Fan Sensor Summary

Figure 1 – Image of KICKR Headwind Bluetooth Fan (Wahoo Fitness n.d.)

Introduction

The Smart Bike project makes use of the KICKR Headwind Bluetooth Fan by Wahoo Fitness which is pictured in figure 1. This device is designed to adjust its fan speed based on the efforts of the cyclist. Ultimately, if the intensity of the workout changes and consequently the cyclist's heartrate and speed changes, the fan will also change its speed accordingly. This sensor summary document will briefly explain some of the code required for this device, examples of the payload and how this device fits into the architecture of the Smart Bike project.

More information on the KICKR Headwind Bluetooth Fan can be found via the following links:
https://au.wahoofitness.com/devices/indoor-cycling/accessories/kickr-headwind-buy-au

https://au.wahoofitness.com/devices/indoor-cycling/accessories/kickr-headwind

Code

The KICKR Headwind Bluetooth Fan requires two scripts to operate as intended which includes 'mqtt_client.py' and 'fan.py'. The 'mqtt_client.py' script, as with the other sensors, provides a number of necessary functions needed to connect with/setup important components such as setting up Transport Layer Security (TLS), connecting to the HiveMQ MQTT Broker and also publishing and subscribing to MQTT topics. The 'fan.py' script is responsible for connecting to the KICKR Headwind Bluetooth Fan device and also using the functionalities provided by the 'mqtt_client.py' script to connect to the necessary components and to also publish and subscribe to MQTT topics via the 'mqtt_client.py' script. In this particular case, it needs to subscribe to the 'control' MQTT topic to listen to and respond to instructions and also report the current fan speed similar to other sensors.

The folder also contains two other scripts called ‘subsciber.py’ and ‘publish.py’. The ‘subsciber.py’ script subscribes to the relevant topics and prints what it receives to the console. This scripts is largely got debugging purposes. The ‘publish.py’ script is for testing purposes similar to that of ‘subsciber.py’, however, in this case it is used to publish speed values to MQTT.

Example Payloads

A table has been provided below showcasing example payloads. The first three are responsible for setting the speed of the fan and the last three report the fan speed at the given time. The value indicated in the payload column represents the fan of the speed with a range of 0 to 100.

Topic	Payload	Description
bike/00001/fan/control	100	Bike One’s fan is operating at 100% of its maximum speed
bike/00002/fan/control	65	Bike Two’s fan is operating at 65% of its maximum speed
bike/00002/fan/control	20	Bike Two’s fan is operating at 20% of its maximum speed
bike/00001/fan	{“ts”: 1029293309, “value”: 100}	Bike One’s fan is operating at 100% of its maximum speed
bike/00002/fan	{“ts”: 1029293309, “value”: 65}	Bike Two’s fan is operating at 65% of its maximum speed
bike/00002/fan	{“ts”: 1029293309, “value”: 20}	Bike Two’s fan is operating at 20% of its maximum speed

Architecture

Figure 2 showcases how the KICKR Headwind Bluetooth Fan fits into the architecture of the Smart Bike project. As with the other sensors, this device communicates with a Raspberry Pi and publishes to the HiveMQ MQTT broker before being stored in a database (indicated by MongoDB Database). The data stored in the database can then be used by end user software such as a mobile application, unity software and web browser.

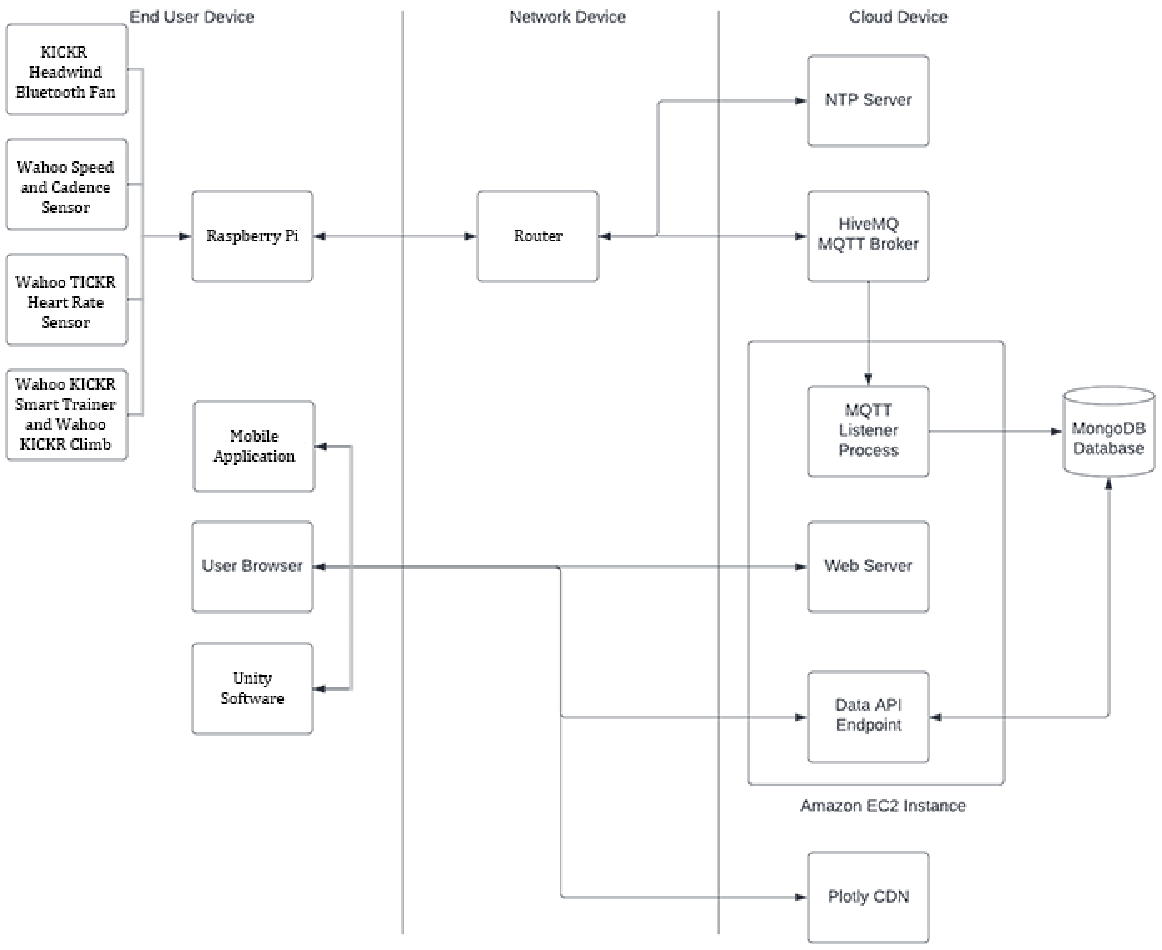

Figure 2 - Suitable Architecture based on the architecture provided by Adrian Grigo and modified by myself

References

Wahoo Fitness (n.d.) Image of KICKR Headwind Bluetooth Fan, Wahoo Fitness, accessed 25 September 2022. https://au.wahoofitness.com/devices/indoor-cycling/accessories/kickr-headwind-buy-au