Smart Bike Light API Documentation

Group 2

April 2025

1 Introduction

This documentation provides information about the Azure-deployed APIs for the Smart Bike Light system. These RESTful endpoints allow you to retrieve device information, historical data, and control device modes.

2 Base URL

```
https://bikelight-functions.azurewebsites.net/api/
```

3 Authentication

All API endpoints require an authentication code as a query parameter:

```
?code=PXFdKdQQHCcN6kTxGsbpryQgq_7-9YkB6b3FnGSwoE2mAzFuOHNNRw==
```

4 Endpoints

4.1 Get Device List

Retrieves a list of all available device IDs.

- **URL**: `/devices`
- **Method**: GET
- **Response**: A JSON array of device IDs

Example Request:

```
GET https://bikelight-functions.azurewebsites.net/api/devices?code=
    PXFdKdQQHCcN6kTxGsbpryQgq_7-9YkB6b3FnGSwoE2mAzFuOHNNRw==
```

Example Response:

```
[
  "bike-light-001",
  "bike-light-002"
]
```

4.2 Get Device Data

Retrieves the latest data for a specific device.

- **URL**: `/devices/{deviceId}`
- **Method**: GET
- **Parameters**:
  - `deviceId` (path): ID of the device
- **Response**: Device data including battery level, mode, location, and timestamp

Example Request:

```
GET https://bikelight-functions.azurewebsites.net/api/devices/bike-light-001?
    code=PXFdKdQQHCcN6kTxGsbpryQgq_7-9YkB6b3FnGSwoE2mAzFuOHNNRw==
```

Example Response:

```
{
  "device_id": "bike-light-001",
  "battery_level": 85,
  "mode": "active",
  "latitude": 37.7749,
  "longitude": -122.4194,
  "timestamp": "2025-04-18T10:30:45.123Z"
}
```

4.3 Get Device History

Retrieves historical data for a specific device, with optional date filtering.

- **URL**: `/devices/{deviceId}/history`
- **Method**: GET
- **Parameters**:
  - `deviceId` (path): ID of the device
  - `startDate` (query, optional): Filter data starting from this date (ISO format)
  - `endDate` (query, optional): Filter data up to this date (ISO format)
- **Response**: Array of historical device data records

Example Request:

```
GET https://bikelight-functions.azurewebsites.net/api/devices/bike-light-001/
    history?startDate=2025-04-10T00:00:00Z&endDate=2025-04-18T23:59:59Z&code=
    PXFdKdQQHCcN6kTxGsbpryQgq_7-9YkB6b3FnGSwoE2mAzFuOHNNRw==
```

Example Response:

```
[
  {
    "device_id": "bike-light-001",
    "battery_level": 85,
    "mode": "active",
    "latitude": 37.7749,
    "longitude": -122.4194,
    "timestamp": "2025-04-18T10:30:45.123Z"
  },
  {
    "device_id": "bike-light-001",
    "battery_level": 87,
    "mode": "park",
    "latitude": 37.7742,
    "longitude": -122.4191,
    "timestamp": "2025-04-17T14:22:33.456Z"
  }
  // More records...
]
```

4.4 Update Device Mode

Changes the operating mode of a specific device.

- **URL**: `/devices/{deviceId}/mode`
- **Method**: POST
- **Parameters**:
  - `deviceId` (path): ID of the device
- **Request Body**:

```
{
  "mode": "active" | "park" | "storage"
}
```

- **Response**: Success message

Example Request:

```
POST https://bikelight-functions.azurewebsites.net/api/devices/bike-light-001/
    mode?code=PXFdKdQQHCcN6kTxGsbpryQgq_7-9YkB6b3FnGSwoE2mAzFuOHNNRw==

{
  "mode": "park"
}
```

Example Response:

```
{
  "success": true,
  "message": "Mode change command sent to device bike-light-001"
}
```

5 Error Handling

All endpoints return standard HTTP status codes:

- 200: Success
- 400: Bad request (e.g., invalid mode)
- 404: Device not found
- 500: Server error

Error responses include a JSON body with an error message:

```
{
  "error": "Error message details"
}
```

6 Implementation Notes

- These APIs connect to an Azure Cosmos DB backend that stores device data
- For the updateDeviceMode endpoint, commands are sent via MQTT to the physical devices
- Historical data is limited to 100 most recent records per query

7 Background Processes

7.1 MQTT Capture Function

The system includes a background function called `mqttCapture` that runs every minute on Azure Functions. This process:

- Connects to the MQTT broker
- Listens to all device status and response topics
- Processes incoming messages
- Stores the data in Azure Cosmos DB
- Updates the latest status for each device

This automatic data collection ensures that device data is continuously updated in the database without requiring explicit API calls.

8 MQTT Message Format Specifications

For ESP32 developers integrating with this system, the following MQTT message formats must be followed to ensure proper interpretation by the `mqttCapture` function.

8.1 Topic Structure

- Device Status: `bikelights/{deviceId}/status`
- Device Commands: `bikelights/{deviceId}/commands`
- Device Responses: `bikelights/{deviceId}/responses`

8.2 Device Status Message

ESP32 devices should publish status updates to their respective status topics with the following JSON format:

```
{
  "deviceId": "bike-light-001",
  "batteryLevel": 85,
  "lightOn": true,
  "mode": "ACTIVE",
  "latitude": 37.7749,
  "longitude": -122.4194,
  "timestamp": "2025-04-18T10:30:45.123Z"
}
```

Required Fields:

- `deviceId` (string): Unique identifier of the device
- `batteryLevel` (number): Battery percentage between 0-100
- `mode` (string): Operating mode, must be one of: "ACTIVE", "PARK", "STORAGE"
- `timestamp` (string): ISO 8601 formatted timestamp

Optional Fields:

- `lightOn` (boolean): Whether the light is currently on
- `latitude` (number): GPS latitude in decimal degrees
- `longitude` (number): GPS longitude in decimal degrees
- `signalStrength` (number): Signal strength indicator (e.g., RSSI value)

8.3 Command Response Message

When receiving commands, ESP32 devices should respond to their respective response topics:

```
{
  "deviceId": "bike-light-001",
  "commandType": "MODE_CHANGE",
  "success": true,
  "timestamp": "2025-04-18T10:31:15.789Z",
  "message": "Mode changed to PARK successfully"
}
```

Required Fields:

- `deviceId` (string): Unique identifier of the device
- `commandType` (string): Type of command being responded to (e.g., "MODE_CHANGE", "LIGHT_CONTROL")
- `success` (boolean): Whether the command was executed successfully
- `timestamp` (string): ISO 8601 formatted timestamp

Optional Fields:

- `message` (string): Additional information about the response

8.4 Publishing Frequency

It is recommended that devices publish status updates:

- At least once every 5 minutes during normal operation
- Immediately after any significant state change (mode change, light on/off)
- Immediately after receiving and processing a command

The `mqttCapture` function polls for new messages every minute, so any published messages will be captured and saved to the database within this timeframe.