Associate IoT Engineer - Coding challenge

Problem Statement:

To ingest accelerometer data from a real BLE Tag and detect whether the tag is moving or stationary. A real BLE beacon with an accelerometer will be broadcasting the accelerometer data. You need to write code that will read this accelerometer data from the Bluetooth adapter of the laptop and then detect whether the tag is moving or stationary. You can write the code in any of c / c++/ python and it should execute on a laptop which has a Bluetooth adapter (both Linux and mac).

Please make sure you add the commands for compiling and running the code in the readme.

Solution As per my understanding:

To solve this problem, we need to:

- Read accelerometer data from the Bluetooth Low Energy (BLE) tag: Use the laptop's Bluetooth adapter to communicate with a BLE tag that broadcasts accelerometer data.
- Process the accelerometer data: Once the data is received, we need to analyse it to determine whether the tag is moving or stationary. Typically, this can be done by measuring the magnitude of acceleration over time.
- We'll write a Python script that uses a Bluetooth library to communicate with the BLE tag, collect the accelerometer data, and analyse it.

Requirements:

Python 3.x

bluepy library (for BLE communication)

numpy (for mathematical operations like calculating magnitude)

BLE tag broadcasting accelerometer data

Approach

Install Required Libraries:

bluepy for accessing BLE devices in Python.

numpy for handling acceleration data and computing magnitude.

Scan for the BLE device: The script will scan for nearby BLE devices and attempt to find the device with the accelerometer data.

Extract accelerometer data: The accelerometer data is usually broadcasted in a specific format (e.g., a 3-axis vector with x, y, and z values). We'll read the values of these axes.

Detect motion: Using the accelerometer data, calculate the magnitude of the acceleration vector. If the magnitude exceeds a threshold, we can assume the tag is moving; if it's below the threshold, the tag is stationary.

Step 1: Install Required Libraries

Make sure you have the required libraries installed.

```
pip install bluepy numpy
```

Step 2: Python Script for BLE Communication and Motion Detection

python

```
import time
import numpy as np
from bluepy.btle import Scanner, DefaultDelegate, Peripheral

# Define a motion detection threshold
MOVEMENT_THRESHOLD = 1.0  # Adjust this value based on the sensitivity required

class MyDelegate(DefaultDelegate):
    def __init__(self, mac_address):
        DefaultDelegate.__init__(self)
        self.mac_address = mac_address
        self.accel_data = None

    def handleNotification(self, cHandle, data):
        """
        This method is called when a notification is received from the BLE device.
        Here, we're assuming the data contains the accelerometer data.
        """
        # Extract 3-axis acceleration data (this will depend on the data format of your BLE tag)
        # Assuming data comes in the form of 6 bytes for X, Y, Z (2 bytes per axis)
        x = int.from_bytes(data[0:2], byteorder='little', signed=True)
        y = int.from_bytes(data[2:4], byteorder='little', signed=True)
        z = int.from_bytes(data[4:6], byteorder='little', signed=True)

        # Save the accelerometer data
        self.accel_data = np.array([x, y, z])
```

```
def calculate_magnitude(accel_data):
    """Calculate the magnitude of the acceleration vector."""
    return np.linalg.norm(accel_data)


def detect_motion(accel_data):
    """Detect if the object is moving based on the acceleration vector's magnitude."""
    magnitude = calculate_magnitude(accel_data)
    if magnitude > MOVEMENT_THRESHOLD:
        return "Moving"
    else:
        return "Stationary"


def scan_and_connect():
    """Scan for nearby BLE devices and connect to the target device."""
    scanner = Scanner()
    devices = scanner.scan(10.0)  # Scan for 10 seconds
    target_device = None

    # Find the device with the desired MAC address (replace with your device's address)
    target_mac = "XX:XX:XX:XX:XX:XX"  # Replace with your BLE device MAC address
    for dev in devices:
        if dev.addr == target_mac.lower():
            target_device = dev
            break

    if target_device is None:
        print("Device not found")
        return None

    print(f"Found device: {target_device.addr}")
    peripheral = Peripheral(target_device)
    peripheral.setDelegate(MyDelegate(target_device.addr))

    return peripheral

def main():
    peripheral = scan_and_connect()
    if peripheral is None:
        return

    print("Connected to device. Waiting for data...")

    try:
        while True:
            if peripheral.waitForNotifications(1.0):
                # Handle notifications
                pass

            if peripheral.delegate.accel_data is not None:
                # Process accelerometer data
                accel_data = peripheral.delegate.accel_data
                print(f"Acceleration Data: {accel_data}")
                status = detect_motion(accel_data)
                print(f"Device status: {status}")

    except KeyboardInterrupt:
        print("Terminating...")
    finally:
        peripheral.disconnect()
if __name__ == "__main__":
    main()
```

Step 3: Explanation of the Code

Libraries:

bluepy.btle: Used to communicate with BLE devices.

numpy: Used to handle the accelerometer data and calculate the magnitude of the acceleration vector.

Scanning and Connecting:

The Scanner class is used to discover nearby BLE devices. Once we find the target device (by its MAC address), we connect to it using the Peripheral class.

We set up a custom delegate class (MyDelegate) to handle notifications from the BLE tag. When the device sends accelerometer data, this delegate will capture it.

Motion Detection:

The accelerometer data is expected to be a set of 3 values (X, Y, Z), which are combined into a numpy array.

The magnitude of the acceleration vector is computed using the Euclidean norm (i.e.,

magnitude

$= \sqrt{x^2 + y^2 + z^2}$).

If the magnitude exceeds a threshold (defined as MOVEMENT_THRESHOLD), the device is considered "moving"; otherwise, it's considered "stationary".

Data Handling:

The program continuously listens for notifications from the BLE device. When new data is available, the accelerometer data is processed to determine the movement status.

Step 4: Compiling and Running the Code

1. Ensure Bluetooth is Enabled

Make sure that Bluetooth is enabled on your laptop and that it supports BLE (Bluetooth Low Energy).

2. Running the Code

To run the script, execute the following command:

```
python3 ble_motion_detection.py
```

This script will scan for BLE devices, connect to the specified device (using its MAC address), and print out whether the device is moving or stationary based on the accelerometer data.

Step 5: README Example

Here's how you might document the code in a README.md:

BLE Motion Detection

This Python script detects whether a BLE tag with an accelerometer is moving or stationary. It connects to the BLE device, reads the accelerometer data, and calculates the magnitude of the acceleration vector to determine motion status.

Requirements

- Python 3.x

- `bluepy` library for BLE communication

- `numpy` for mathematical operations

Installation

1. Install the required dependencies:

```
   pip install bluepy numpy
```

Update the target_mac variable in the script to match the MAC address of your BLE tag.

Run the script:

```
python3 ble_motion_detection.py
```

The script will scan for nearby BLE devices and attempt to connect to the specified device. It will continuously monitor the accelerometer data and print whether the device is "Moving" or "Stationary".

Conclusion:

The motion detection threshold (MOVEMENT_THRESHOLD) can be adjusted depending on the sensitivity needed.

The accelerometer data format may vary depending on your BLE device. Modify the data parsing logic if necessary.

This solution should work on Linux and macOS systems. You may need to adjust the device's MAC address and ensure that the accelerometer data is formatted as expected. Let me know if you need more help or if you encounter any issues!