Wahoo Device Driver Documentation

Overview

This documentation provides an overview of the Wahoo Device Driver located at Drivers/kickr_climb_and_smart_trainer/wahoo_device.py. The driver interfaces with KICKR smart trainers and KICKR climbs, utilizing MQTT topics for communication.

Environment Configuration

The driver requires specific environment variables defined in the .env file. These variables are loaded via a bash script to ensure proper configuration.

Driver Functions

The Wahoo Device Driver handles the following functions:

Connecting to the Device

To establish a connection to the KICKR device, use the following method:

code

```
device = WahooDevice(mac_address)

device.connect()  # No arguments needed
```

MQTT Topics

The following MQTT topics are used to control functions and retrieve data from the bikes. Each bike has a unique ID to prevent interference.

Bike Topic IDs

- **Bike 1**: 000001
- **Bike 2**: 000002
-

Topic Control Functions

MQTT Topic	**Function Description**
bike/000001/incline	Set the incline value for the KICKR climb.
bike/000001/resistance	Set the resistance value for the KICKR trainer.

Data Output Topics

MQTT Topic	**Data Output Description**
bike/000001/speed	Read the speed of the bike (m/s as a float).
bike/000001/cadence	Read the cadence of the bike (rpm as a float).
bike/000001/power	Read the power output of the bike (watts as an integer).
bike/000001/heartrate	Read the BPM of the heart rate monitor.
bike/000001/fan	Read the last accepted fan speed (to be updated in future).

Starting the Driver

To start the driver, execute the following command from the home directory:

bash

code

```
bash iot/scripts/start_kickr.sh
```

Functional Requirements

BLE Communication

• Connection: Establish a BLE connection with the Wahoo Kickr using the MAC address provided in the configuration file.

• Pairing: Manage the pairing process and handle connection states (connected, disconnected, reconnecting).

Control Commands

• **Resistance Control:** Commands to adjust the trainer's resistance.

• **Inclination Control:** Commands to adjust the trainer's inclination.

• **Error Handling:** Error messages are displayed if commands fail or if the BLE connection is lost.

Data Reporting

• **Real-time Data Collection:** Speed, cadence, and power are retrieved from the trainer.

• **MQTT Publishing:** The retrieved data is published to an MQTT broker with configurable topics.

Configuration Management

• **BLE Configuration:** The MAC address of the trainer must be specified to establish a connection.

• **MQTT Configuration:** The MQTT broker's address, port, and topics must be configured to publish data effectively.

Nonfunctional Requirements

• **Latency:** The BLE communication should have a latency of less than 200 milliseconds. MQTT data should be published within 100 milliseconds after collection.

• **Efficiency:** The driver should use system resources efficiently to ensure smooth performance during continuous operation.

Additional Information

The GATT code used in this driver can be referenced from gatt-python. It is recommended to document the GATT library separately as it is reused across different drivers.

Conclusion

This documentation provides the necessary details for developers to integrate and document the inputs and outputs of the Wahoo Device Driver for KICKR smart trainers and climbs. For further questions or issues, refer to the provided code or reach out to the support.