Project 7
By
Rasheeque Ahnaf Brinto
ID: 154600217
Course: CSP450

Introduction

This simple project is a demo system to collect and analyze data from 12 different IOT devices and then provide a simple diagnosis based on predefined parameters on a web server about the vehicle.

We will be using the following open tools for this project:

Hardware

1) IOT Device: ESP8266 (NodeMCU)
2) Router & Firewall: TP Link WR902AC
3) Control Unit: Raspberry Pi 4 Model B

Software

1) Operating System: DietPi (Linux)
2) Ping Tool: arp-scan
3) Programming: Python3, VScode
4) Sorting Tool: awk
5) Coding IOT: Arduino IDE
6) Imager: Raspberry Pi Imager

This Project is available at: https://github.com/rasheequeahnaf/csp450_project7

How it works

We start by setting up our access point using our router and set the ip range to allow 14 devices in a 255.255.255.240 subnet. Our network range will be 192.168.0.1/28

The DHCP has been turned off to better allocate each IOT device correctly. The 12 IOT devices are pre-programmed with their unique static IP to automatically connect to the access point with password and provide vehicle information from different parts of the vehicle upon request

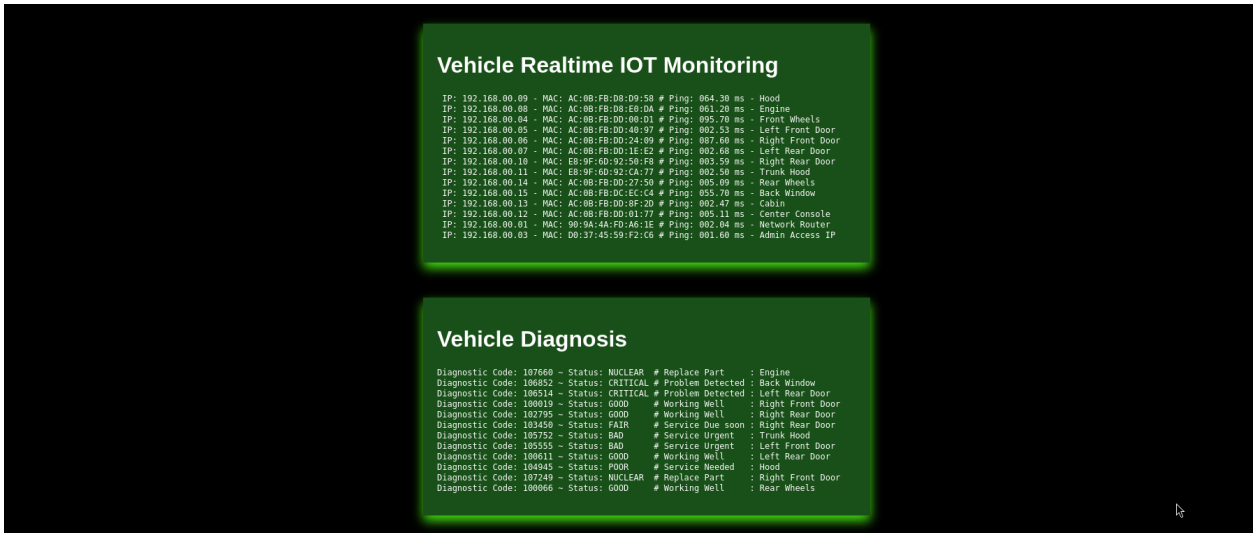
Then we install the operating system DietPi from this link **https://dietpi.com/downloads/images/DietPi_RPi-ARMv8-Bookworm.img.xz** and continue to set it up by exchanging keys between our host computer and Raspberry pi

We can either set up ssh keys before installation in raspberry pi imager or this can be done post installation.

Then we create a service called **serve.service** in **/etc/systemd/system/serve.service** and enable the service and start it.

The service will deploy a web server using python3 and every time the website is accessed or refreshed, it will present data output of **script.py** and **diagnosis.py**

Each unique IOT device is connected and responses to the control unit every time the website is accessed or refreshed



The first box provides iot information and the second box provides received error for diagnosis.

Definition of how each code works

diagnosis.py

This Python script generates random diagnostic codes and statuses for a specified number of times. It iterates over the specified number of times, generating a random diagnostic code within the range of 100000 to 108000 and determining its status based on predefined ranges. Additionally, it randomly selects a location from a specified list and appends all these details into a list of diagnostic codes. Finally, it prints out the generated diagnostic codes along with their statuses and corresponding locations.

script.py

This Python script retrieves information about connected devices in a local network using the `arp-scan` command. It executes the `arp-scan` command and parses its output to extract the IP addresses and MAC addresses of connected devices. Then, it attempts to ping each device to obtain its round-trip time (ping time) and determines whether the device is connected or under maintenance. It prints out the IP address, MAC address, ping time, and status (connected or under maintenance) for each predefined device in the network.

serve.py

This Python script serves as an HTTP server using the `http.server` module. It defines a custom request handler class that responds to GET requests by generating the output of two separate Python scripts (`script.py` and `diagnosis.py`). It executes these scripts using the `subprocess` module and embeds their outputs into an HTML template. The HTML template contains two containers, each displaying the output of one of the scripts. Upon receiving a GET request, the server sends this HTML template as the response. Additionally, it listens for incoming connections on port 80 and serves HTTP requests indefinitely.

Serve.service

The systemd service file `serve.service` manages a service called "Serve Service". This service is configured to start at boot. It runs with root privileges (`User=root`, `Group=root`) and sets the working directory to `/root/`. The service executes the command in `ExecStart`, which is `/usr/bin/python3 /root/serve.py`. This command launches a Python script called `serve.py` using Python 3 interpreter. The `serve.py` script serves the output of another script via HTTP

The `[Install]` section specifies that it will start automatically when the system enters multi-user mode.

ARDUINO_IOT_DEIVCE_CODE.ino

This code is written for the ESP8266 microcontroller using the Arduino IDE. It includes necessary libraries for handling WiFi connections and creating a web server asynchronously.

The `ssid` and `password` variables store the credentials for connecting to a WiFi network. The `ledPin` and `Ldigit` variables define the GPIO pins connected to an LED and a segment display, respectively.

In the `setup()` function, the LED pin is configured as an output, and the LED is turned on. Then, the `connectToWiFi()` function is called to establish a WiFi connection.

The `loop()` function continuously checks if the ESP8266 is connected to the WiFi network. If it is, the `blinkLED()` function is called to blink the LED.

The `connectToWiFi()` function configures the ESP8266 with a static IP address, gateway, and subnet, and then attempts to connect to the WiFi network. It waits in a loop until the connection is established.

The `blinkLED()` function toggles the LED pin to create a blinking effect and introduces a delay between blinks.

Overall, this code initializes the ESP8266, connects it to a WiFi network, and blinks an LED in a loop when the connection is established.

Works Cited

DietPi. (n.d.). [Website]. https://dietpi.com/#downloadinfo

Wood, R. (n.d.). *arp-scan* [Computer software]. https://github.com/royhills/arp-scan

Here are the APA citations for the software and tool you requested:

Python Software Foundation. (n.d.). *Python 3.* [Software]. https://www.python.org/

Weinberger, P. J., & Aho, A. V. (n.d.). *AWK Tool.* [Software]. https://www.gnu.org/software/gawk/

Raspberry Pi Foundation. (n.d.). *Raspberry Pi Imager.* [Software]. https://www.raspberrypi.com/software/

Arduino. (n.d.). *Arduino IDE.* [Software]. https://www.arduino.cc/en/software

TP-Link. (n.d.). *TP-Link Router WR902AC.* [Product]. https://www.tp-link.com/us/home-networking/