به نام خدا

درس : آزمایشگاه سخت افزار

استاد : دکتر اجلالی

گزارش پیشرفت

اعضای گروه :

محمدمعین صمدی آزاد ۴۰۰۱۰۵۰۹۳

امیرحسین عزیزی ۴۰۰۱۰۵۱۲۲

محمدشایان شعبانی ۴۰۰۱۰۵۰۶۹

تابستان ۱۴۰۳

شرح پیشرفت:

بعد از راه اندازی سنسور ها و عملگر ها، راه اندازی بروکر و سرور، و در نهایت اتصال این اجزا به یک دیگر، نوبت به مرحله ارائه دادن یک رابط کاربری به یوزر رسید، در این مرحله، با اعمال تغییرات تذکر داده شده توسط دستیار محترم، ساختار کد ها به ویژه در سرور دستخوش تغییراتی شد تا قابلیت توسعه بیشتری داشته باشد. برای این منظور برای دستگاه ها کلاس های مختلفی ایجاد شد و همچنین بر اساس این کلاس ها تمپلیت HTML مناسب برای درخواست های یوزر ایجاد می شود، همچنین با اتصال سرور به ردیس، اطلاعات سنسور ها ذخیره و نگه داری میشود.

نمونه کد ها:

کلاس های فعلی:

```
class IntegerActuator(Device):
    def __init__(self, name, topic, mqtt_client, redis_client):
        super().__init__(name, topic, mqtt_client, redis_client)

    def send_int(self, number):
        self.publish_message(str(number))


class Sensor(Device):
    def __init__(self, name, topic, mqtt_client, redis_client):
        super().__init__(name, topic, mqtt_client, redis_client)

    def on_message(self, message):
        timestamp = int(time.time())
        self.redis_client.setex(f"{self.name}:{timestamp}", 3600, message)

    def get_data(self):
        current_time = int(time.time())
        one_hour_ago = current_time - 3600

        recent_messages = []

        for key in self.redis_client.scan_iter(f"{self.name}:*"):
            key_timestamp = int(key.decode().split(':')[1])
            if one_hour_ago <= key_timestamp <= current_time:
                message = self.redis_client.get(key).decode("utf-8")
                recent_messages.append((key_timestamp, message))

        recent_messages.sort()

        return recent_messages
```

```python
class Device:
    def __init__(self, name, topic, mqtt_client, redis_client):
        self.name = name
        self.topic = topic
        self.mqtt_client = mqtt_client
        self.redis_client = redis_client
        self.elements = None

    def publish_message(self, message):
        self.mqtt_client.publish(self.topic, message)

    def on_message(self, message):
        pass

    def get(self):
        pass

    def get_element(self):
        pass


class BooleanActuator(Device):
    def __init__(self, name, topic, mqtt_client, redis_client):
        super().__init__(name, topic, mqtt_client, redis_client)

    def turn_on(self):
        self.publish_message('on')

    def turn_off(self):
        self.publish_message('off')
```

بخش دستگاه های کد تمپلیت:

```
    {% if device.name == 'boolean actuator' %}
        <div>
            <button type="button" class="btn btn-success" onclick="toggleActuator('on')">On</button>
            <button type="button" class="btn btn-danger" onclick="toggleActuator('off')">Off</button>
        </div>
    {% endif %}

    {% if device.name == 'integer actuator' %}
        <div>
            <form action="{{ device.topic }}/" method="POST">
                {% csrf_token %}
                <div class="form-group">
                    <label for="integerValue">Enter Integer Value:</label>
                    <input type="number" class="form-control" id="integerValue" name="integerValue" required>
                </div>
                <button type="submit" class="btn btn-primary">Submit</button>
            </form>
        </div>
    {% endif %}

{% endfor %}

<script>
    function toggleActuator(state) {
        const url = '{{ device.topic }}/' + state;
        fetch(url, { method: 'POST' })
            .then(response => {
                if (response.ok) {
                    return response.json();
                }
```

در مرحله بعدی، با افزودن CSS ، رابط کاربری را زیبا تر کرده و همینطور نمودار زنده برای سنسور ها را نمایش خواهیم داد.