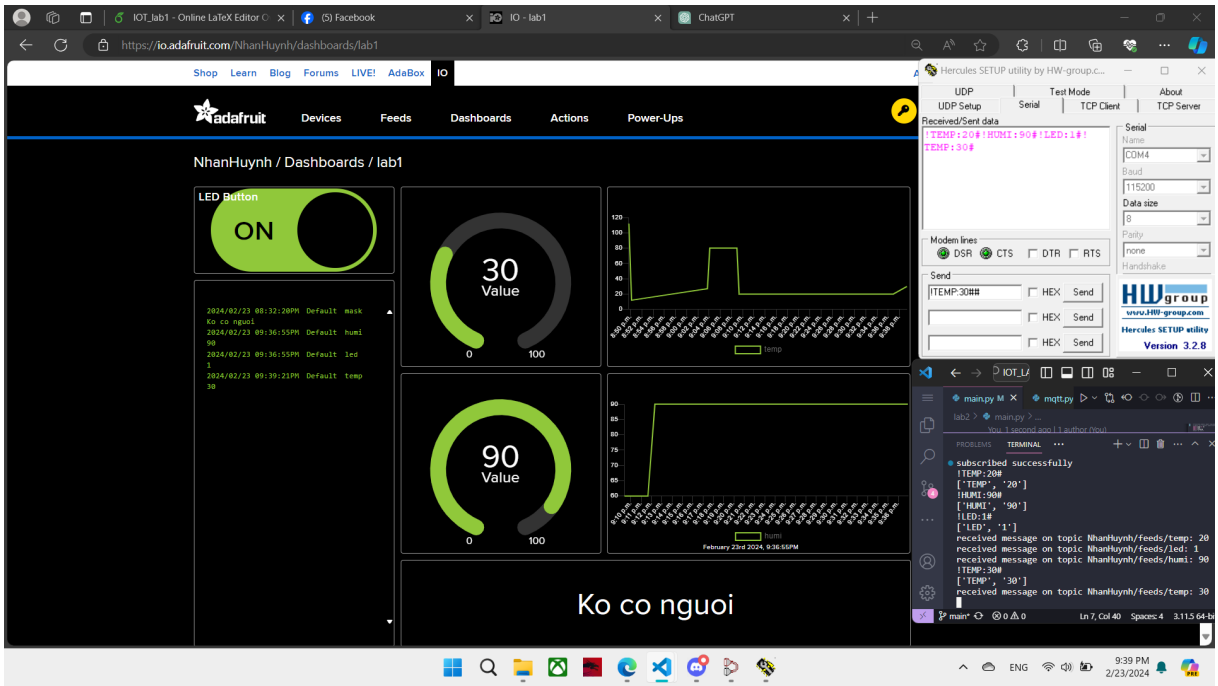
```
        mqtt_handler.publish("NhanHuynh/feeds/humi",splitData[1])
    else:
        print("we dont have that measurement to publish")
def main():
    # serial init
    my_serial = Serial_module("COM3",115200)
    my_serial.set_processData(processData_external)

    #mqtt init
    mqtt_handler = MQTT_Module(AIO_USERNAME, AIO_KEY, AIO_FEED_ID, BROKER_ADDRESS, PORT)

    while True:
        my_serial.readSerial(mqtt_handler)
        time.sleep(1)
```

Method processData_external_define được định nghĩa và dùng setter để gán vào thuộc tính processData của của **my_serial**(instance của class Serial_module). sau đó **method readSerial** của **my_serial** sẽ thực thi hàm này bằng method call_processData, với tham số truyền vào ở đây phải đúng thức với hàm được định nghĩa ở ngoài tầm vực của class.

2.1.3 Kết quả



Hình 2.2: kết quả khi gửi chuỗi "!TEMP:20#!HUMI:90#!LED:1#"



Hình 2.3: kết quả khi gửi chuỗi "!TEMP:30#"

2.2 Hiện thực trên Linux với môi trường WSL tích hợp trong VS Code

2.2.1 Cài đặt và sử dụng socat và minicom trên WSL để thực hiện mô phỏng

Giống tương tự như khi thực hiện trên window về code hiện thực tuy nhiên sẽ có khác biệt về quá trình mô phỏng vì WSL rất khó để dùng trực tiếp các device của máy tính PC của chúng. Nên để cho đơn giản, tương tự như mô phỏng trên window bằng herculer và com0com, ta cũng sẽ tạo các cổng ảo ở phía WSL bằng **socat** và **minicom** tải về bằng lệnh(lưu ý bản distro đang dùng là ubuntu20.14):

```
sudo apt install socat
sudo apt install minicom
```

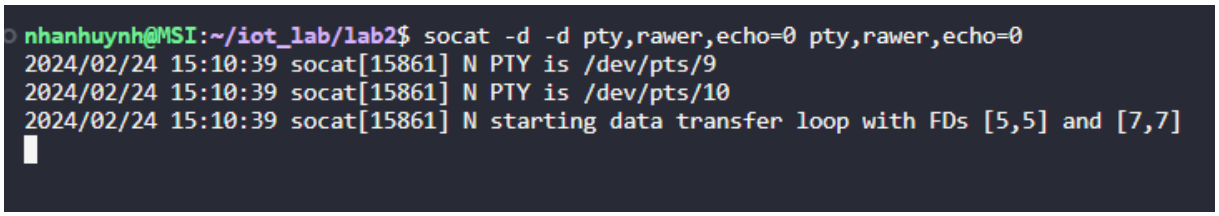
sau đó tạo 2 cổng ảo kết nối với nhau bằng lệnh:

```
socat -d -d pty,rawer,echo=0 pty,rawer,echo=0
```

sau khi chạy xong thì ta sẽ có được 2 đường dẫn của 2 cổng ảo được tạo. có thể thêm link=<path> với path chính tệp biểu diễn cho cổng mà bản thân muốn chỉ định.

```
socat -d -d pty,raw,echo=0,link=/dev/ttyS10 pty,raw,echo=0,
    link=/dev/ttyS11
```

sau khi thực thi lệnh ta sẽ thu được output như hình bên dưới với 2 cổng ảo được cấp là **/dev/pts/9** và **/dev/pts/10**.



Hình 2.4: Kết quả khi sử dụng socat tạo cổng ảo

Để mô phỏng quá trình Đọc dữ liệu từ cảm biến ta sẽ dùng minicom để gửi dữ liệu qua port **/dev/pts/10** và chương trình python của chúng ta sẽ nhận xử lý và đẩy dữ

liệu qua port **/dev/pts/9** lên adafruit io.

ta chỉ cần thay đổi đường port cần khởi tạo:

```
my_serial = Serial_module("/dev/pts/9",115200)
```

Khi chạy lệnh minicom để mở port ta thực hiện như sau:

```
sudo minicom -D /dev/pts/10 -b 115200
```

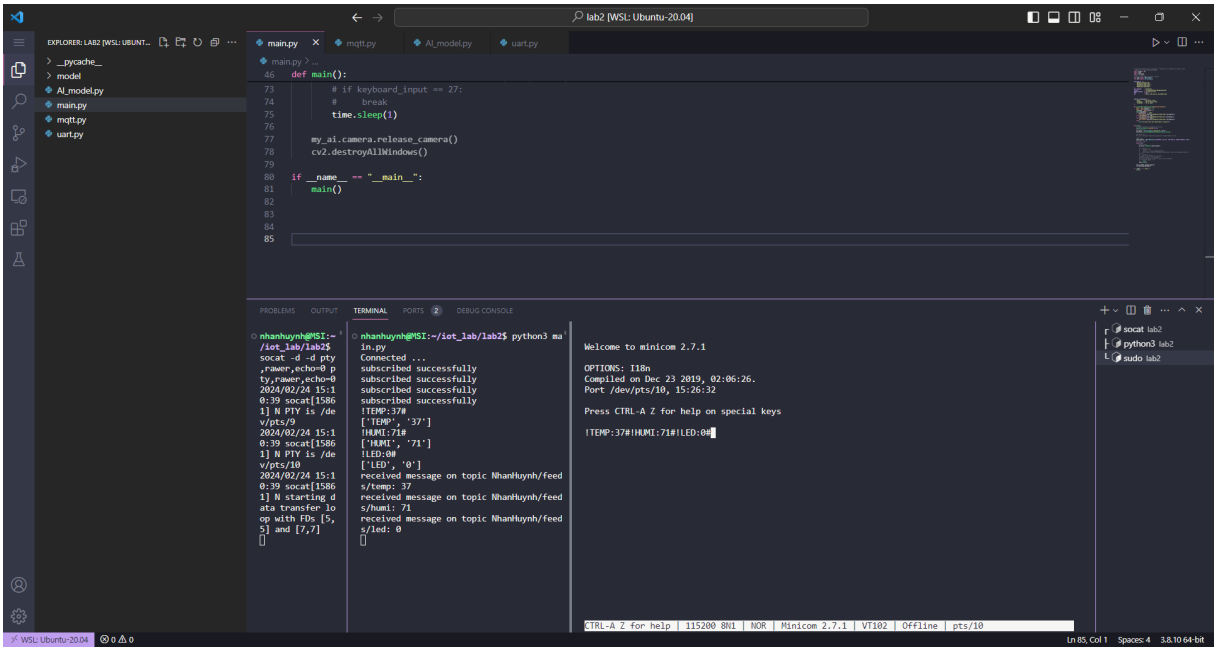
Khi gọi lệnh xong thì bất kì ta gõ lên port đều sẽ gửi tử port **/dev/pts/10** đến port **/dev/pts/9**, tuy nhiên thì dữ liệu sẽ không hiện ra những gì ta gõ nên để quan sát, ta sẻ thực hiện gọi lệnh:

```
sudo minicom -s
```

Sau đó chọn **Screen and keysboard**, sau đó nhấn **Q** để bật **local echo** để kích hoạt tính năng hiển thị chữ mà ta đã nhập, cuối bấm ESC và chọn **Set setup as dlf**

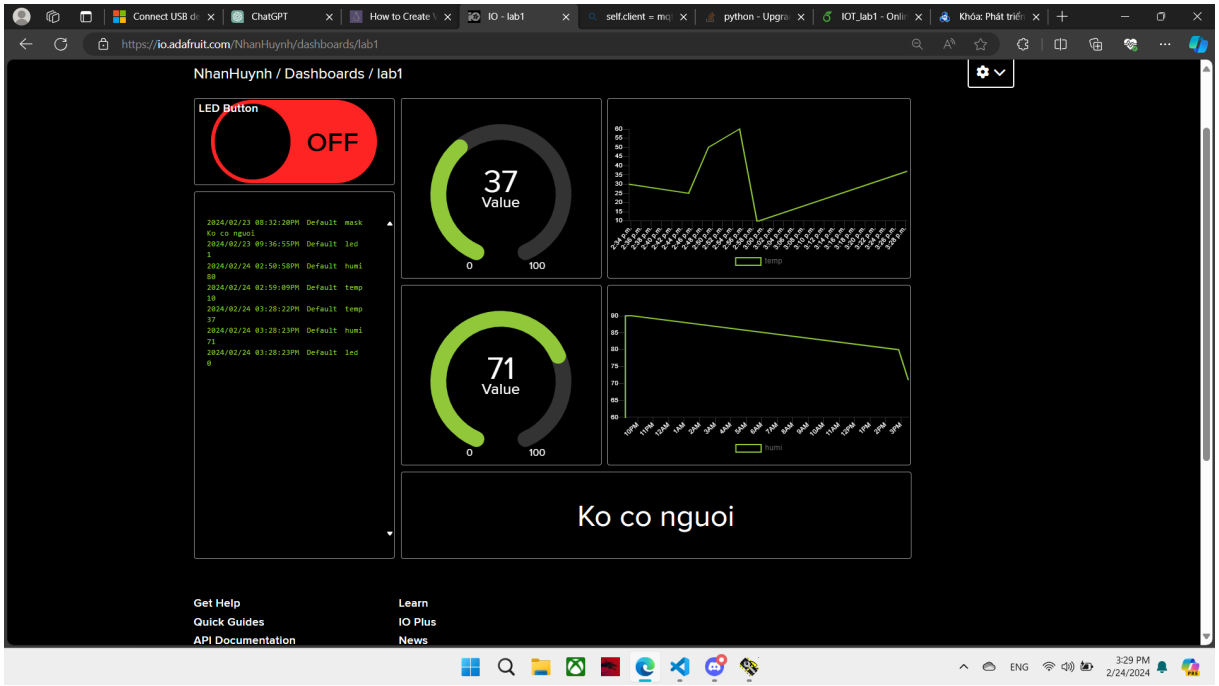
2.2.2 Kết quả

Trường hợp 1 với chuỗi "!TEMP:31#!HUMI:71#!LED:0#"

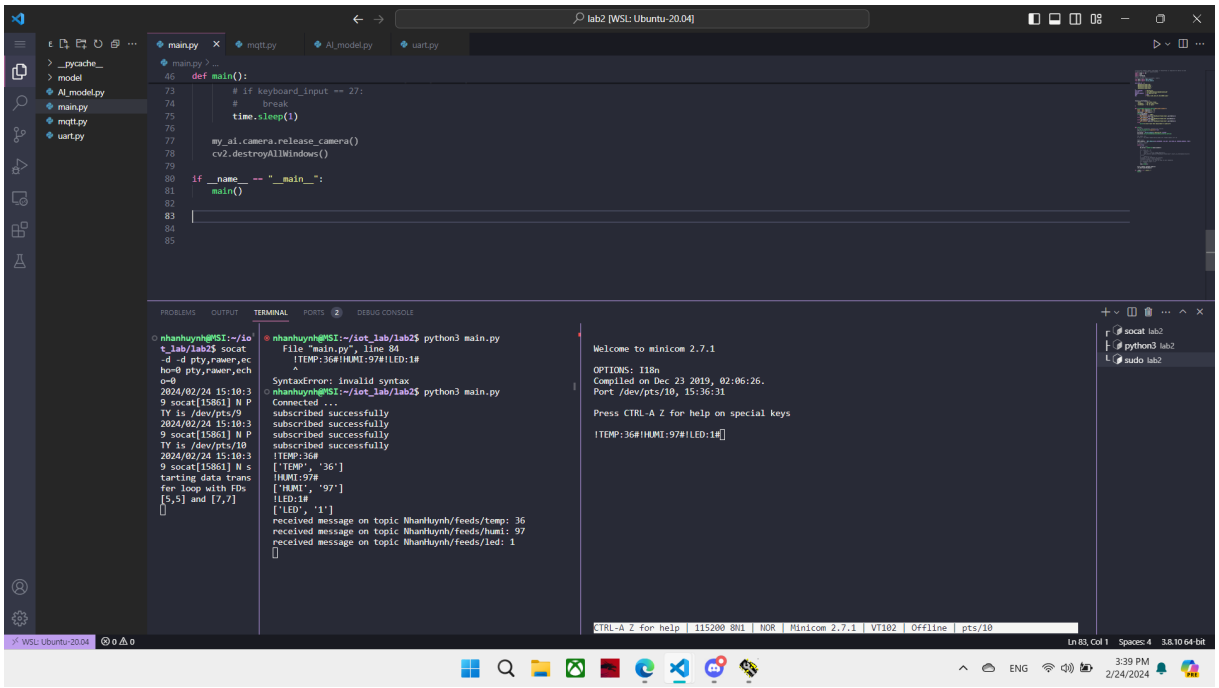


Hình 2.5: Kết quả vscode khi nhập chuỗi "!TEMP:31#!HUMI:71#!LED:0#"

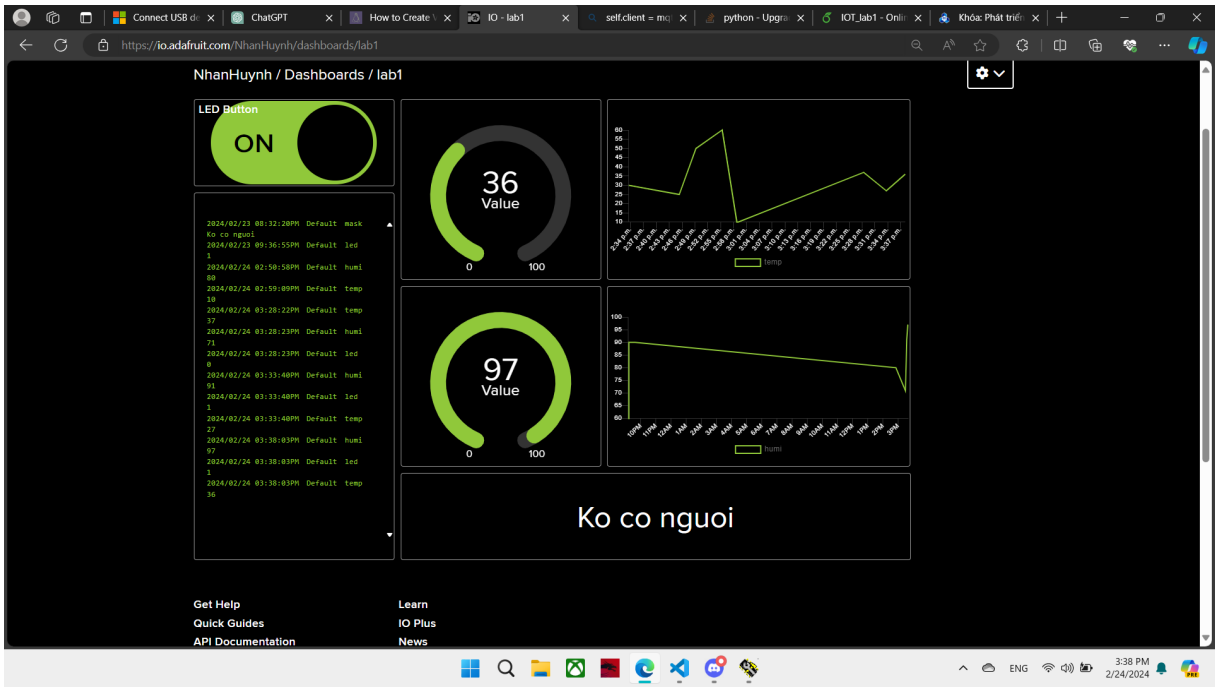
Trường hợp 2 với chuỗi !TEMP:36#!HUMI:97#!LED:1#



Hình 2.6: Kết quả dashboard trên adafruit.io khi nhập chuỗi "!TEMP:31#!HUMI:71#!LED:0#"



Hình 2.7: Kết quả vscode khi nhập chuỗi "!TEMP:36#!HUMI:97#!LED:1#"



Hình 2.8: Kết quả dashboard trên adafruit.io khi nhập chuỗi "!TEMP:36#!HUMI:97#!LED:1#"

3 Link Github source code

https://github.com/HuynhNhan0104/IOT_LAB/tree/main/lab2