HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

BÁO CÁO BÀI TẬP LỚN

ĐỀ TÀI: XÂY DỰNG THÙNG RÁC THÔNG MINH

Giảng viên: ***TS. TRẦN TIẾN CÔNG***

Nhóm học phần: ***04***

Nhóm bài tập: ***01***

Thực hiện:

Nguyễn Đức Anh	***B20DCCN057***
Nguyễn Thanh Trúc	***B20DCCN693***
Đậu Anh Quân	***B20DCCN545***

Mục lục

I. MÔ TẢ HỆ THỐNG

Các chức năng chính của hệ thống:

a. Tính năng tự động mở khi có người tiến đến gần

- Mục tiêu: Giúp người dùng dễ dàng bỏ rác mà không cần chạm tay vào nắp thùng, đảm bảo vệ sinh và an toàn.
- Giải pháp: Sử dụng cảm biến siêu âm để phát hiện sự hiện diện của người dùng. Khi phát hiện có người, nắp thùng sẽ tự động mở.
- Yêu cầu:
    - Khoảng cách phát hiện: 5 cm
    - Thời gian mở nắp: 1 giây
    - Độ nhạy: Không quá nhạy, tránh mở nắp khi không có người

b. Tính năng tự động đóng khi không có người ở gần

- Mục tiêu: Giúp tiết kiệm điện năng và tránh tình trạng nắp thùng bị mở lâu gây mất vệ sinh.
- Giải pháp: Sử dụng cảm biến siêu âm để phát hiện sự hiện diện của người dùng. Khi không phát hiện có người trong khoảng thời gian nhất định, nắp thùng sẽ tự động đóng.
- Yêu cầu:
    - Thời gian đóng nắp: 1 giây
    - Khoảng thời gian không phát hiện người: 3 giây

c. Tính năng phát hiện thùng rác đầy

- Mục tiêu: Giúp người dùng biết được khi nào cần đổ rác, tránh tình trạng thùng rác bị tràn gây mất vệ sinh.
- Giải pháp: Sử dụng cảm biến siêu âm để đo của rác trong thùng. Khi trọng lượng rác vượt quá một ngưỡng nhất định, hệ thống sẽ phát hiện thùng rác đầy.
- Yêu cầu:
    - Ngưỡng đầy: 80% – 100% thùng

d. Tính năng thông báo cho người dùng

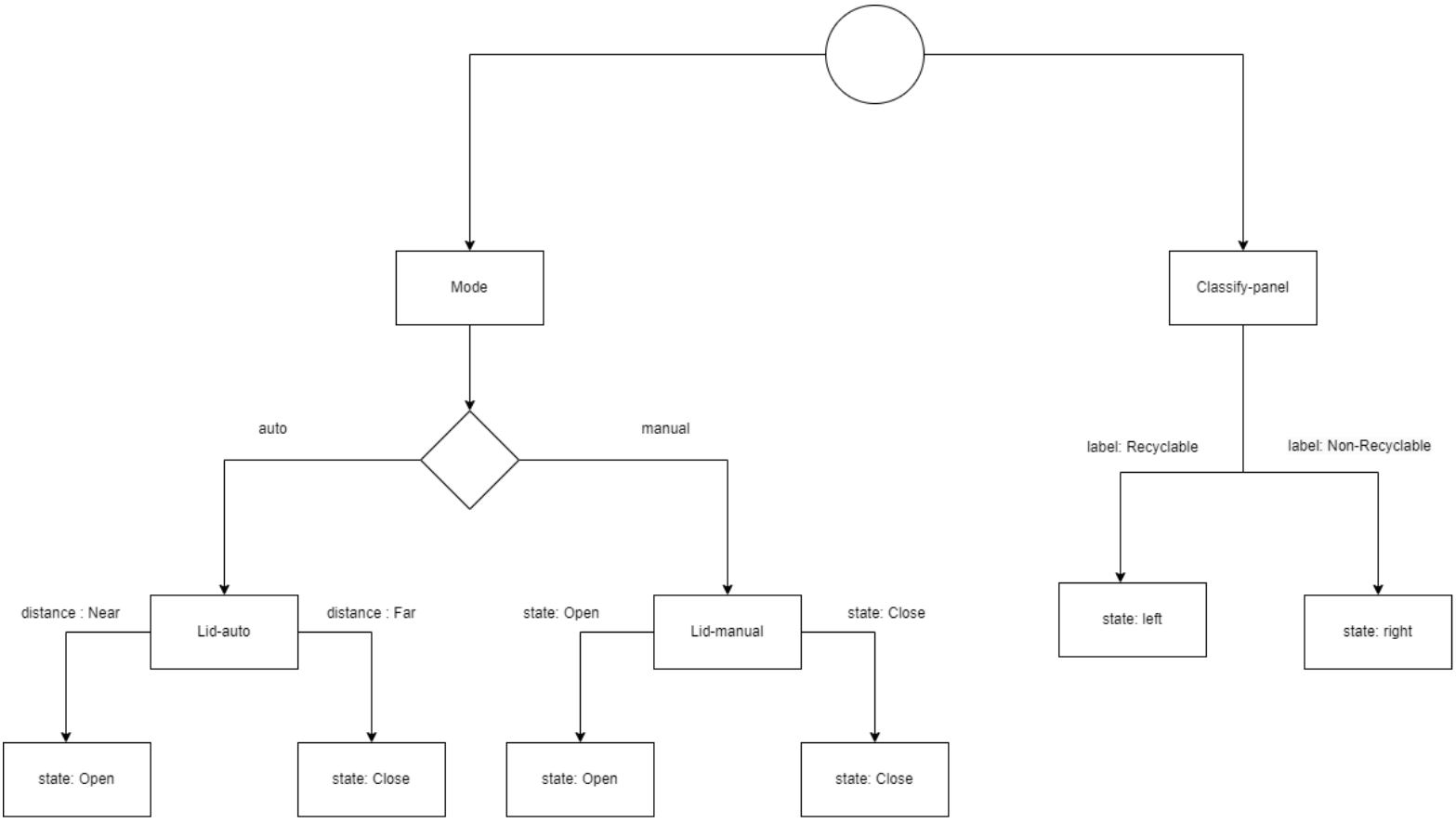
- Mục tiêu: Thông báo cho người dùng khi thùng rác đầy để người dùng có thể đổ rác.
- Giải pháp: Thông báo trên ứng dụng để thông báo cho người dùng.

- Yêu cầu:
    - Thời gian thông báo : Tức thời

e. Tính năng phân loại rác

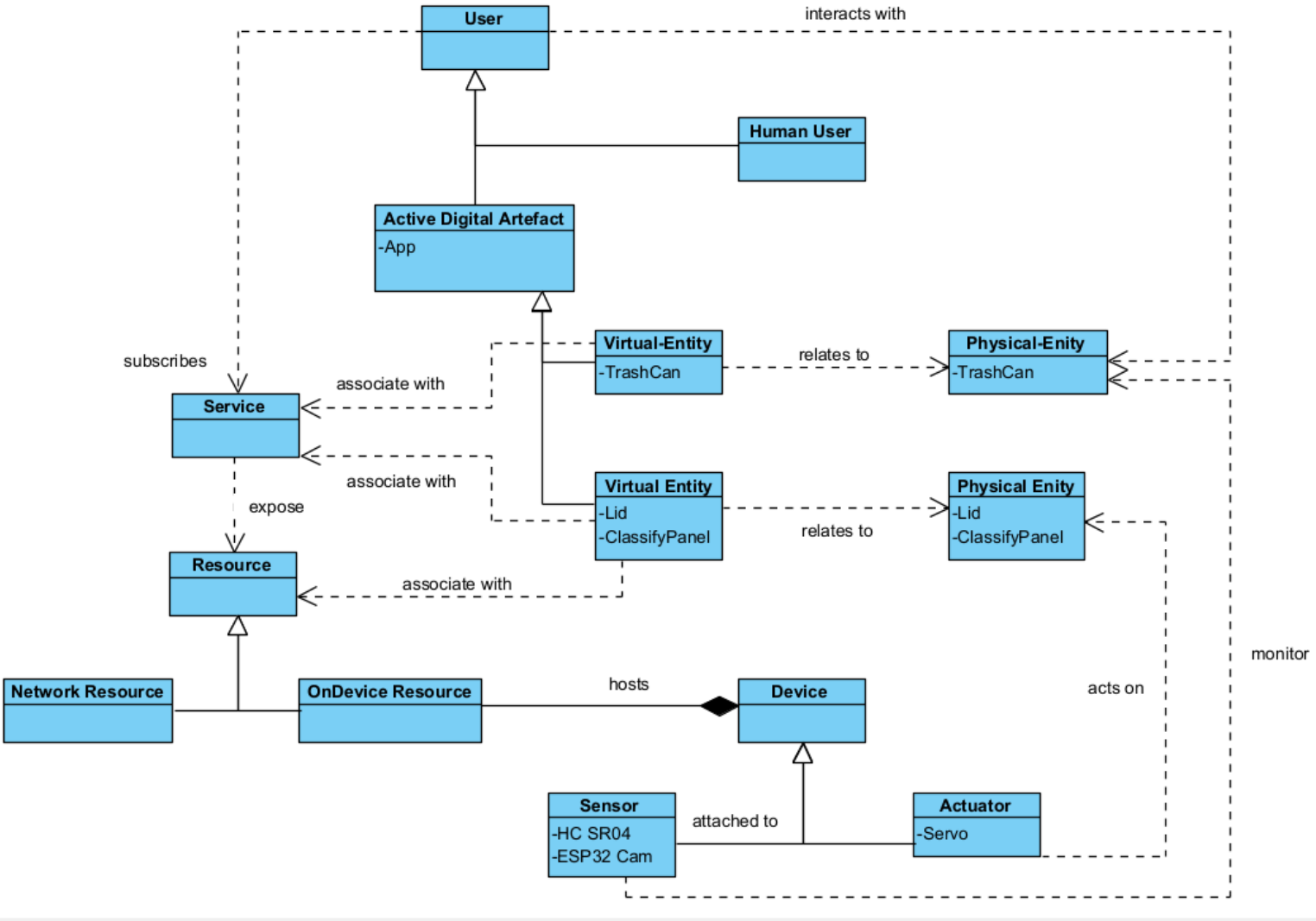
- Mục tiêu: Giúp phân loại rác thải một cách chính xác, góp phần bảo vệ môi trường.
- Giải pháp: Sử dụng công nghệ nhận dạng hình ảnh hoặc kỹ thuật trí tuệ nhân tạo để phân loại rác thải.
- Yêu cầu:
    - Độ chính xác: Trên 90%
    - Thời gian phân loại: Dưới 1 giây

II. ĐẶC TẢ TIẾN TRÌNH



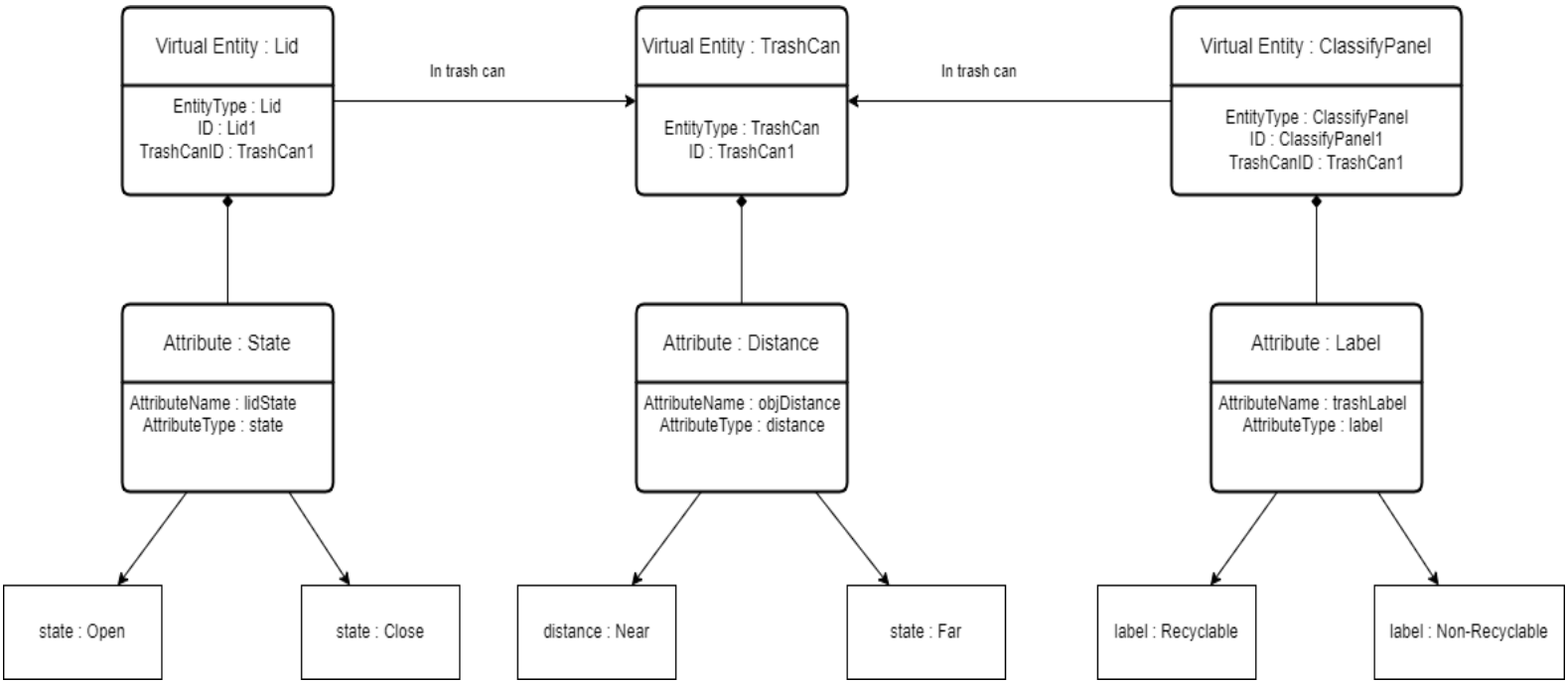
Hình 1: *Sơ đồ đặc tả tiến trình*

III. ĐẶC TẢ MÔ HÌNH NỀN



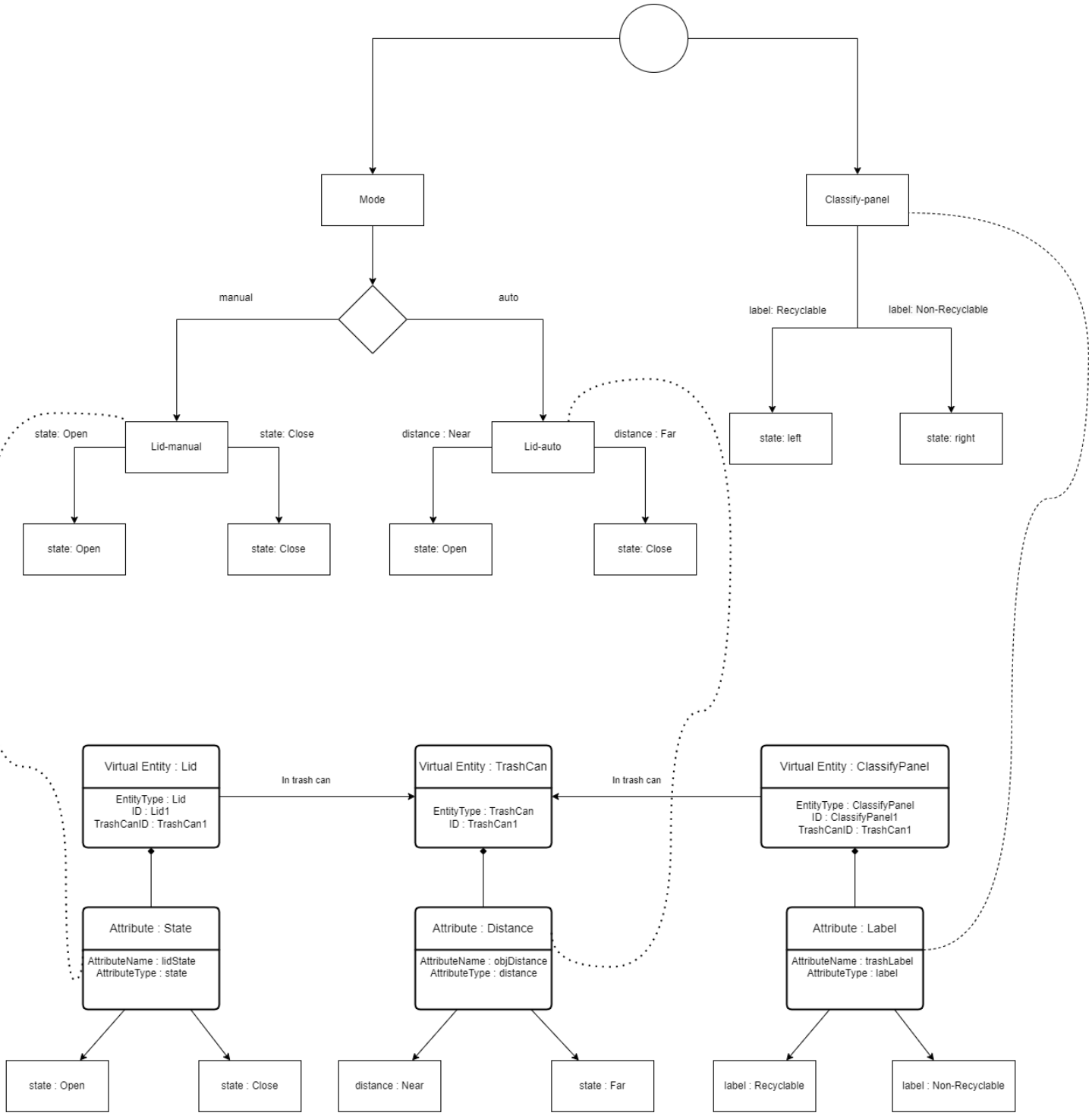
Hình 2: *Sơ đồ đặc tả mô hình nên*

IV. ĐẶC TẢ MÔ HÌNH THÔNG TIN

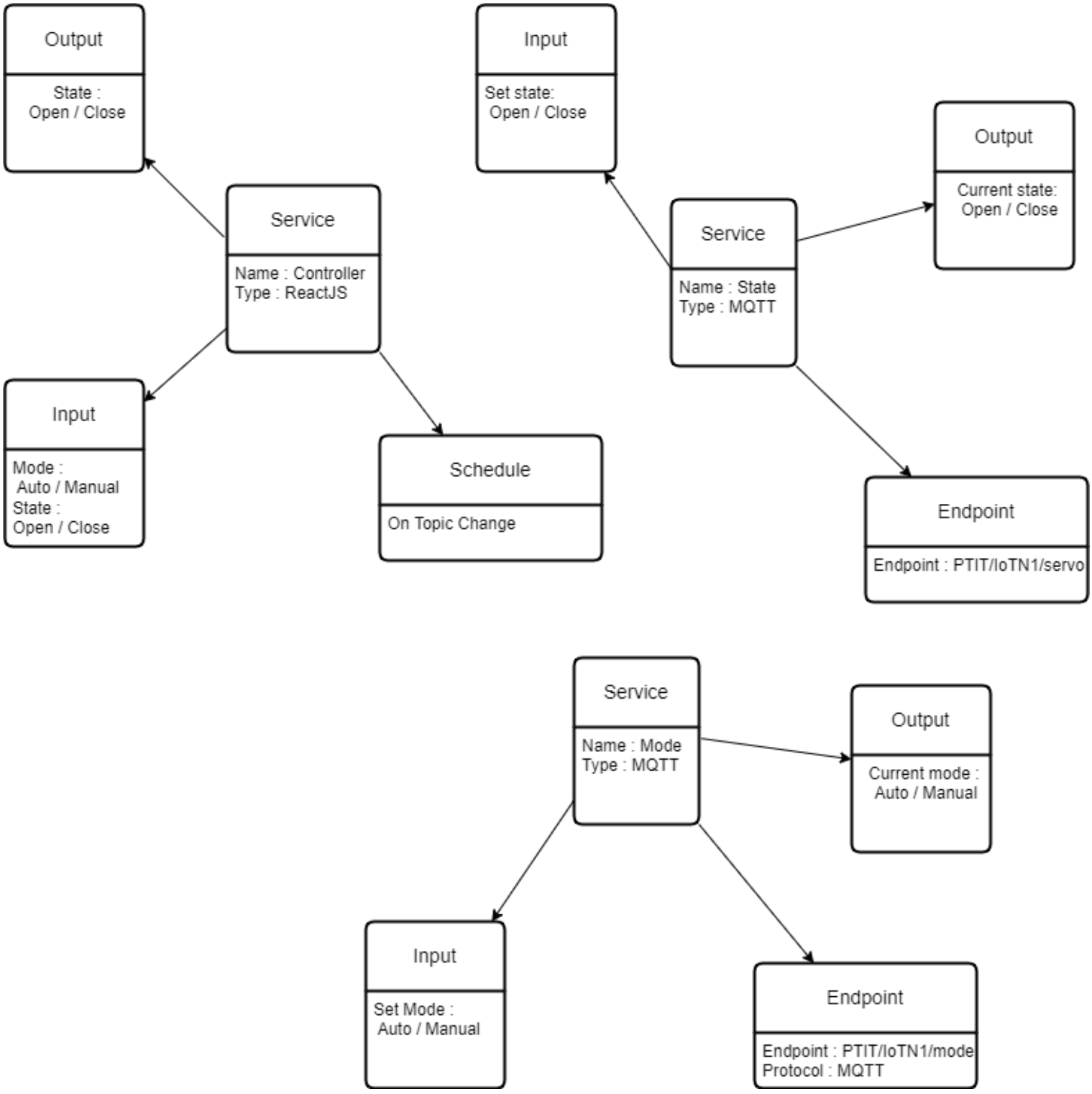


Hình 3 :*Sơ đồ đặc tả mô hình thông tin*

V. THÔNG SỐ DỊCH VỤ



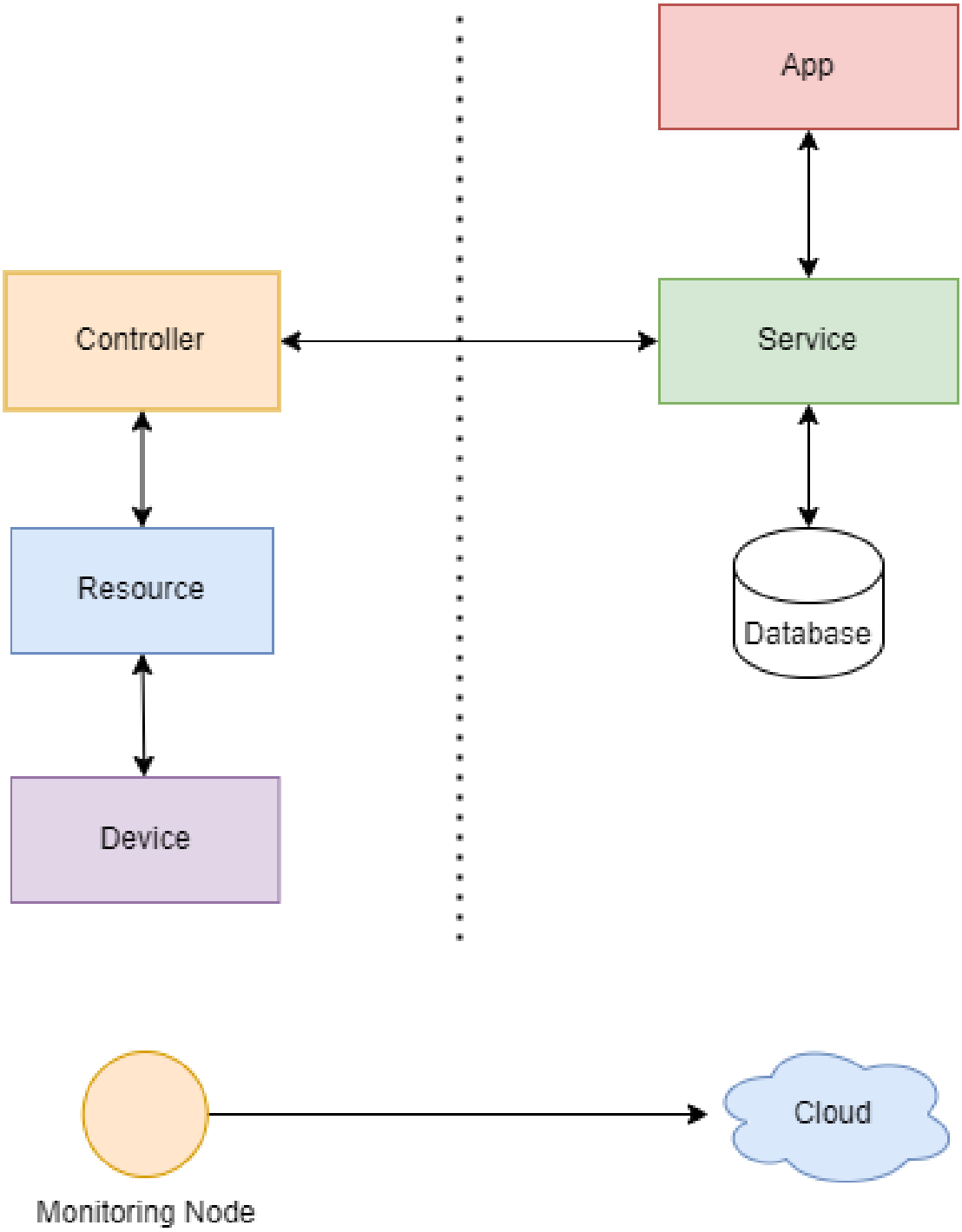
Hình 4.1 :*Sơ đồ thông số dịch vụ*



Hình 4.2: *Sơ đồ thông số dịch vụ*

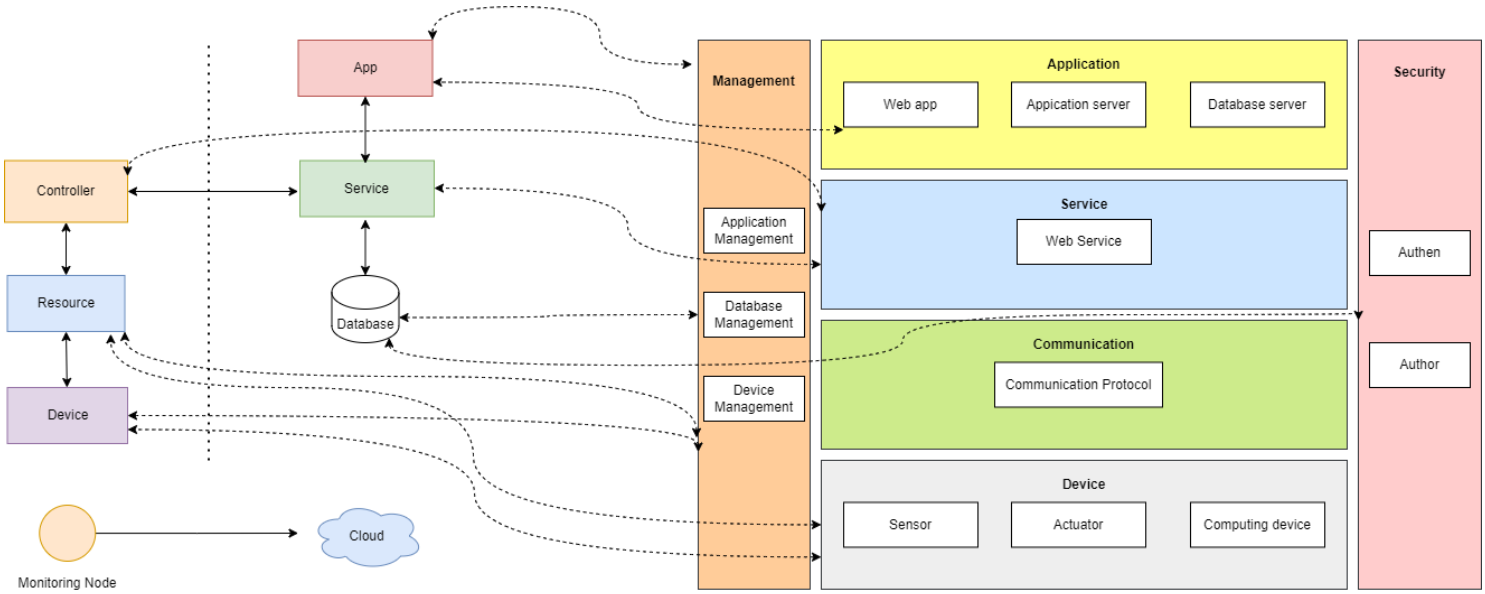
VI. ĐẶC TẢ CẤP ĐỘ IOT

Cấp độ IoT : Cấp độ 3



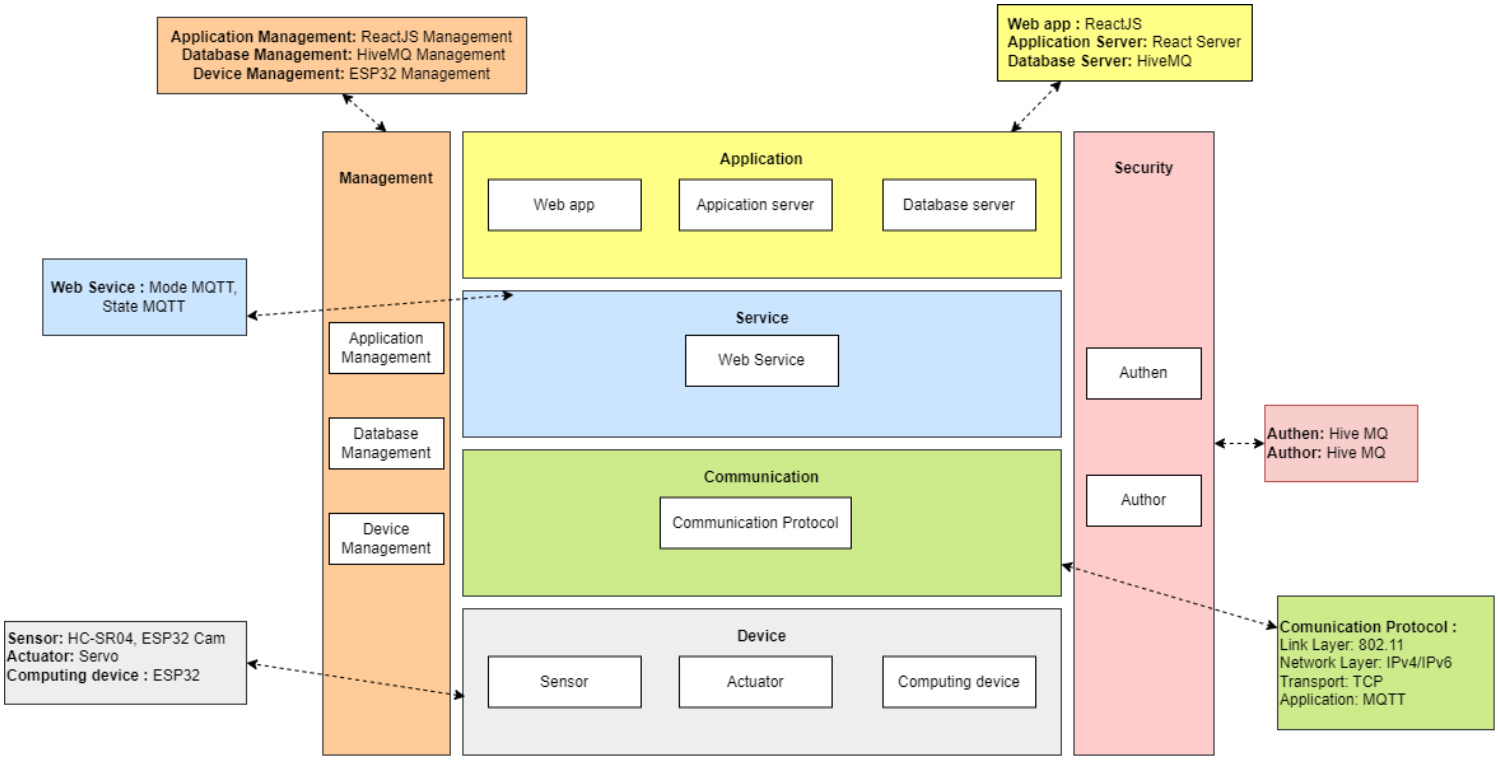
Hình 5:*Sơ đồ đặc tả cấp độ*

VII. ĐẶC TẢ THÀNH PHẦN CHỨC NĂNG



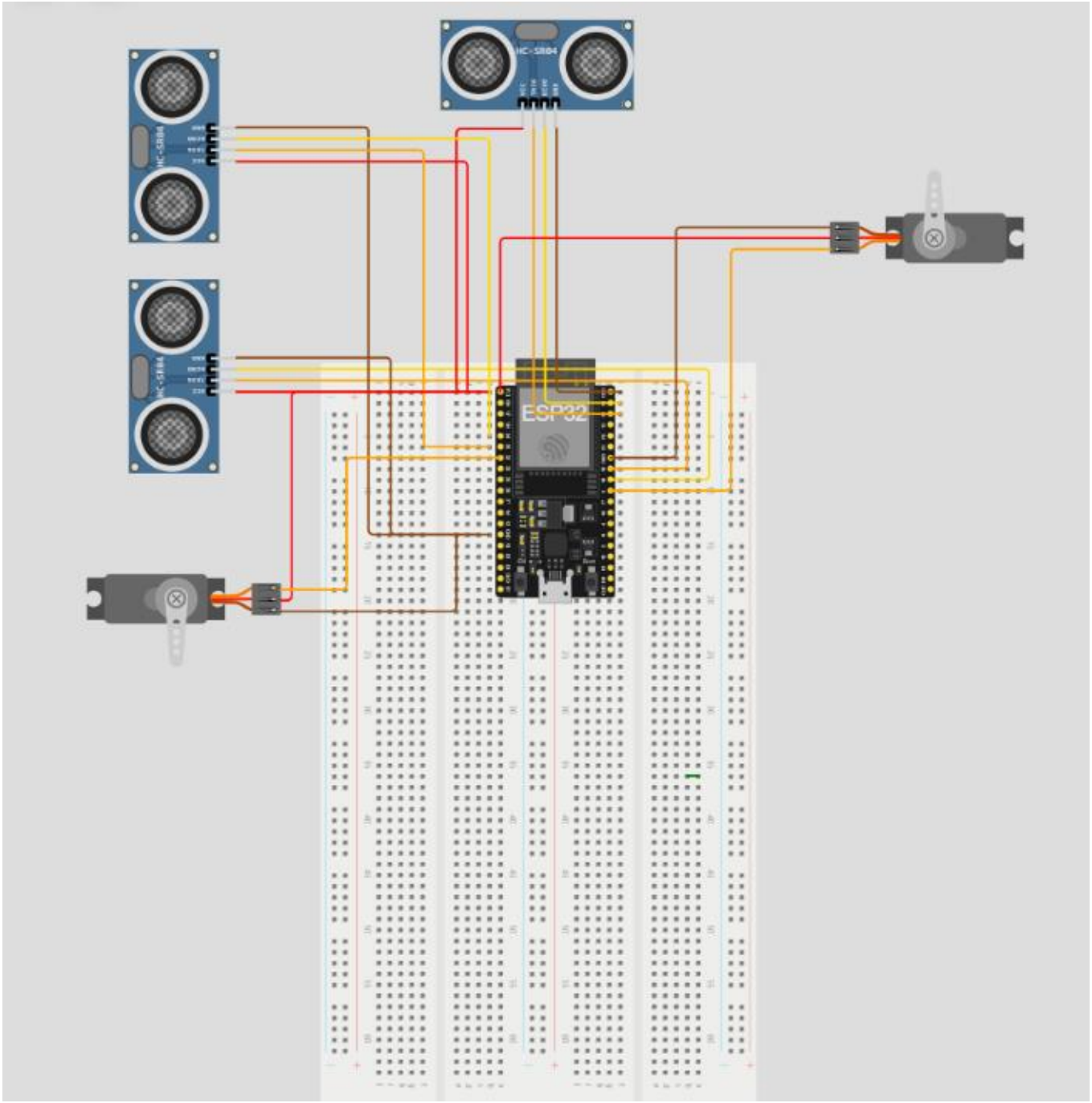
Hình 6: *Sơ đồ đặc tả thành phần chức năng*

VIII. ĐẶC TẢ THÀNH PHẦN HOẠT ĐỘNG



Hình 7 : *Sơ đồ đặc tả thành phần hoạt động*

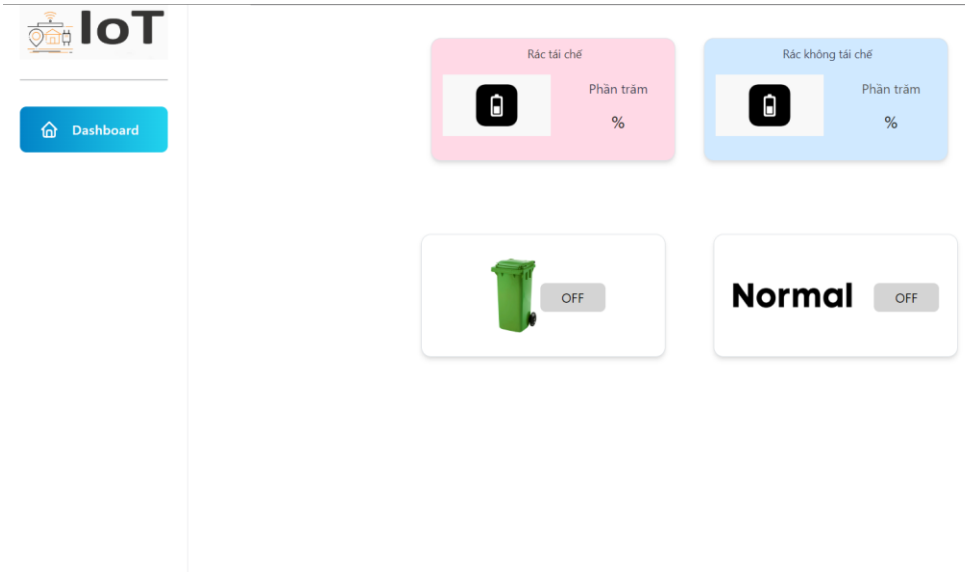
IX. TÍCH HỢP THIẾT BỊ



Hình 8 : *Sơ đồ tích hợp các thiết bị*

X. PHÁT TRIỂN ỨNG DỤNG

a. Ứng dụng toàn bộ hệ thống :



Hình 9.1 : *Giao diện ứng dụng*

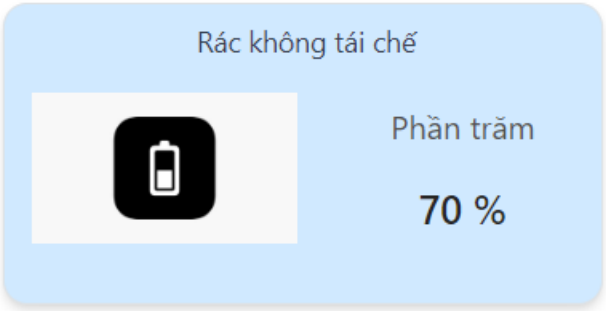
Thông báo lượng rác tái chế :

- Thông báo lượng rác trong ngăn rác tái chế tự động

Hình 9.2 : *Giao diện ứng dụng*

Thông báo lượng rác không tái chế :

- Thông báo lượng rác trong ngăn rác không tái chế tự động



Hình 9.3 : *Giao diện ứng dụng*

Thông báo khi thùng rác đóng / mở :

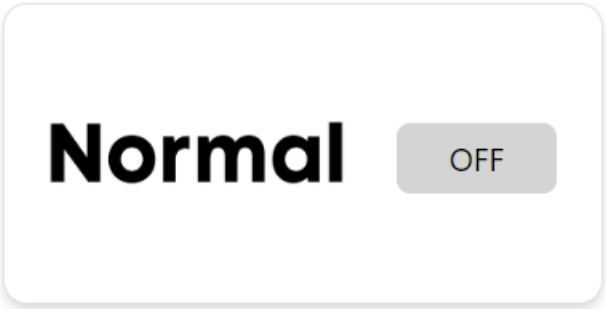
Hình 9.4: *Giao diện ứng dụng*

Hình 9.5: *Giao diện ứng dụng*

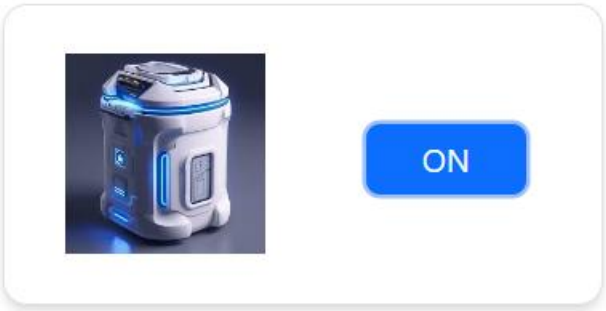
Thông báo chế độ bình thường / thông minh:

-Khi ở chế độ bình thường : Chế độ này để điều khiển thiết bị theo cách thủ công.

-Khi ở chết độ thông minh : Ở chế độ này thiết bị có thể tự động đóng mở nắp 1 cách thông minh khi có người lại gần



Hình 9.6 : *Giao diện ứng dụng*



Hình 9.7 : *Giao diện ứng dụng*

b. Code :

- Publish và lấy dữ liệu thông qua mqtt :
  Server.js

```
JS server.js > ...
const express = require('express');
const mqtt = require('mqtt');
const app = express();
const port = 8688;
const bodyParser = require('body-parser');
const cors = require('cors');
app.use(cors());
app.use(bodyParser.urlencoded({ extended: true }));//
app.use(bodyParser.json());
app.use('/data', express.static('data'));
const server = require('http').Server(app);
const io = require('socket.io')(server, {
    cors: {
        origin: 'http://localhost:3000',
    },
});
server.listen(port);
var client = mqtt.connect('mqtt://broker.hivemq.com');
client.on('connect', function () {
    console.log('mqtt connected');
    client.subscribe('PTIT/IoTN1/servo');
    client.subscribe('PTIT/IoTN1/dulieu');
    client.subscribe('PTIT/IoTN1/mode');
});
client.on('message', function (topic, message) {
    if (topic == 'PTIT/IoTN1/dulieu') {
        const data = JSON.parse(message);
        io.emit('napthung', data.napthung);
        io.emit('mode', data.mode);
        io.emit('taiche', data.taiche);
        io.emit('khongtaiche', data.khongtaiche);
        console.log(data.taiche, data.khongtaiche, data.napthung, data.mode);
    }
});
io.on('connection', function (socket) {
    socket.on('control_relay_1', function (state1) {
        if (state1 == '1') {
            client.publish('PTIT/IoTN1/servo', '1')
        } else {
            client.publish('PTIT/IoTN1/servo', '0')
        }
    });
    socket.on('control_relay_2', function (state2) {
        if (state2 == '1') {
            client.publish('PTIT/IoTN1/mode', '1')
        } else {
            client.publish('PTIT/IoTN1/mode', '0')
        }
    });
});
```

Hình 10.1 : *Code bên ứng dụng*

TrashCan.js:

```
const TrashCan = ({
    mode,
    setMode

}) => {
    const socket = io('http://localhost:8688');

    useEffect(() => {

        socket.on('mode', (data) => {
            setMode(data);
        });
        return () => {
            socket.disconnect();
        };
    }, [socket]);

    const handleTurnOnMode = () => {
        setMode(true);
        socket.emit('control_relay_2', 1);
    };

    const handleTurnOffMode = () => {
        setMode(false);
        socket.emit('control_relay_2', 0);
    };
```

Hình 10.2 : *Code bên ứng dụng*

Lid.js:

```
const Lid = ({
    state,
    setState,

}) => {
    const socket = io('http://localhost:8688');

    useEffect(() => {

        socket.on('napthung', (data_received) => {
            setState(data_received);
        });
        return () => {
            socket.disconnect();
        };
    },);

    const handleTurnOn = () => {
        setState(true);
        socket.emit('control_relay_1', 1);
    };

    const handleTurnOff = () => {
        setState(false);
        socket.emit('control_relay_1', 0);
    };
```

Hình 10.3 : *Code bên ứng dụng*