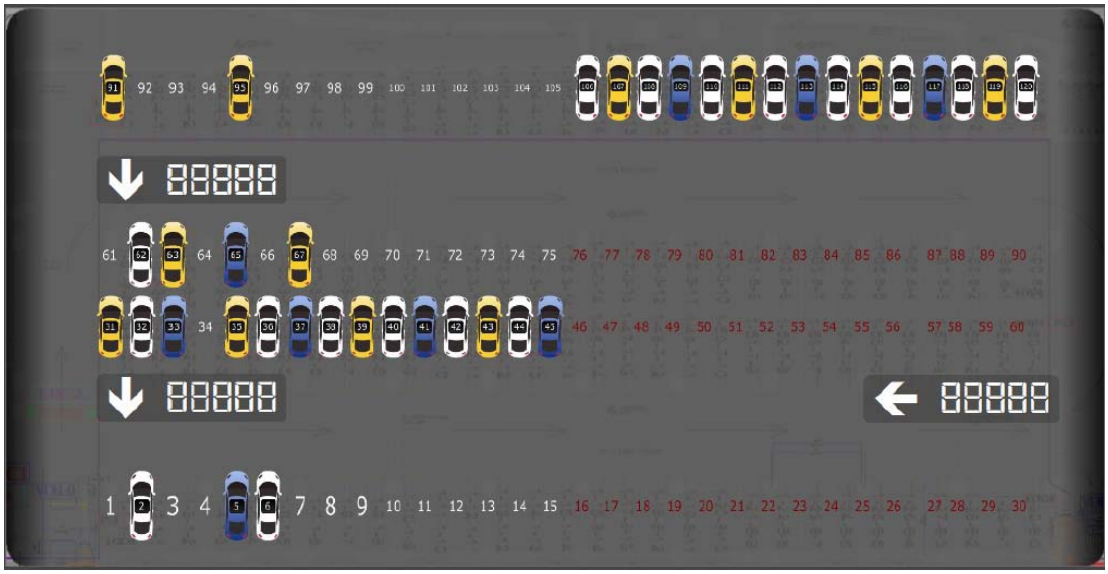
THIẾT BỊ PHÁT HIỆN CHỖ ĐỖ XE THÔNG MINH

1. Hệ thống Smart Parking

Hệ thống bãi đỗ xe thông minh bao gồm nhiều cụm chức năng. Trong tài liệu này chỉ tập trung mô tả cụm thiết bị IoT phát hiện có hay không có xe đỗ tại các ô quy định trong bản đồ bãi đỗ xe.

Hình 1 mô tả một giải pháp hệ thống đỗ xe thông minh trong hầm. Trạng thái bãi đỗ xe được mô tả có hoặc không có tại những ô đỗ được đánh số (địa chỉ ID) theo quy định. Người dùng có thể theo dõi tình trạng bãi đỗ xe thông qua một ứng dụng được cài đặt trên máy tính quản lý và (hoặc ứng dụng web có thể truy nhập trên các thiết bị thông minh cầm tay).



Hình 1. Bản đồ bãi đỗ xe ô tô trong nhà

Hình 2 mô tả sơ đồ hệ thống kết nối thiết bị phát hiện chỗ để xe. Tại mỗi một ô để xe, một thiết bị cảm biến thông minh phát hiện có hay không có xe đang đỗ, được treo phía bên trên trần khu vực bãi đỗ. Khi có xe, đèn đỏ trên thiết bị được bật sáng. Khi không có xe, đèn xanh bật sáng.

Thông tin được gửi về máy tính trung tâm từ một thiết bị IoT sẽ mang hai trường dữ liệu:

+ ID của thiết bị (**ID**)

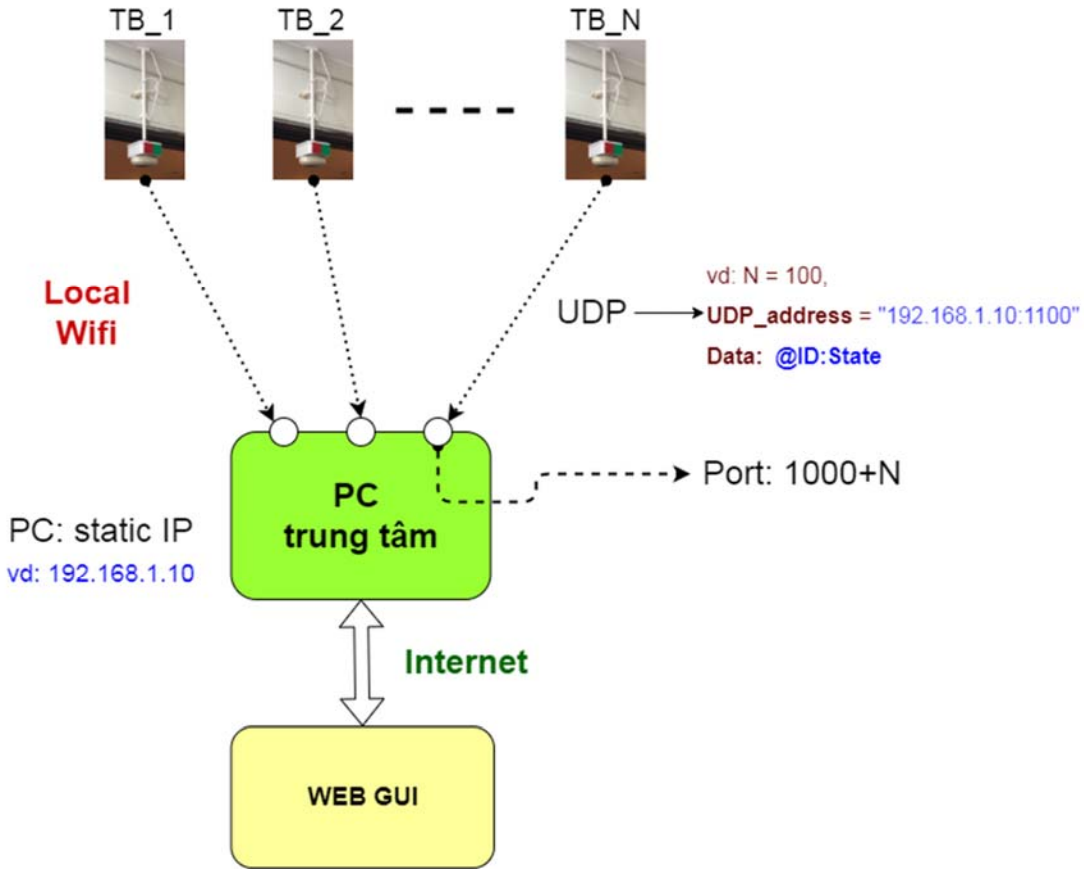
+ Trạng thái có hay không có xe (**State**)

Khung dữ liệu được truyền về có định dạng kiểu String:

"@ID:State"

+ State = 1: có xe, State = 0: không có xe

Ví dụ, ô để xe số 95 đang có xe sẽ truyền về dữ liệu: "@95:1"



Hình 2. Sơ đồ hệ thống phát hiện chỗ để xe thông minh

Giao thức UDP được sử dụng để truyền nhận dữ liệu giữa máy tính trung tâm và các thiết bị IoT. Hệ thống kết nối và cấu hình mạng được quy định như sau:

+ PC trung tâm sẽ được gán một **IP tĩnh cố định**. Ví dụ: "192.168.1.10"

+ Ứng dụng quản lý bãi đỗ xe sẽ có một mô đun chức năng thu nhận thông tin gửi về từ tất cả các thiết bị IoT. Khi một thiết bị IoT được thêm vào hệ thống tương ứng với một vị trí để xe trong bãi đỗ, thiết bị này sẽ được gán một ID trùng với số thứ tự của vị trí để xe được quy định trên bãi đỗ, ví dụ ID = 55.

Khi đó, kênh kết nối UDP được mở để giao tiếp giữa PC trung tâm và thiết bị IoT. Để dễ quản lý, cổng kết nối được hình thành theo quy luật tự động, ví dụ:

Port = 1000 + ID

Trong trường hợp thiết bị IoT có ID = 55, thì sẽ được gán với cổng UDP 1055.

+ Địa chỉ định nghĩa kết nối UDP sẽ là: "Host_Static_IP:Port". Trong ví dụ này, địa chỉ và cổng để truyền nhận thông tin là "192.168.1.10:1055"

Như vậy mỗi một thiết bị IoT khi được bổ sung vào hệ thống sẽ được cấu hình với một số thông tin cố định sau:

- **Thông tin về mạng wifi cục bộ**: **SSID, password**
- **ID của thiết bị IoT**
- **Địa chỉ và cổng được cấp tương ứng trên PC trung tâm**

2. Demo truyền nhận dữ liệu

Dưới đây là demo giả lập hệ thống IoT, thu nhận dữ liệu từ các thiết bị IoT. Số lượng thiết bị được kiểm soát bằng biến N. Có thể tùy ý thay đổi giá trị của N để kiểm tra tính ổn định của phần mềm quản lý chung. Trong 2 demo, N = 100.

a) Tạo tín hiệu IoT giả lập: demo_fake_IoT.py

```
#! /usr/bin/env python
from PyQt5.QtWidgets import *
from PyQt5.QtCore import QThread
import socket, sys
from time import sleep
import random
class SendUDP(QThread):
    def __init__(self, ID,remote_address,parent = None):
        QThread.__init__(self,parent)
        # Lấy thông tin về ID của IoT
        self.ID = ID
        # Gán trạng thái mặc định cho IoT
```

```
        #   state = 0   : không có xe;   state = 1   : có xe
        self.state = 0
        # Remote_address: địa chỉ IP + cổng UDP tương ứng trên máy chủ
        # Ví dụ IP của máy chủ là 192.168.1.10, cổng UDP là 1001,
        # thì Remote_address = "192.168.1.10:1001"
        self.remote_address = remote_address
        # Thiết lập kết nối UDP
        self.send_socket = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
        # Lấy thông tin IP và Port để tạo ra địa chỉ UDP chuẩn
        address = self.remote_address.split(':')
        self.remote_UDP_address = (address[0],int(address[1]))
        self.parent = parent
    """
    Gửi thông tin phản hồi về State của IoT lên PC trung tâm, giả lập với hàm random
    """
    def run(self):   # Implement own function
        while True:
            # Tạo dữ liệu giả lập trạng thái chỗ để xe
            if random.randint(0,10)>6:
                self.state = 1-self.state
            # Đóng gói msg_feedback: "@ID:State"
            msg = "@{0}:{1}"
            msg_feedback = msg.format(self.ID,self.state)
            # Gửi dữ liệu đi
            self.send_socket.sendto(msg_feedback.encode(),self.remote_UDP_address)
            sleep(1)    # Time delay between 2 consecutive sends
if __name__ == "__main__":
    app = QApplication(sys.argv)
    # Tạo các thread giả lập thiết bị IoT với từng ID riêng
    # Chú ý, gán địa chỉ cổng cần giống với ID của IoT cho dễ quản lý,
    Host_IP = "192.168.1.10"
    N = 100
    List_IoT = []
    # Tạo ra N thiết bị IoT ảo
    for k in range(N):
        address = '{0}:{1}'
        ID = k+1
        Port = 1000+ID
        udp_address = address.format(Host_IP,Port)
        IoT = SendUDP(k+1,udp_address)
        List_IoT.append(IoT)
    # Chạy các thiết bị IoT ảo
    for k in range(N):
        List_IoT[k].start()
    sys.exit(app.exec_())
```

b) Mô đun thu thập dữ liệu IoT: demo_UDP_receivers.py

```
#! /usr/bin/env python
from PyQt5.QtWidgets import *
from PyQt5.QtCore import QThread
import socket, sys

class ReceiveUDP(QThread):

    def __init__(self, local_address, parent = None):
        QThread.__init__(self,parent)
        # Thiết lập kết nối UDP trên máy chủ
        # local_address = "HostIP:Port" ,
        # Ví dụ: local_address = "192.168.1.10:1003" ,
        # + địa chỉ IP của máy chủ trong mạng cục bộ là static IP = 192.168.1.10
        # + cổng UDP được mở cho IoT với ID số 3: Port = 1003
        self.local_address = local_address
        self.local_socket = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
        address = self.local_address.split(':')
        self.local_socket.bind((address[0],int(address[1])))

    def run(self):
        cnt = 0
        while True:
            msg_master, add = self.local_socket.recvfrom(1024)
            if msg_master:
                    # Chuyển chuỗi ký tự nhận được sang dạng str
                    msg = msg_master.decode('utf-8')
                    # Tách trường dữ liệu @ID:state, lấy ID và state
                    msg_decode = msg.replace('@','').split(':')
                    ID = int(msg_decode[0])
                    state = int(msg_decode[1])
                    # In kết quả nhận được
                    # ID và state sẽ được sử dụng để hiển thị trạng thái trên bản đồ
bố trí chỗ để xe
                    cnt = cnt+1
                    print(cnt,':',ID,':',state)
if __name__ == "__main__":
    app = QApplication(sys.argv)
    # Tạo các thread nhận dữ liệu phản hồi từ các thiết bị IoT
    N = 100
    Host_IP = "192.168.1.10"
    List_IoT = []
    # Tao ra các thread quản lý N thiết bị IoT
    for k in range(N):
```

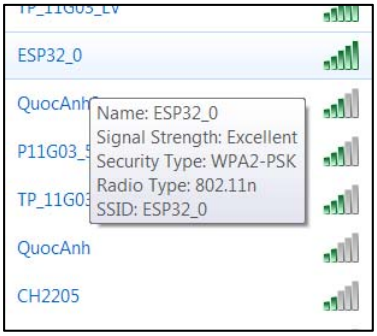
```
        address = '{0}:{1}'
        Port = 1001+k
        udp_address = address.format(Host_IP,Port)
        IoT = ReceiveUDP(udp_address)
        List_IoT.append(IoT)
    # Chạy N threads quản lý các TB IoT
    for k in range(N):
        List_IoT[k].start()

    sys.exit(app.exec_())
```

3. Hướng dẫn cấu hình thiết bị IoT

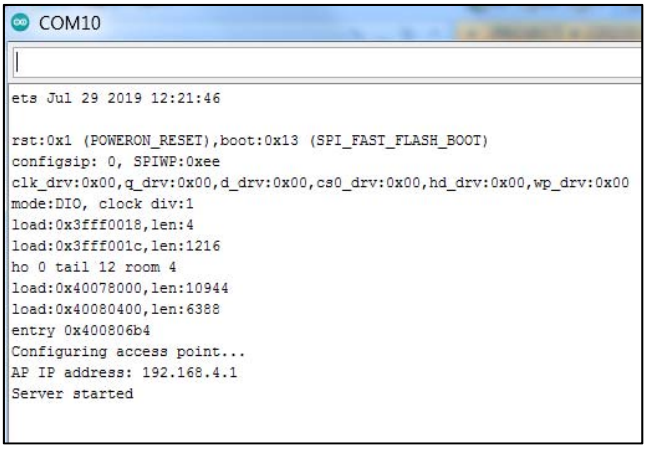
a) Thiết bị mới hoàn toàn

Đối với thiết bị mới hoàn toàn, khi cấp nguồn cho thiết bị, sẽ xuất hiện một mạng Wifi có tên là "**ESP32_0**" , tương ứng với ID hiện tại của thiết bị là 0.

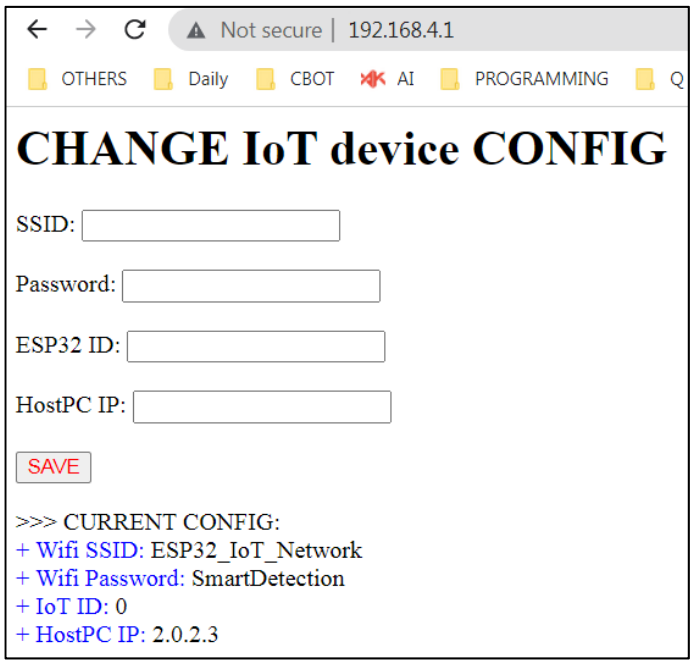


Có thể sử dụng điện thoại thông minh hoặc máy tính để truy nhập vào mạng "ESP32_0", mật khẩu mặc định là "**esp32iot**".

Truy nhập vào trang web cấu hình của thiết bị tại địa chỉ "**192.168.4.1**". Địa chỉ này là cố định đối với tất cả các thiết bị IoT. Có thể kiểm tra thông tin này khi kết nối ESP32 với PC thông qua cổng USB, sử dụng một Serial Terminal bất kỳ:



Trên cửa sổ trình duyệt tại địa chỉ "**192.168.4.1**" sẽ hiện ra giao diện cấu hình bao gồm những thông tin sau:



+ SSID: nhập thông tin về SSID của mạng WIFI cục bộ của người dùng

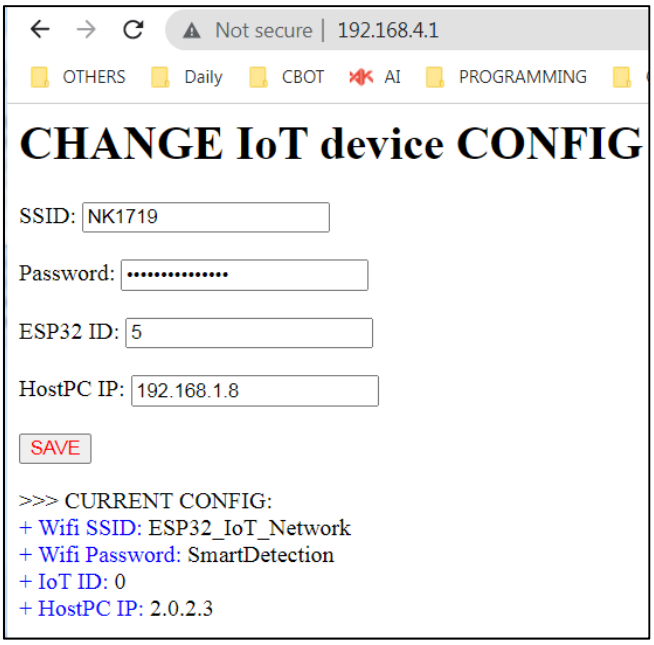
+ Password: mật khẩu của mạng WIFI cục bộ

+ ESP32 ID: nhập số ID của thiết bị IoT, số ID này nhận các giá trị số nguyên dương từ 1,2,... Địa chỉ của cổng giao tiếp UDP sẽ được thiết lập tự động theo công thức UDP_Port = 1000 + ID.

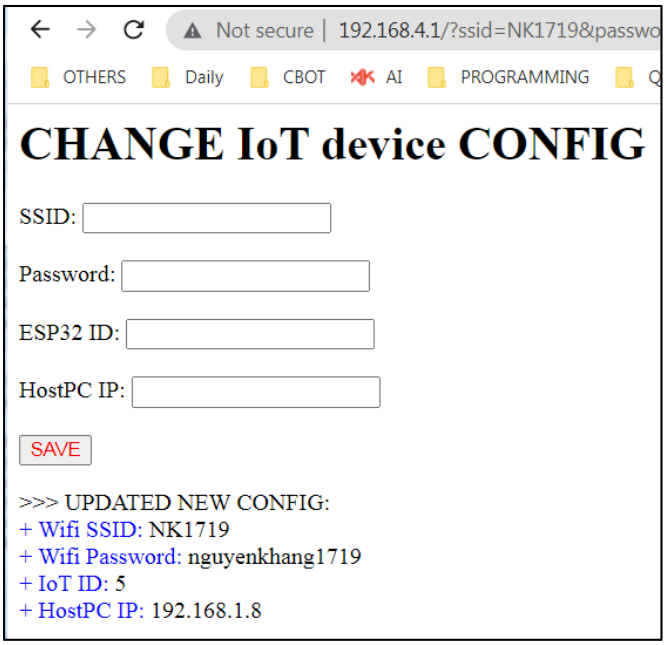
+ HostPC IP: nhập địa chỉ IP của máy tính người dùng trong mạng WIFI cục bộ. Địa chỉ này nên được đặt là static. Nếu bị thay đổi, cần phải thay đổi lại cấu hình thiết bị trên trang web này.

Cấu hình mặc định của thiết bị IoT mới được hiển thị ở bên dưới "CURRENT CONFIG". Về cơ bản chỉ là một khởi tạo giả lập. Do đó người dùng cần thiết lập cấu hình mới cho thiết bị ngay từ lần sử dụng đầu tiên.

Ví dụ: bổ sung thông tin mới cho thiết bị với mạng WIFI sẵn có, ID của thiết bị và địa chỉ IP của máy tính người dùng như trên hình sau:

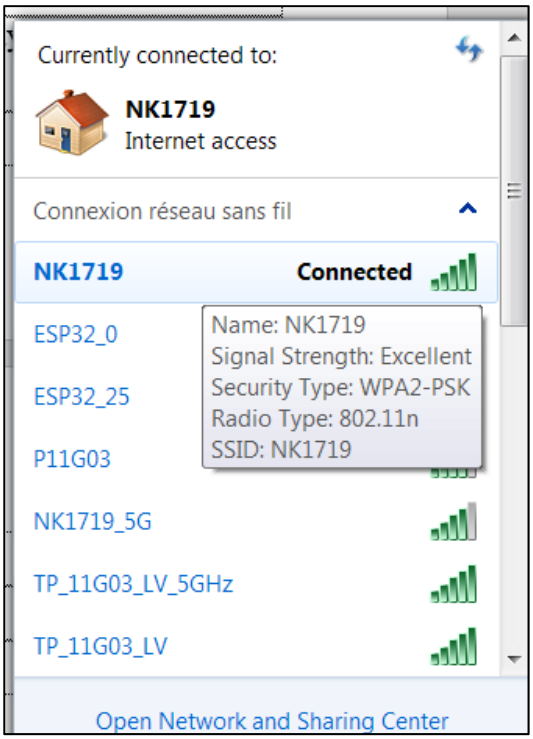


Nhấn nút "**SAVE**" để ghi cấu hình vào bộ nhớ EEPROM của thiết bị. Lúc này thông tin mới được cập nhật sẽ được hiển thị dưới dòng "UPDATED NEW CONFIG":



Thiết bị mới đã được ghi với thông tin sau: SSID, Password, ID, HostPC_IP.

Truy nhập máy tính vào mạng WIFI vừa được thiết lập, và chạy chương trình demo " **demo_UDP_receivers.py** " trên Python. Kiểm tra lại HostPC IP trong mạng wifi này xem đã đúng chưa. Nếu sai cần đặt lại cấu hình cho thiết bị IoT.



Dữ liệu thu thập về máy tính sẽ được hiển thị trên terminal. Trong hình dưới là dữ liệu thu thập từ 2 thiết bị IoT với ID = 5 và ID = 25.

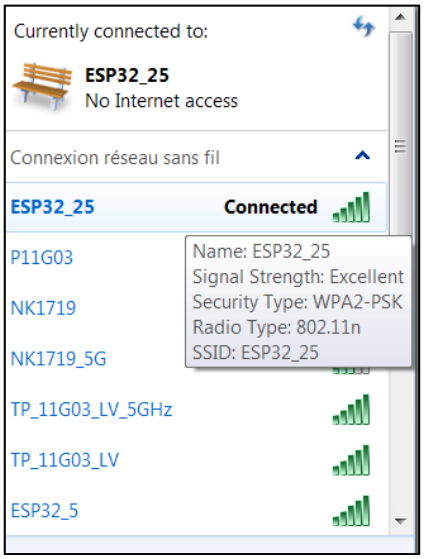
```
37  if __name__ == "__main__":
38      app = QApplication(sys.argv)
39      # Tạo các thread nhận dữ liệu phản hồi từ các thiết bị I
40      N = 100
41      Host_IP = "192.168.1.8"
42      List_IoT = []
43      # Tao ra các thread quản lý N thiết bị IoT
44      for k in range(N):
45          address = '{0}:{1}'
46          Port = 1001+k
47          udp_address = address.format(Host_IP,Port)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ··· Python

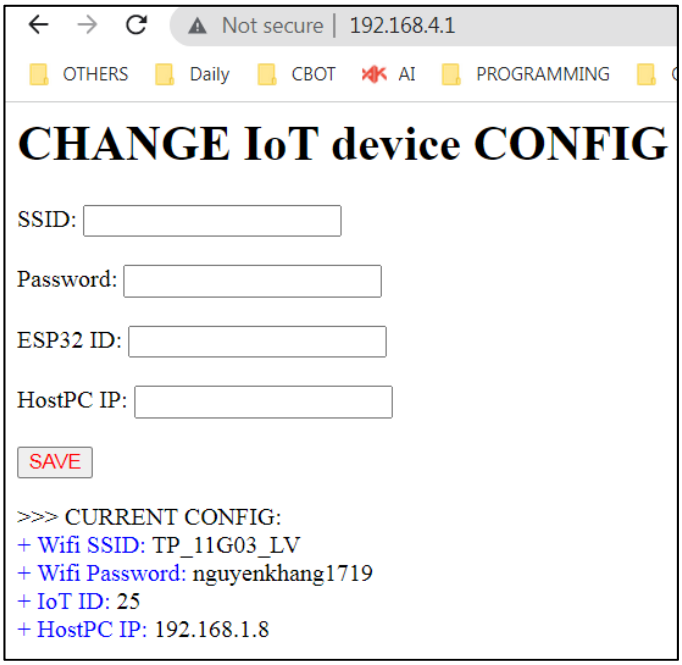
```
3016 : 25 : 1
122 : 5 : 1
3017 : 25 : 1
123 : 5 : 1
3018 : 25 : 1
124 : 5 : 1
3019 : 25 : 1
125 : 5 : 1
3020 : 25 : 1
126 : 5 : 1
3021 : 25 : 1
```

b) Thiết bị đã được ghi cấu hình

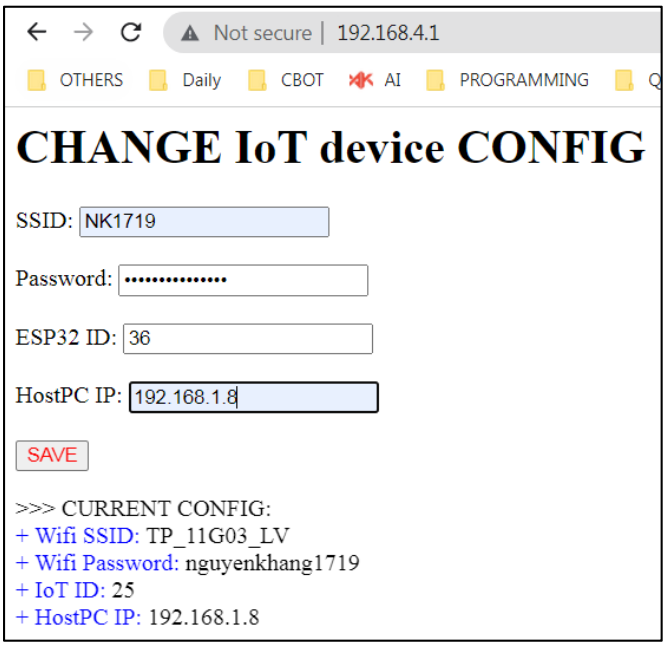
Đối với các thiết bị đã được cấu hình, khi cấp nguồn, sẽ hiển thị mạng WIFI của thiết bị IoT với SSID có định dạng "**ESP32_xxx**" , trong đó, 'xxx' chỉ địa chỉ ID hiện tại của thiết bị. Mật khẩu truy nhập vào mạng này mặc định vẫn là "**esp32iot**". Trong ảnh dưới đây là thiết bị IoT đã được ghi với ID = 25.



Để kiểm tra thông tin đầy đủ về cấu hình hiện tại, ta truy nhập vào địa chỉ "**192.168.4.1**" để mở ra cửa sổ web thiết lập cấu hình thiết bị. Các thông tin cấu hình hiện tại được hiển thị dưới "CURRENT CONFIG".

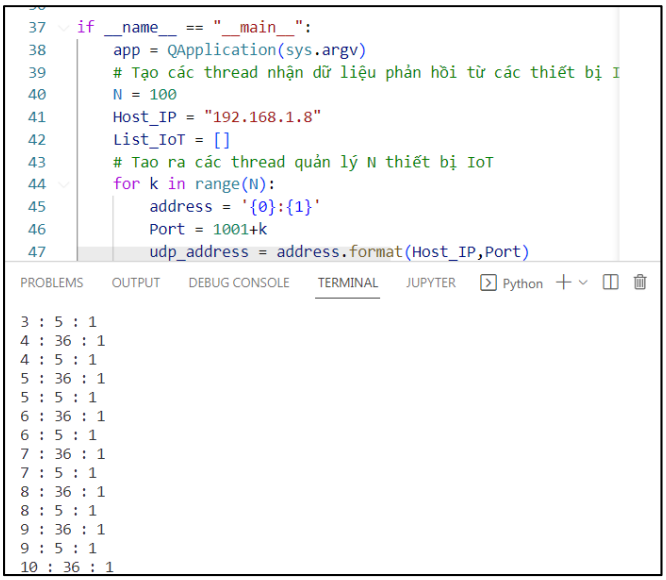


Ví dụ ta thay đổi lại cấu hình mạng và ID cho thiết bị này:



Nhấn nút "**SAVE**" để ghi cấu hình vào bộ nhớ EEPROM của thiết bị:

Kết nối máy tính người dùng vào mạng wifi hiện tại và kiểm tra terminal của demo " **demo_UDP_receivers.py** " ta nhận được:



4. Một số tính năng của thiết bị

- **Thời gian thu thập tín hiệu:** cứ mỗi 2s, thiết bị sẽ gửi trạng thái vị trí đỗ xe về máy tính trung tâm. Do ứng dụng này không cần quá nhanh, 2s là chu kỳ truyền dữ liệu vừa phải, đồng thời giúp giảm tải cho đường truyền cũng như máy tính trung tâm khi phải quản lý một mạng lớn gồm nhiều thiết bị IoT

- **Số lượng thiết bị IoT đăng ký**: số lượng không hạn chế, lưu ý công thức thiết lập cổng UDP là UDP_Port = 1000 + ID.

- **Khả năng kết nối lại vào mạng Wifi**: khi bị mất mạng wifi tạm thời, thiết bị có chế độ tự động kiểm tra kết nối và thiết lập lại kết nối khi hệ thống mạng được phục hồi.

- **Khả năng làm việc độc lập**: việc phát hiện chỗ để xe và bật đèn tín hiệu XANH / ĐỎ là độc lập với chức năng truyền dữ liệu trong mạng WIFI. Ngay cả khi bị mất hoàn toàn kết nối với mạng WIFI, thiết bị vẫn báo đèn tín hiệu bình thường khi có xe đến và xe đi.

- **Thiết lập cấu hình**: Thiết bị cho phép người sử dụng tùy ý thay đổi cấu hình mạng cũng như địa chỉ ID của thiết bị. Điều này tạo ra sự linh hoạt khi cần thay đổi thông tin hệ thống cũng như thay thế, lắp đặt…

- **Điều chỉnh khoảng cách phát hiện xe**: trên thiết bị cảm biến siêu âm có hàng nút gạt để thay đổi khoảng cách phát hiện xe ô tô, **từ 1m đến 3.5m** (8 mức điều chỉnh khoảng cách, **chỉ gạt ba vị trí 6,7,8** , **các vị trí gạt khác từ 1-5 giữ nguyên trạng thái mặc định là OFF**).