HALLYM UNIVERSITY

CIC Convergence Information & Communications

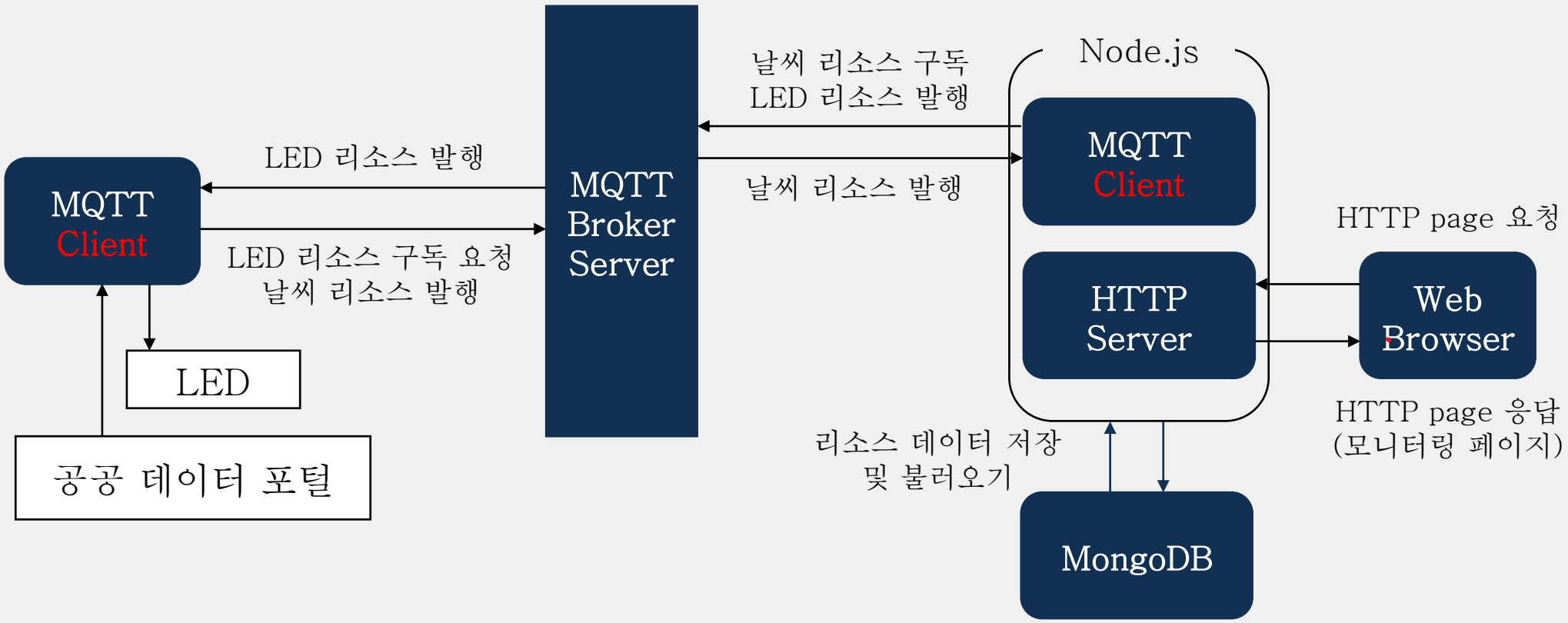
(IoT Network) Practice -11-

Node.js 서버에서 MQTT Publish 구현

Index

Practice Overview

- **Mini Project (수정)**
  - 주제: 기관지 만성질환자를 위한 우리동네 날씨 모니터링



www.js 파일 수정

- www.js 파일

```
104     // 미세먼지 데이터
105     var cursor = dbObj.db("Resources").collection("PM");
106     var options = {sort:{"_id":-1}, projection: {_id:0, pm:1, creat_at:1},};
107     var sending_data =cursor.find({},options).limit(1);
108     sending_data.toArray(function(err,results){
109       if(!err){
110         socket.emit("socket_up_pm", JSON.stringify(results[0]));
111       }
112     });
113   });
114   // LED 버튼 입력시 MQTT 정보 발행
115   socket.on("socket_evt_bnt", function(data){
116     console.log("MQTT Publish- LED control");
117     console.log(data);
118     client.publish("led", data);
119   });
120 });
```

MQTT.html 파일 수정

- MQTT.html 파일

```
27      function timer_1(){
28          socket.emit("socket_evt_update", JSON.stringify({}));
29      }
30      function button_on(){
31          socket.emit("socket_evt_bnt", "ON");
32      }
33      function button_off(){
34          socket.emit("socket_evt_bnt", "OFF");
35      }
36
37  </script>
38  </head>
39  <body>
40  MQTT Mornitoring Service
41      <div id="msg">
42          <div id="mqtt_logs">
43              <ul class="mqttlist_temp"></ul>
44              <ul class="mqttlist_humi"></ul>
45              <ul class="mqttlist_pm"></ul>
46              <button id="button1" onclick="button_on();"><b>LED ON</b></button>
47              <button id="button2" onclick="button_off();"><b>LED OFF</b></button>
48          </div>
49      </div>
50  </body>
```

MqttPublisher_API.java 수정

- **MqttPublisher_API.java 수정**

```java
public class MqttPublisher_API implements MqttCallback{ // implement callback 추가 & 필요한 메소드 정의
    static MqttClient sampleClient;// Mqtt Client 객체 선언

    public static void main(String[] args) {
        MqttPublisher_API obj = new MqttPublisher_API();
        obj.run();
    }
    public void run() {
        connectBroker(); // 브로커 서버에 접속
        try { // 여기 추가
            sampleClient.subscribe("led"); // LED 리소스 구독
        } catch (MqttException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        while(true) {
            try {
                String[] weather_data  = get_weather_data(); // 공공 API
                String pm_data = get_pm_data(); // 공공 API
                publish_data("tmp", "{\"tmp\": "+weather_data[0]+"}"); // 온도 데이터 발행
                publish_data("humi", "{\"humi\": "+weather_data[1]+"}"); // 습도 데이터 발행
                publish_data("pm", "{\"pm\": "+pm_data+"}"); // 미세먼지 데이터 발행
                Thread.sleep(5000); // @@@@@@
            }catch (Exception e) {
                // TODO: handle exception
                try {
                    sampleClient.disconnect();
                } catch (MqttException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                }
                e.printStackTrace();
                System.out.println("Disconnected");
                System.exit(0);
            }
        }
    }
```

MqttPublisher_API.java 수정

- **MqttPublisher_API.java 수정**

```
56    public void connectBroker() {
57        String broker = "tcp://127.0.0.1:1883"; // 브로커 서버의 주소
58        String clientId = "practice"; // 클라이언트의 ID
59        MemoryPersistence persistence = new MemoryPersistence();
60        try {
61            sampleClient = new MqttClient(broker, clientId, persistence);// Mqtt Client 객체 초기화
62            MqttConnectOptions connOpts = new MqttConnectOptions(); // 접속시 접속의 옵션을 정의하는 객체 생성
63            connOpts.setCleanSession(true);
64            System.out.println("Connecting to broker: "+broker);
65            sampleClient.connect(connOpts); // 브로커서버에 접속
66            sampleClient.setCallback(this);// Call back option 추가
67            System.out.println("Connected");
68        } catch(MqttException me) {
69            System.out.println("reason "+me.getReasonCode());
70            System.out.println("msg "+me.getMessage());
71            System.out.println("loc "+me.getLocalizedMessage());
72            System.out.println("cause "+me.getCause());
73            System.out.println("excep "+me);
74            me.printStackTrace();
75        }
76    }
```

MqttPublisher_API.java 수정

- **MqttPublisher_API.java 수정**

```
167     ///@@@@@@@@@@@@@@@@@@
168     @Override
169     public void connectionLost(Throwable arg0) {
170         // TODO Auto-generated method stub
171         System.out.println("Connection lost");
172     }
173
174     @Override
175     public void deliveryComplete(IMqttDeliveryToken arg0) {
176         // TODO Auto-generated method stub
177     }
178
179     @Override
180     public void messageArrived(String topic, MqttMessage msg) throws Exception {
181         // TODO Auto-generated method stub
182         if (topic.equals("led")){
183             System.out.println("--------------------Actuator Function--------------------");
184             System.out.println("LED Display changed");
185             System.out.println("LED: " + msg.toString());
186             System.out.println("---------------------------------------------------------");
187         }
188     }
189 }
```