

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



PHÁT TRIỂN ỨNG DỤNG INTERNETHINGS (TN) (CO3038)

Lab 02

Tích hợp Google Teachable Machine

Đọc dữ liệu từ cảm biến

Advisor: Vũ Trọng Thiên
Students: Võ Văn Dũng - 2110102

HO CHI MINH CITY, FEBRUARY 2024

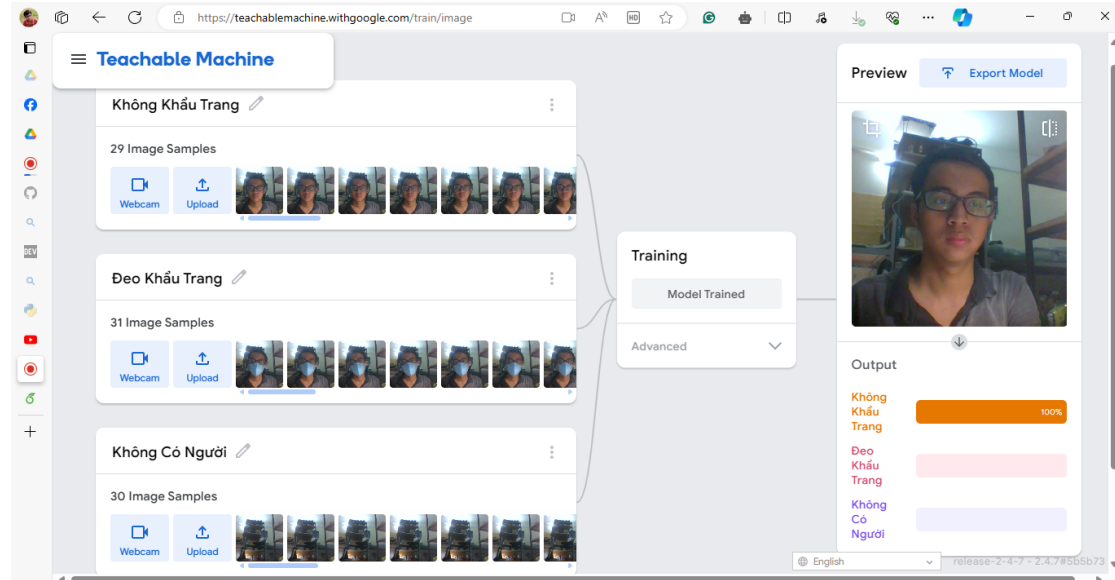


Contents

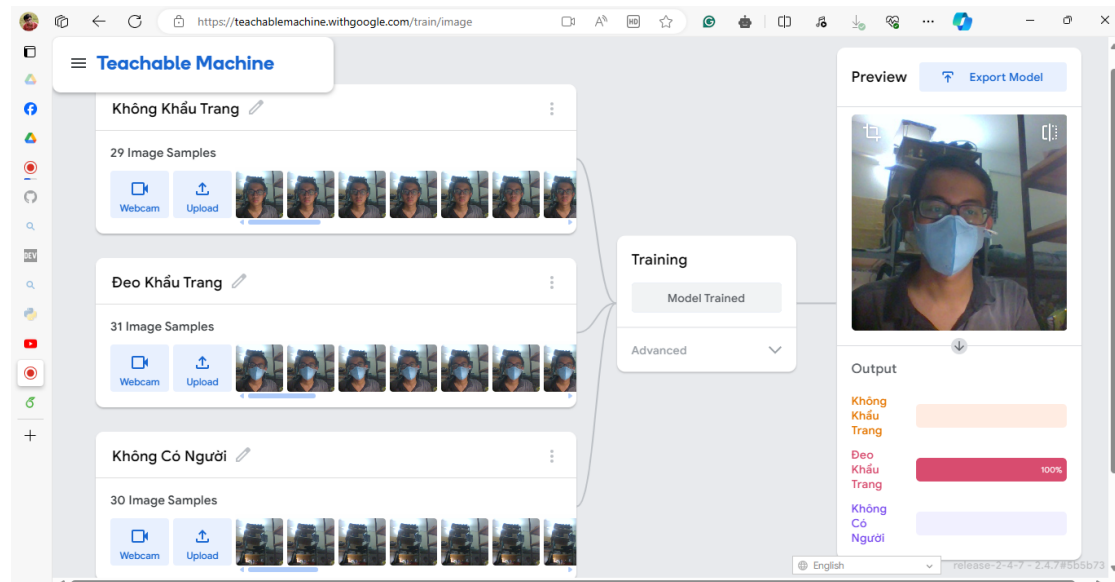
1	Tích hợp Google Teachable Machine	2
1.1	Add Image Samples, Train Model, Export Model	2
1.2	Hiện thực trên Windows với VSCode	3
1.3	Hiện thực trên Linux với môi trường WSL tích hợp trong VS Code	4
1.4	Kết nối với Adafruit IO	5
2	Đọc dữ liệu từ cảm biến	7
2.1	Cài đặt phần mềm giả lập	7
2.2	Hiện thực trên Windows với VSCode - Gửi và nhận dữ liệu	7
2.3	Hiện thực trên Linux với môi trường WSL tích hợp trong VS Code - Gửi và nhận dữ liệu	8
2.4	Kết nối với Adafruit IO	9
3	Github	11

1 Tích hợp Google Teachable Machine

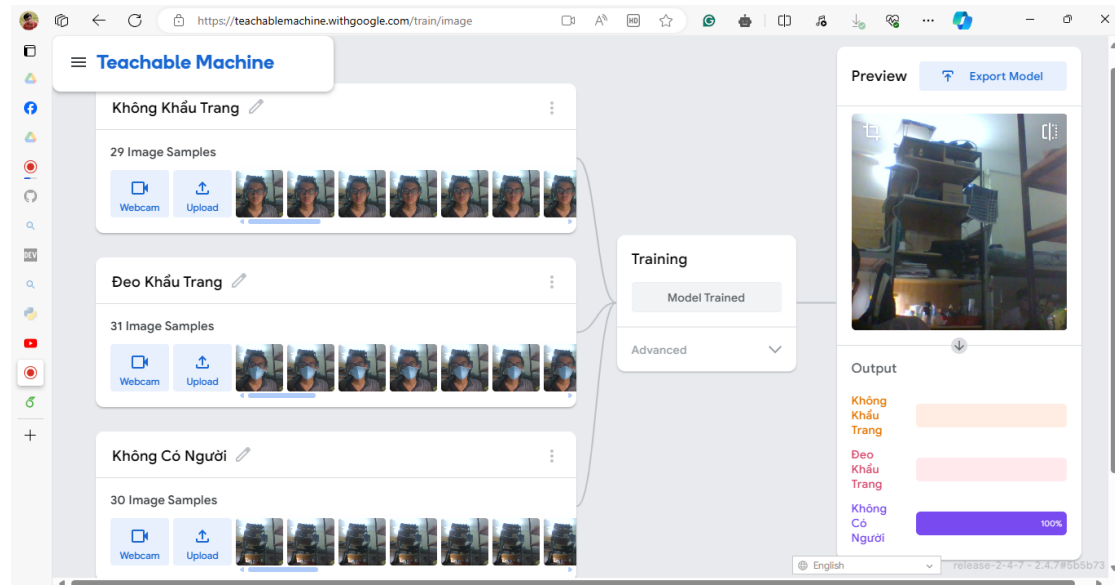
1.1 Add Image Samples, Train Model, Export Model



Hình 1: Model nhận biết khi không đeo khẩu trang



Hình 2: Model nhận biết khi đeo khẩu trang

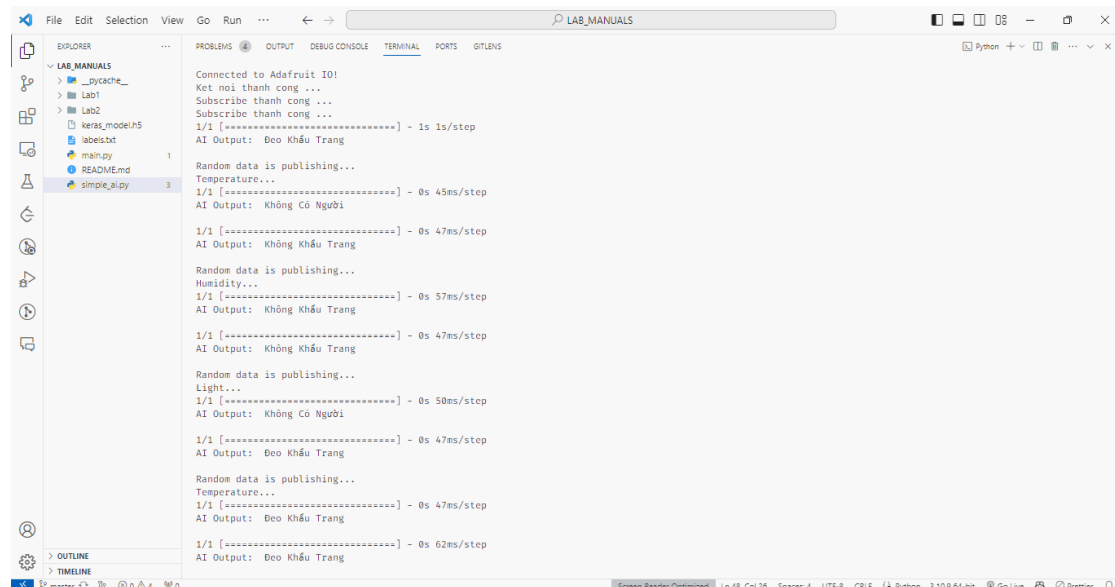


Hình 3: Model nhận biết khi không có người

1.2 Hiện thực trên Windows với VSCode

Với Windows, trong hàm VideoCapture(), với các tham số truyền vào 0, phần mềm truy cập camera của máy tính để lấy dữ liệu video.

```
1 camera = cv2.VideoCapture(0)
```

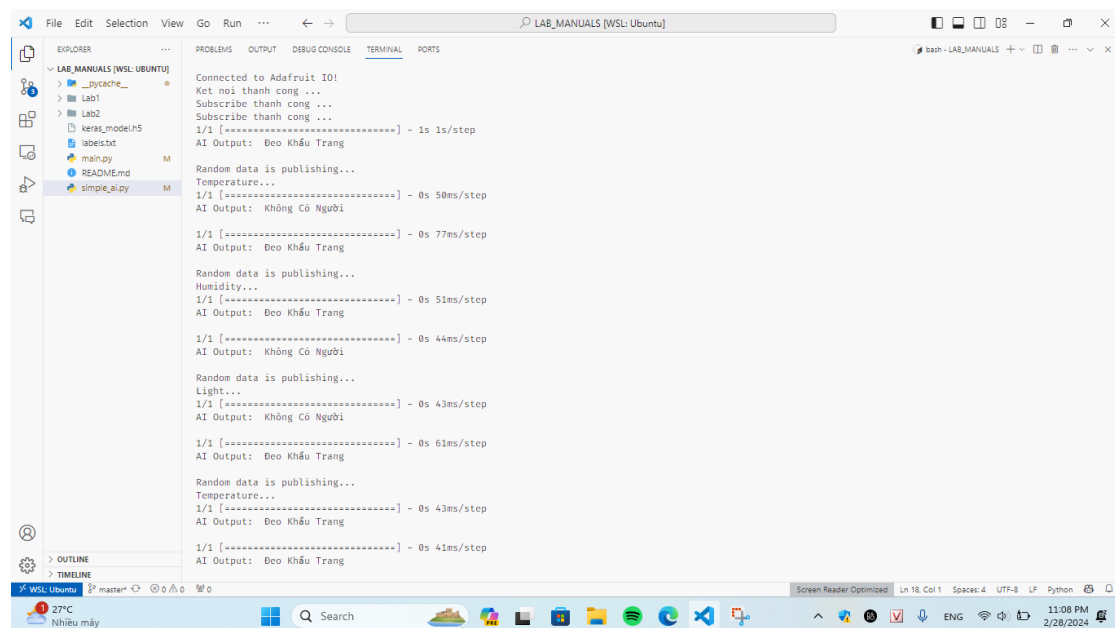


Hình 4: Implementation on Windows

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

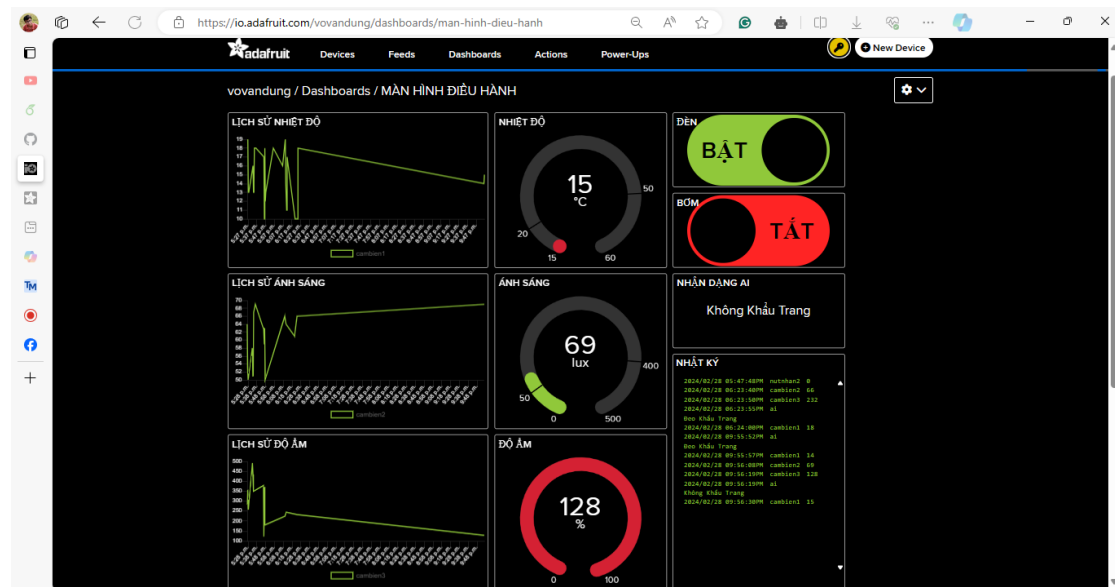
Với WSL, trong hàm VideoCapture(), với các tham số truyền vào 0 hoặc 1, phần mềm không thể truy cập camera của máy tính để lấy dữ liệu video. Do đó, em dùng phần mềm DroidCam để giả lập camera của điện thoại như là camera máy tính để thực hiện với đoạn code xử lý video từ camera điện thoại bên dưới.

```
1 ip_address = "192.168.175.55"
2 port = 4747
3
4 # Create a VideoCapture object
5 camera = cv2.VideoCapture(f"http://{ip_address}:{port}/video")
```

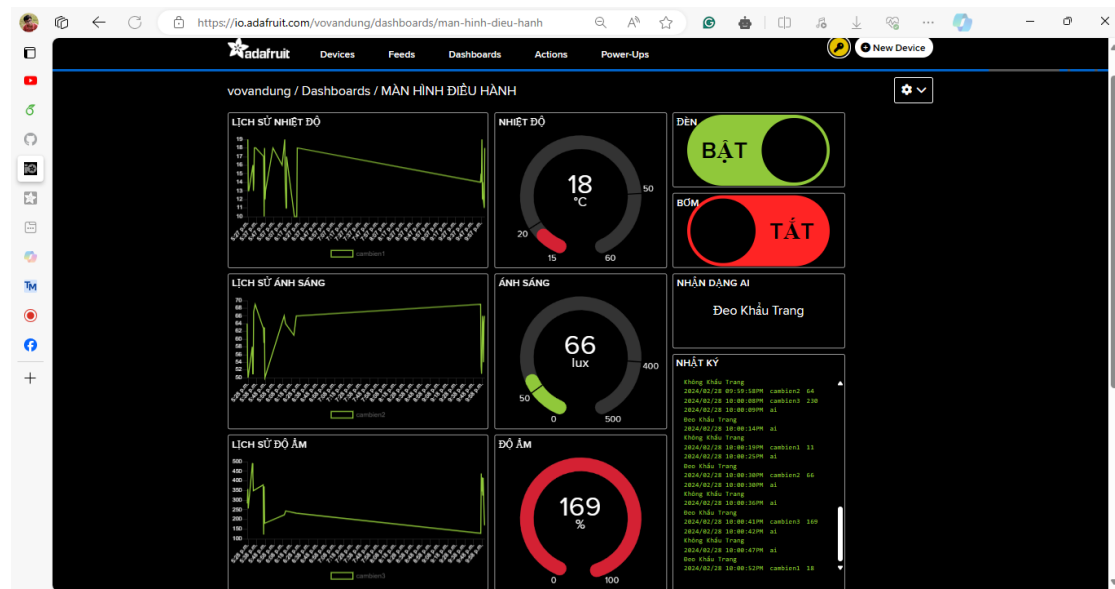


Hình 5: Implementation on Linux

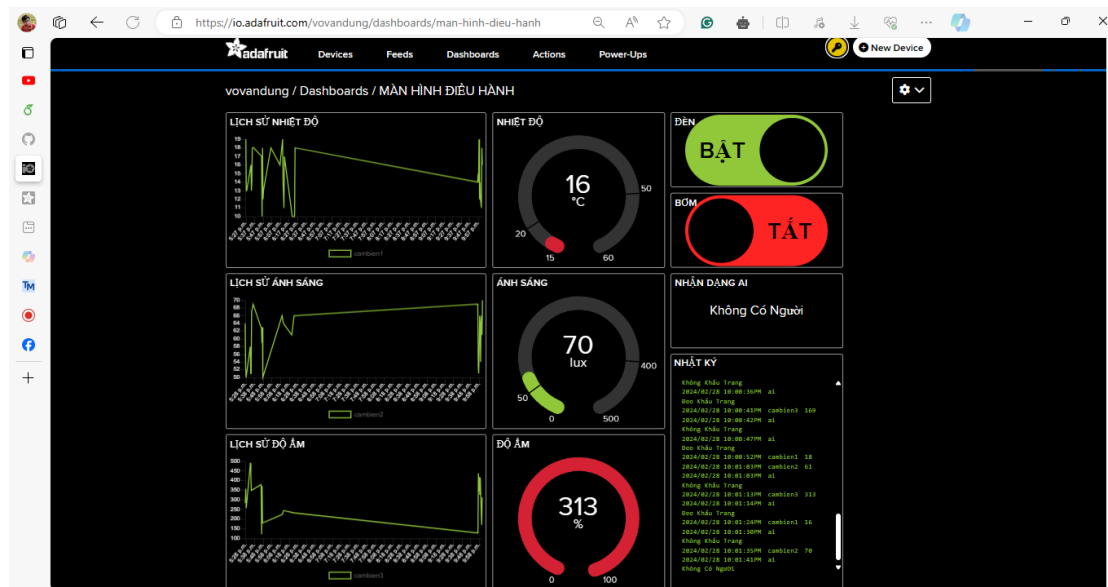
1.4 Kết nối với Adafruit IO



Hình 6: Kết nối với Adafruit khi không đeo khẩu trang



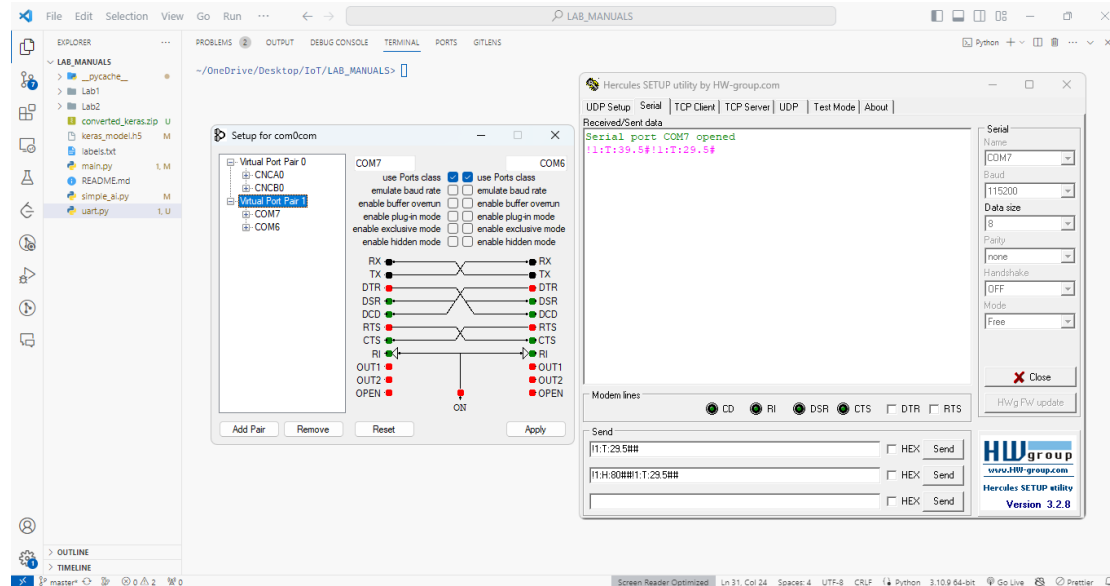
Hình 7: Kết nối với Adafruit khi đeo khẩu trang



Hình 8: Kết nối với Adafruit khi không có người

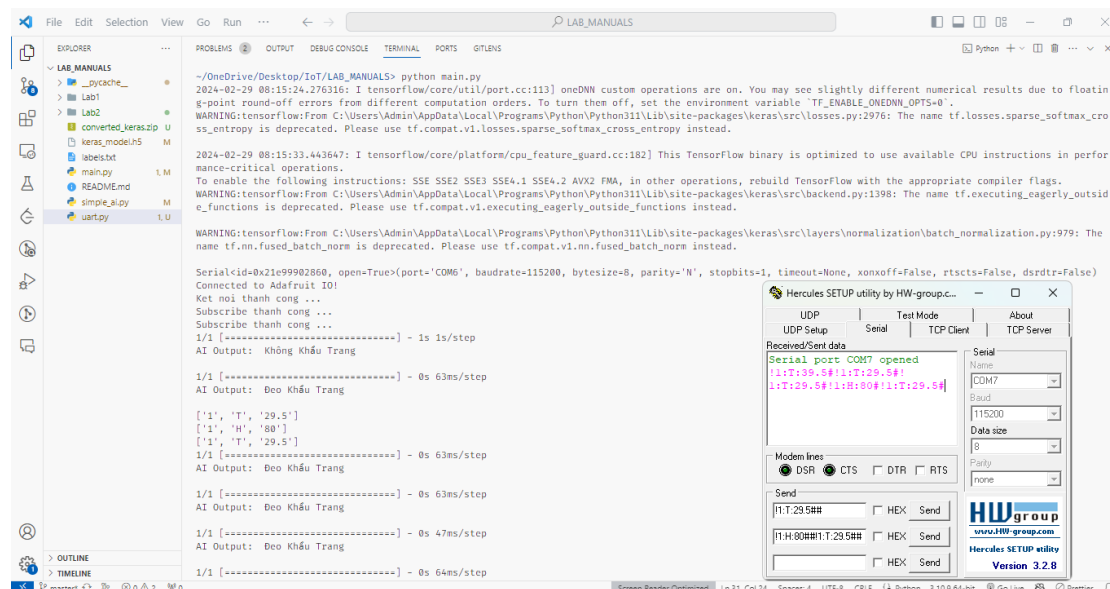
2 Đọc dữ liệu từ cảm biến

2.1 Cài đặt phần mềm giả lập



Hình 9: Phần mềm giả lập

2.2 Hiện thực trên Windows với VSCode - Gửi và nhận dữ liệu

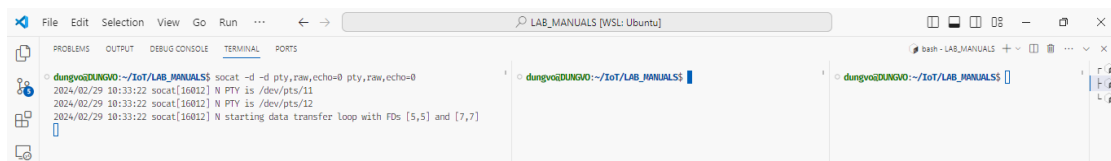


Hình 10: Gửi và nhận dữ liệu trên Windows

2.3 Hiện thực trên Linux với môi trường WSL tích hợp trong VS Code - Gửi và nhận dữ liệu

Vì WSL là môi trường ảo, nên ta phải xử lý WSL nối ra USB ảo, port ảo. Do đó, ta giả lập hai cổng ảo (virtual serial ports) có thể giao tiếp với nhau bằng cách sử dụng câu lệnh socat. Tương tự với việc đọc dữ liệu từ cảm biến.

- Bước 1: Cài đặt socat (nếu chưa có) bằng câu lệnh `sudo apt-get install -y socat`.
- Bước 2: Tạo ra hai cổng ảo có thể giao tiếp với nhau, dùng câu lệnh `"socat -d -d pty, raw, echo=0 pty, raw, echo=0"` trong terminal. Một cổng được dùng bởi chương trình `"main.py"` để đọc dữ liệu, một cổng được dùng để gửi dữ liệu.

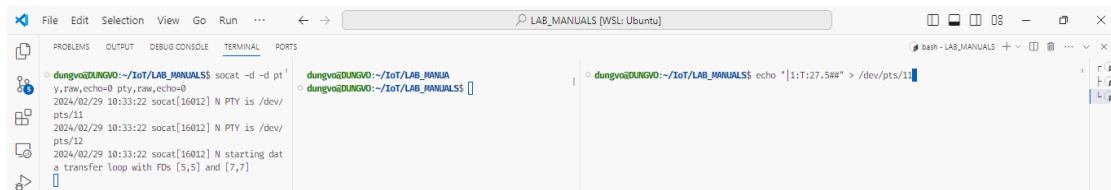


Hình 11: Cổng ảo có thể giao tiếp

- Bước 3: Thay đổi hàm `getPort()` để có thể giao tiếp qua cổng ảo vừa được tạo.

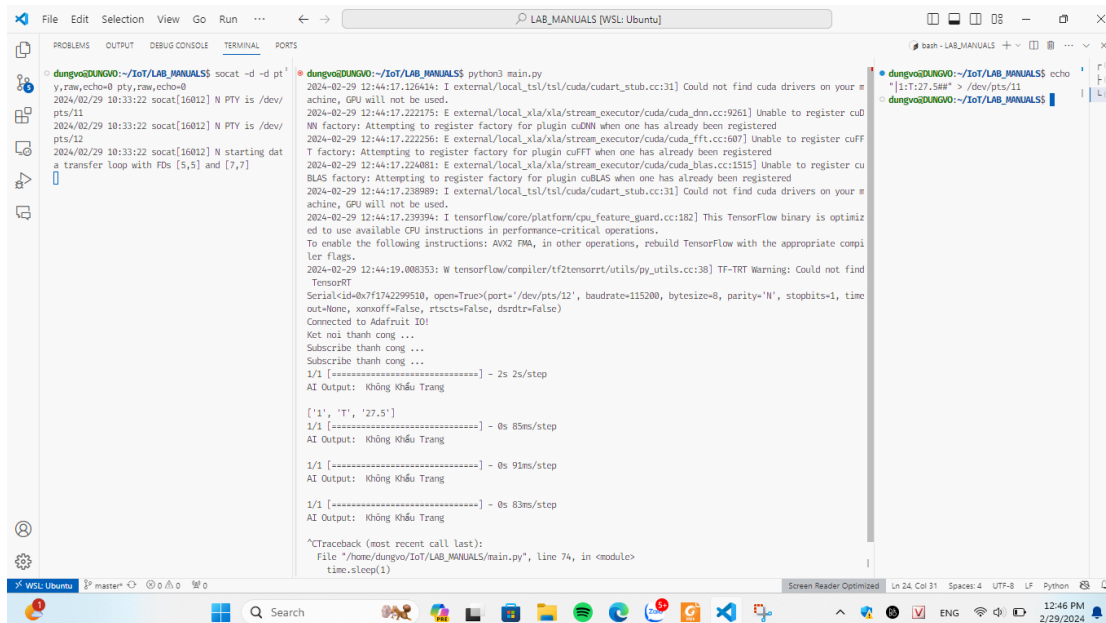
```
1 def getPort():
2     ports = serial.tools.list_ports.comports()
3     N = len(ports)
4     commPort = "None"
5     for i in range(0, N):
6         port = ports[i]
7         strPort = str(port)
8         if "USB Serial Device" in strPort:
9             splitPort = strPort.split(" ")
10            commPort = (splitPort[0])
11     return "/dev/pts/12"
```

- Bước 4: Gửi dữ liệu từ terminal qua cổng ảo bằng lệnh `"echo "|1:T:27.5##" > /dev/pts/11"`



Hình 12: Gửi dữ liệu qua cổng ảo

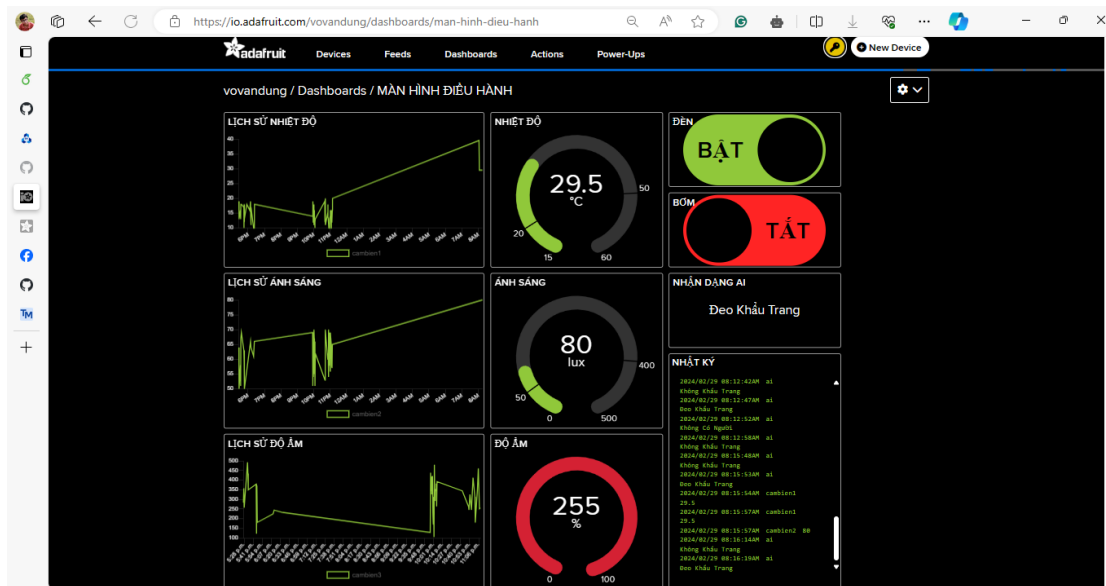
Thực hiện gửi dữ liệu nhiệt độ 27.5 bằng cổng ảo, và chương trình main đã nhận được và in ra màn hình.



Hình 13: Gửi và nhận dữ liệu trên WSL

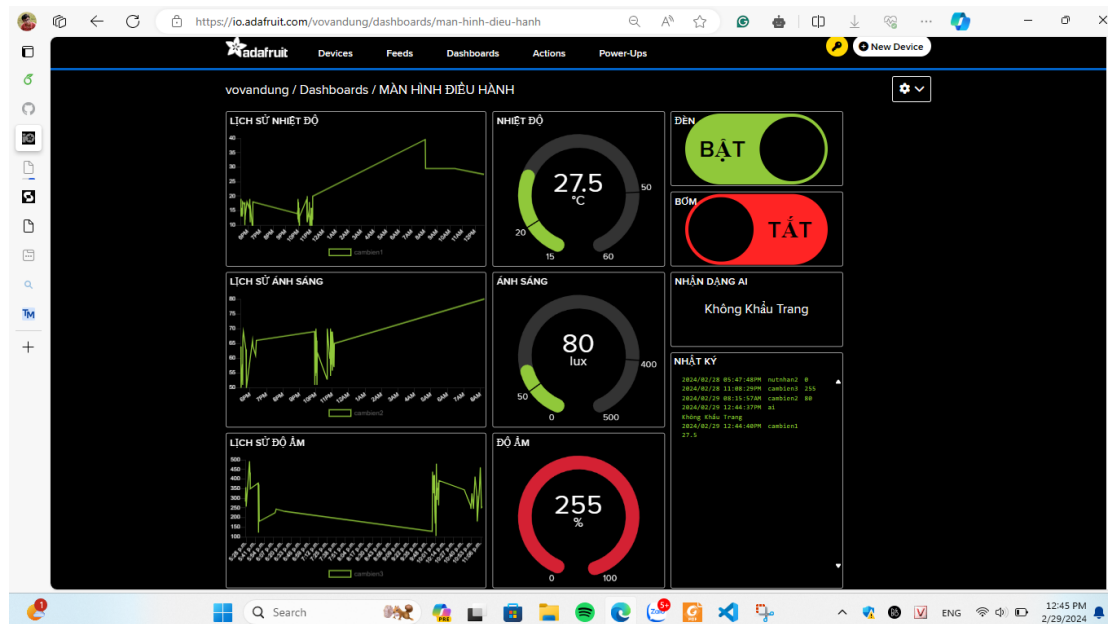
2.4 Kết nối với Adafruit IO

Gửi dữ liệu nhiệt độ và ánh sáng từ hai cảm biến "cambien1" và "cambien2" lên Adafruit IO trên môi trường Windows.



Hình 14: Gửi dữ liệu từ hai cảm biến lên Adafruit IO với môi trường Windows

Gửi dữ liệu nhiệt độ từ cảm biến "cambien1" lên Adafruit IO trên Linux với môi trường WSL.



Hình 15: Gửi dữ liệu từ cảm biến nhiệt độ lên Adafruit IO với môi trường WSL

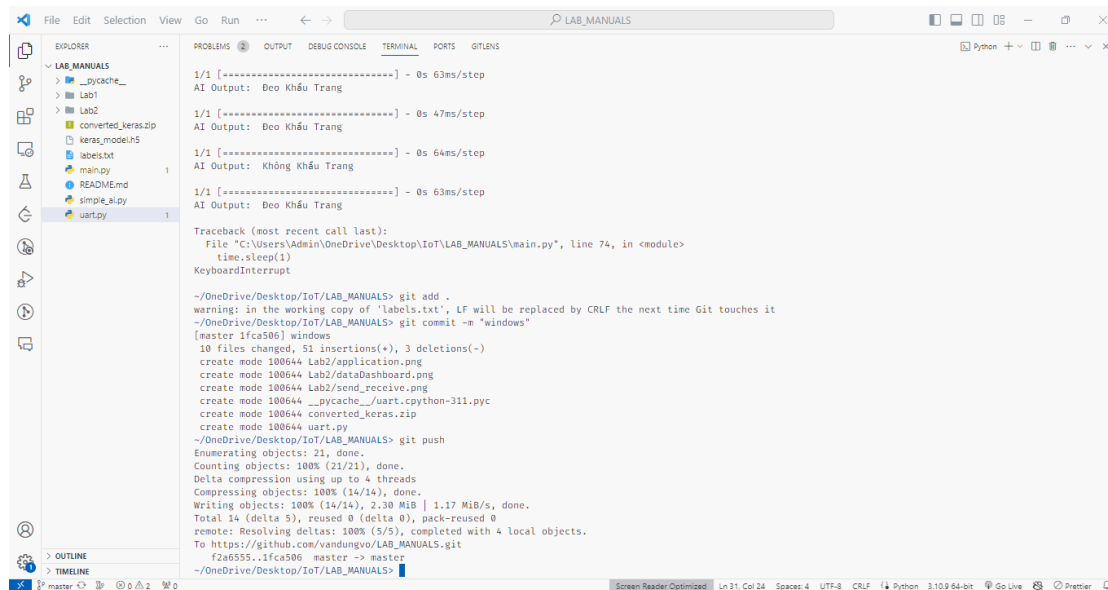
3 Github

Git source: https://github.com/vandungvo/LAB_MANUALS

Mã nguồn trong file main.py và hình ảnh trong folder Lab2.

Thầy có thể ấn vào tên hình để xem hình rõ hơn.

Khi upload code lên Github, em dùng các câu lệnh như git add ., git commit -m "message", git push.



```
1/1 [=====] - 0s 63ms/step
AI Output: Đeo Khẩu Trang

1/1 [=====] - 0s 47ms/step
AI Output: Đeo Khẩu Trang

1/1 [=====] - 0s 64ms/step
AI Output: Không Khẩu Trang

1/1 [=====] - 0s 63ms/step
AI Output: Đeo Khẩu Trang

Traceback (most recent call last):
  File "C:\Users\Admin\OneDrive\Desktop\IoT\LAB_MANUALS\main.py", line 74, in <module>
    time.sleep(1)
KeyboardInterrupt

~/OneDrive/Desktop/IoT/LAB_MANUALS> git add .
warning: in the working copy of 'labels.txt', LF will be replaced by CRLF the next time Git touches it
~/OneDrive/Desktop/IoT/LAB_MANUALS> git commit -m "windows"
[master 1fca506] windows
10 files changed, 51 insertions(+), 3 deletions(-)
create mode 100644 Lab2/application.png
create mode 100644 Lab2/dataDashboard.png
create mode 100644 Lab2/send_receive.png
create mode 100644 __pycache__/uart.cpython-311.pyc
create mode 100644 converted_keras.zip
create mode 100644 uart.py
~/OneDrive/Desktop/IoT/LAB_MANUALS> git push
Enumerating objects: 21, done.
Counting objects: 100% (21/21), done.
Delta compression using up to 4 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (14/14), 2.30 MiB | 1.17 MiB/s, done.
Total 14 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), completed with 4 local objects.
To https://github.com/vandungvo/LAB_MANUALS.git
 f2a6555..1fca506 master -> master
~/OneDrive/Desktop/IoT/LAB_MANUALS>
```

Hình 16: Quá trình sử dụng git