```
import mqtt from "mqtt";
import "dotenv/config";

import { randomUUID } from "node:crypto";

const DEVICE_ID = process.env.DEVICE_ID;
const PUBLISH_TOPIC = `/d2c/${DEVICE_ID}`;

function getHeartbeatMessage() {
    return {
        data: {
            ueid: randomUUID(),
            ts: new Date().toISOString(),
            did: DEVICE_ID,
            eid: 2001,
            pl: {
                uptime: "5h 32m 13sec",
        signalStrength: "55/70",
        battery: 85.45,
        temperature: 42.56
            }
        }
    }
}

const connectionOptions = {
    protocol: "mqtts",
    protocolVersion: 5,
    host: process.env.MQTT_HOST,
    clientId: DEVICE_ID,
    port: 8883,
    username: DEVICE_ID,
    password: process.env.DEVICE_PASSWORD,
    rejectUnauthorized: false,
    clean: false,
    properties: {
        sessionExpiryInterval: 86400,
    },
}

console.log('Connecting with:');
console.log('Host:', process.env.MQTT_HOST);
```

```
console.log('Username:', DEVICE_ID);
console.log('Password:', process.env.DEVICE_PASSWORD);
console.log('Client ID:', DEVICE_ID);

const client = mqtt.connect(connectionOptions);

let intervalId;

client.on("connect", () => {
    console.log(`${DEVICE_ID} connected`);

    intervalId = setInterval(() => {
        const message = getHeartbeatMessage();
        client.publish(PUBLISH_TOPIC, JSON.stringify(message), { qos: 1 },
(err) => {
            if (err) {
                console.error("Publish error:", err);
            } else {
                console.log("Message sent:", message);
            }
        });
    }, 10000);
});

client.on("close", () => {
    console.log("Connection with MQTT broker is lost. Clearing out the
interval");
    clearInterval(intervalId);
});

client.on("error", (error) => {
    console.log("Error in connecting to MQTT broker : ", error);
    clearInterval(intervalId);
});

client.on("message", (topic, message) => {
    console.log(`Received message: ${message.toString()} on ${topic}`);
});
```