임베디드 시스템 프로젝트

# 감정과 대기 상태에 따른 영화 추천 시스템

20213242 이동민

# 목차

# 프로젝트 배경 및 목적


## 감정, 날씨 등에 따른 영화 추천 기능 없음

- OTT 서비스는 이용자의 평가(별점)을 기반으로 영화 추천
- 감정과 날씨는 밀접한 연관성을 갖음
- 현재 이용자의 감정이나 날씨 등에 따른 영화 추천 기능은 아직 없음

## 영화는 종합예술

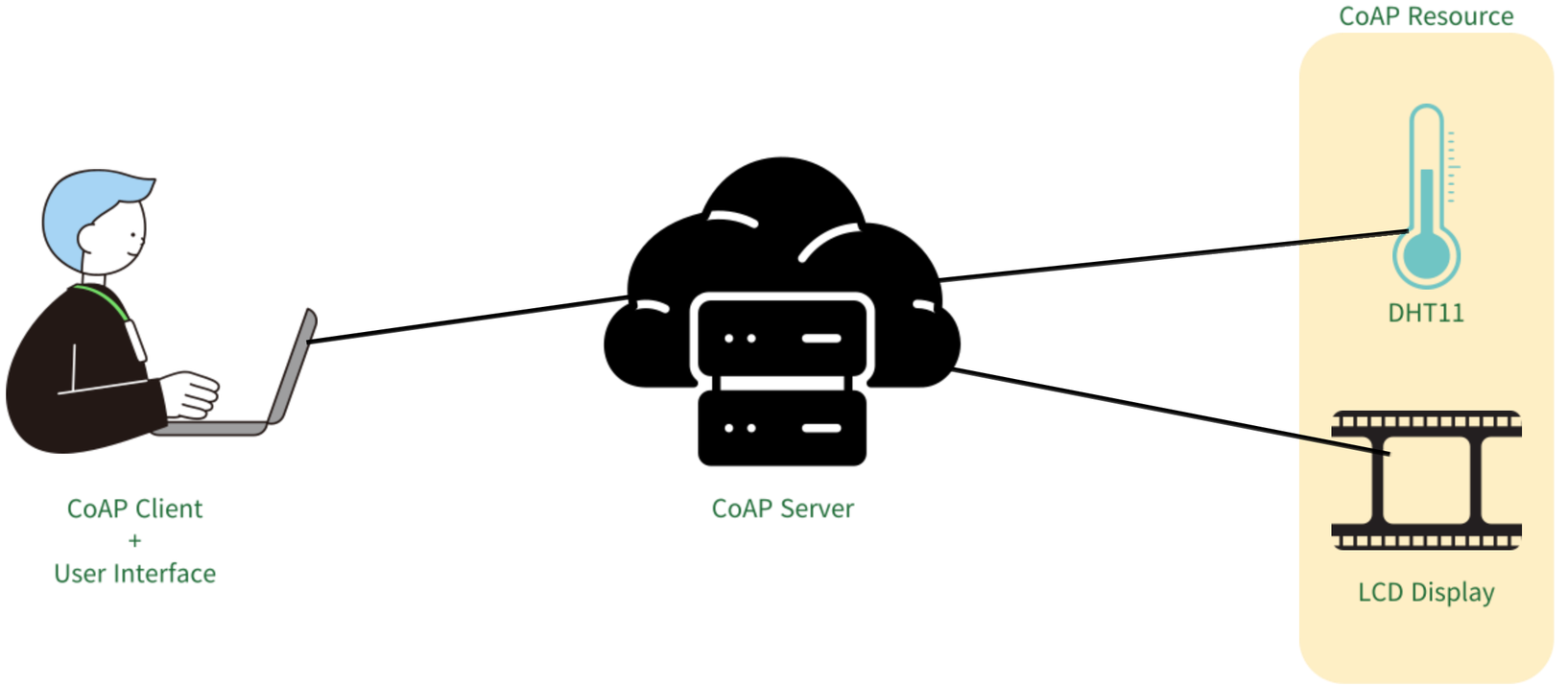
- 영화는 문학, 공연, 미술, 음악 등이 종합적으로 작용하는 종합예술
- 영화 한 편은 약 2시간 정도이며, 여러 예술을 한 번에 접할 수 있음
- 음악 다음으로 접근성이 낮은 문화

## 영화를 통해 윤택한 삶을 만듦

- 영화를 통해 자아 성찰을 하거나 깨달음을 얻을 수 있음
- 많은 논문에서 영화는 심리 치료에도 효과적이라는 결과 도출
- 바쁜 현대인에게 윤택한 삶을 위한 효율적 기회를 제공

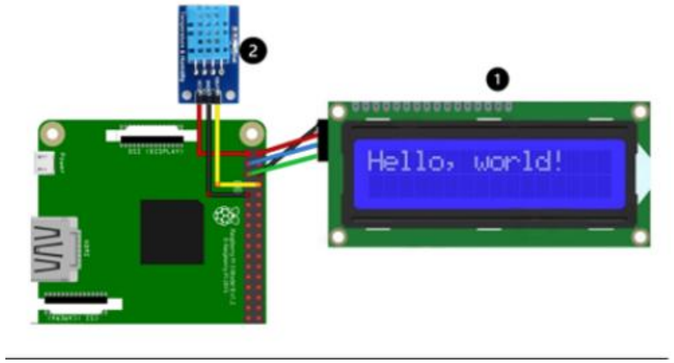
# 시스템 구조

- DHT11 : 온도와 상대 습도 감지, Observe Option을 통한 주기적인 모니터링
- LCD Display : Client에게 추천 영화 출력
- CoAP Server : CoAP 통신을 사용한 Resource 관리
- CoAP Client/UI : 현재 감정 입력 및 DHT11의 모니터링 값에 따른추천 영화를 제공 받음

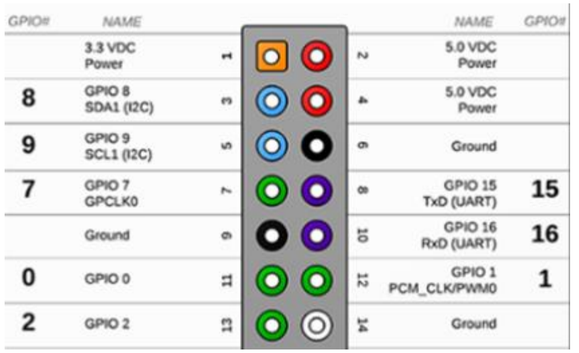


# 하드웨어 구현

- Raspberry Pi : Raspberry Pi 3 - Model B+
- DHT11 : DOUT (GPIO 15), Ground, VCC (3.3 VDC)
- LCD Display : Ground, VCC (5.0 VDC), I2C SDA (GPIO 8), I2C SCL (GPIO 9)




회로도

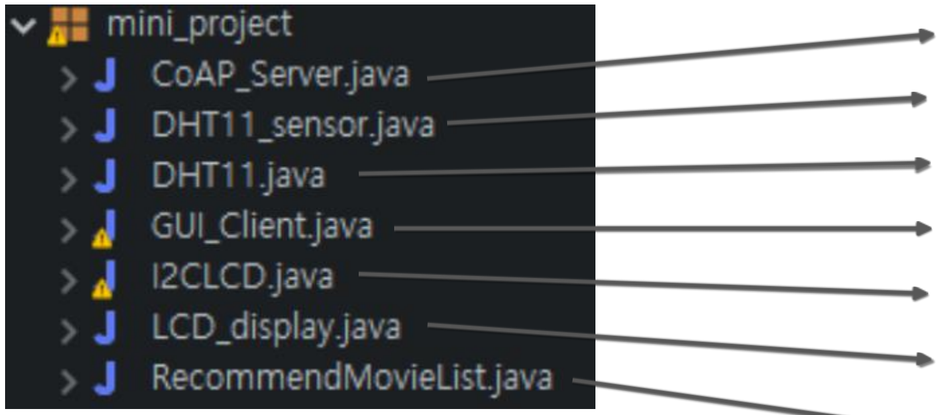



GPIO Pin Map

실제 구현 사진

# 소프트웨어 구현

- Java 기반으로 개발
- 아래의 소스 코드를 포함




CoAP Server

DHT11 센서에서 받은 값은 대기 상태로 변환, CoAP Resource

DHT11 센서

CoAP Client

I2C 통신을 통한 LCD 모듈 사용 라이브러리 (실습에서 제공)

LCD 모듈, CoAP Resource

영화 리스트와 감정 및 대기 상태에 따른 영화 추천

# 소프트웨어 구현

- DHT11.java
  - Data 교환 시 Time out 설정
  - DHT11 센서의 Data Format 생성
  - 특정 GPIO Pin과 연결된 DHT11 센서의 값을 가져옴
    - Checksum 확인
    - {Humidity, Temperature (°C), Temperature(°F)} 배열을 반환

※ 소스 코드 주석 참고

```
public class DHT11 {
    private static final int MAXTIMINGS = 85;   // Data 교환이 이루어질 수 있는 최대 경과 시간 정의
    private final int[] dht11_f = { 0, 0, 0, 0, 0 };    // DHT11 data format (5 bytes)

    /* DHT11 생성자 */
    public DHT11() {
        // Setup wiringPi
        if (Gpio.wiringPiSetup() == -1) {
            System.out.println(" ==>> GPIO SETUP FAILED");
            return;
        }
    }
```

DHT11.java 소스 코드 (1)

```
/* 매개변수 pin 값에 따라 해당 GPIO pin에 연결된 습습도 센서의 데이터를 가져옴 */
public float[] getData(final int pin) {
    int laststate = Gpio.HIGH;  // signal 상태 변화를 알기 위해 기존 상태를 기억
    int j = 0;  // 수신한 Bit의 index counter
    float h = -99;  // 습도
    float c = -99;  // 섭씨 온도
    float f = -99;  // 화씨 온도

    // Integral RH, Decimal RH, Integral T, Decimal
    dht11_f[0] = dht11_f[1] = dht11_f[2] = dht11_f[3] = dht11_f[4] = 0;

    // 1. DHT11 센서에게 start signal 전달
    Gpio.pinMode(pin, Gpio.HIGH);
    Gpio.digitalWrite(pin, Gpio.LOW);
    Gpio.delay(18); // 18 ms

    // 2. Pull-up -> 수신 모드로 전환 -> 센서의 응답 대기
    Gpio.digitalWrite(pin, Gpio.HIGH);
    Gpio.pinMode(pin, Gpio.INPUT);

    // 3. 센서의 응답에 따른 동작
    for (int i = 0; i < MAXTIMINGS; i++) {
        int counter = 0;

        while (Gpio.digitalRead(pin) == laststate) {    // Gpio pin 상태가 바뀌지 않으면 대기
            counter++;
            Gpio.delayMicroseconds(1);
            if (counter == 255) {
                break;
            }
        }

        laststate = Gpio.digitalRead(pin);
        if (counter == 255) {
            break;
        }

        // 각각의 bit 데이터 저장
        if (i >= 4 && i % 2 == 0) { // 첫 3개의 상태 변화는 무시, last state가 low에서 high로 바뀔 때만 값을 저장
            //data 저장
            dht11_f[j / 8] <<= 1;   // 0 bit
            if (counter > 16) {
                dht11_f[j / 8] |= 1;    // 1 bit
            }

            j++;
        }
    }

    /* Checksum 확인
    Check we read 40 bits (8 bit x 5) + verify checksum in the last */
    if (j >= 40 && getChecksum()) {
        h = (float) ((dht11_f[0] << 8) + dht11_f[1]) / 10;
        if (h > 100) {
            h = dht11_f[0]; // for DHT11
        }

        c = (float) (((dht11_f[2] & 0x7F) << 8) + dht11_f[3]) / 10;
        if (c > 125) {
            c = dht11_f[2]; // for DHT11
        }
        if ((dht11_f[2] & 0x80) != 0) {
            c = -c;
        }

        f = c * 1.8f + 32;
        System.out.println("Humidity = " + h + "% Temperature = " + c + "°C | " + f + "°F");
    }
    else {
        System.out.println("Checksum Error");
    }

    float[] result = {h, c, f};
    return result;
}

/* Checksum 값이 맞는지 확인 */
private boolean getChecksum() {
    return dht11_f[4] == (dht11_f[0] + dht11_f[1] + dht11_f[2] + dht11_f[3] & 0xFF);
}
```

DHT11.java 소스 코드 (2)

# 소프트웨어 구현

- DHT11_sensor.java
  - 생성자를 통해 URI 경로를 "/dht"로 설정
  - 온도와 습도 값에 따라 DHT11의 값을 대기 상태(TEXT) 형태로 만듦
  - GET 메시지 수신 시, DHT11 센서 값을 대기 상태 값으로 변경 후 메시지 반환
  - 대기 상태 값이 변경될 때만 메시지 반환 (Optional_Changed)

※ 소스 코드 주석 참고

```java
public class DHT11_sensor extends BasicCoapResource{    // CoAp 통신을 위해 BasicCoapResource 클래스를 상속 받음

    private String state = "";  // 대기 상태
    DHT11 dht = new DHT11();    // 온습도 정적 객체 생성

    /* DHT11_sensor 생성자 */
    private DHT11_sensor(String path, String value, CoapMediaType mediaType) {
        super(path, value, mediaType);
    }

    /* DHT11_sensor 생성자 */
    public DHT11_sensor() {
        this("/dht", "", CoapMediaType.text_plain); // URI, state의 초기값, CoAp Media Type 설정
    }


    /* GET 메시지 수신 시 */
    @Override
    public synchronized CoapData get(List<String> query, List<CoapMediaType> mediaTypesAccepted) {
        return get(mediaTypesAccepted);
    }

    /* GET 메시지 수신 시 */
    @Override
    public synchronized CoapData get(List<CoapMediaType> mediaTypesAccepted) {
        float[] sensing_data = dht.getData(15); // GPIO 15번 Pin에 연결된 온습도 센서의 데이터를 가져옴
        this.state = figureToState(sensing_data);   // DHT11 센서 값은 대기 상태 값으로 변환

        return new CoapData(Encoder.StringToByte(this.state), CoapMediaType.text_plain);    // 대기 상태 값을
    }

    /* state 값이 변경될 때만 서버로 전송 */
    public synchronized void optional_changed() {
        float[] sensing_data = dht.getData(15); // GPIO 15번 Pin에 연결된 온습도 센서의 데이터를 가져옴
        String result = figureToState(sensing_data);    // 센서로 받아온 수치들을 대기 상태(텍스트)로 변환

        // 중복값 비교 후, 중복값 변경 시에만 Observe 응답 전송
        if (result.equals(this.state)) {
            System.out.println("The State has not changed. ");
        }
        else {
            this.state = result;
            this.changed(this.state);

            System.out.println(this.state);
        }
    }
```

생성자, GET, Optional_Changed 동작 정의

```java
/* 온도와 습도 값에 따른 대기 상태를 Text 형태로 변환 */
public String figureToState(float[] data) {
    float humi = data[0];   // 상대 습도
    float temp = data[1];   // 온도 (섭씨)
    String result = ""; // 대기 상태

    // Cheaksum error로 값 변경하지 않기
    if (humi == -99f || temp == -99f) {
        return this.state;
    }

    // 값에 따른 대기 상태 구분
    if (humi > 70f && temp > 23f) {
        result = "고온다습";
    }
    else if (humi > 70f && temp > 7f) {
        result = "온화다습";
    }
    else if (humi > 70f && temp <= 7f) {
        result = "한랭다습";
    }
    else if (humi > 50f && temp > 23f) {
        result = "고온";
    }
    else if (humi > 50f && temp > 7f) {
        result = "온화";
    }
    else if (humi > 50f && temp <= 7f) {
        result = "한랭";
    }
    else if (humi <= 50f && temp > 23f) {
        result = "고온건조";
    }
    else if (humi <= 50f && temp > 7f) {
        result = "온화건조";
    }
    else if (humi <= 50f && temp <= 7f) {
        result = "한랭건조";
    }

    return result;
}
```

DHT11의 값에 따른 대기 상태 변환

| 습도 \ 온도 | 23 ~ | 7 ~ 23 | ~ 7 |
|---|---|---|---|
| 70 ~ 100 | 고온다습 | 온화다습 | 한랭다습 |
| 50 ~ 70 | 고온 | 온화 | 한랭 |
| ~ 50 | 고온건조 | 온화건조 | 한랭건조 |

온습도 값에 따른 대기 상태

# 소프트웨어 구현

- RecommendMovieList.java
  - 각각의 감정과 대기 상태에 어울리는 영화 리스트 (Array's Length = 2)
  - 감정과 대기 상태에 따른 영화 추천
    - ■영화 리스트에서 랜덤으로 하나 선택

※ 소스 코드 주석 참고

```java
public class RecommendMovieList {
    // 대기에 따른 영화 리스트
    private static String[] h_h_list = {"Florida Project", "It"};   // 고온 다습한 대기에 어울리는 영화 리스트
    private static String[] h_m_list = {"Dune", "Oppenheimer"}; // 고온의 대기에 어울리는 영화 리스트
    private static String[] h_l_list = {"Mad Max", "The Hurt Locker"};  // 고온 건조한 대기에 어울리는 영화 리스트
    private static String[] m_h_list = {"Cure", "The Wailing"}; // 온화하고 다습한 대기에 어울리는 영화 리스트
    private static String[] m_m_list = {"Big Fish", "Cinema Paradiso"}; // 온화한 대기에 어울리는 영화 리스트
    private static String[] m_l_list = {"Her", "Late Autumn"};  // 온화하고 건조한 대기에 어울리는 영화 리스트
    private static String[] l_h_list = {"Love Letter", "Eternal Sunshine"}; // 한랭 다습한 대기에 어울리는 영화 리스트
    private static String[] l_m_list = {"Nostalghia", "The Fortress"};  // 한랭한 대기에 어울리는 영화 리스트
    private static String[] l_l_list = {"Doubt", "Bleak Night"};    // 한랭 건조한 대기에 어울리는 영화 리스트

    // 감정에 따른 영화 리스트
    private static String[] p_list = {"WALL-E", "Crayon Shinchan"}; // 기쁠 때 보기 좋은 영화 리스트
    private static String[] s_list = {"Aftersun", "A.I."};  // 슬플 때 보기 좋은 영화 리스트
    private static String[] a_list = {"Bourne Ultimatum", "The Avengers"};  // 화날 때 보기 좋은 영화 리스트
    private static String[] n_list = {"Drive my car", "UP"};    // 불안할 때 보기 좋은 영화 리스트
```

각각의 감정과 대기 상태에 어울리는 영화 리스트

```java
/* 대기 상태에 따른 영화 추천 */
public static String weatRecommend(String state) {
    Random random = new Random();   // 랜덤 객체 생성
    random.setSeed(System.currentTimeMillis());
    int randValue = random.nextInt(2);  // 0과 1 중 랜덤으로 뽑기
    String result = ""; // 랜덤으로 선택된 추천 영화

    if (state.equals("고온다습")) {
        result = h_h_list[randValue];
    }
    else if (state.equals("온화다습")) {
        result = m_h_list[randValue];
    }
    else if (state.equals("한랭다습")) {
        result = l_h_list[randValue];
    }
    else if (state.equals("고온")) {
        result = h_m_list[randValue];
    }
    else if (state.equals("온화")) {
        result = m_m_list[randValue];
    }
    else if (state.equals("한랭")) {
        result = l_m_list[randValue];
    }
    else if (state.equals("고온건조")) {
        result = h_l_list[randValue];
    }
    else if (state.equals("온화건조")) {
        result = m_l_list[randValue];
    }
    else if (state.equals("한랭건조")) {
        result = l_l_list[randValue];
    }

    return result;
}
```

대기 상태에 따른 영화 추천

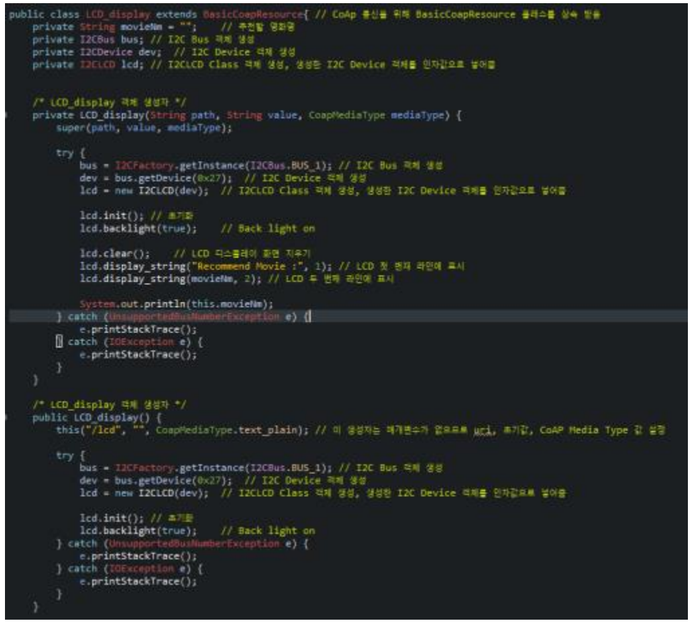
```java
/* 감정에 따른 영화 추천 */
public static String emotionRecommend(String state) {
    Random random = new Random();   // 랜덤 객체 생성
    random.setSeed(System.currentTimeMillis());
    int randValue = random.nextInt(2);  // 0과 1 중 랜덤으로 뽑기
    String result = ""; // 랜덤으로 선택된 추천 영화

    if (state.equals("기쁨")) {
        result = p_list[randValue];
    }
    else if (state.equals("슬픔")) {
        result = s_list[randValue];
    }
    else if (state.equals("화남")) {
        result = a_list[randValue];
    }
    else if (state.equals("불안")) {
        result = n_list[randValue];
    }
    else {  // 잘못 입력하였을 때
        result = "Incorrectly Inputted";
    }

    return result;
}
```

감정에 따른 영화 추천

# 소프트웨어 구현

- LCD_display.java
  - 생성자에서 LCD 모듈을 제어하기 위한 I2C 객체 생성 및 LCD 모듈 초기화
  - 생성자를 통해 URI 경로를 "/lcd"로 설정
  - PUT 메시지 수신 시, LCD의 첫 번째 라인에 "Recommend Movie"를 표시, 두 번째 라인에 수신 받은 메시지의 Payload인 추천 영화명을 표시

※ 소스 코드 주석 참고



```
public class LCD_display extends BasicCoapResource{ // CoAp 통신을 위해 BasicCoapResource 클래스를 상속 받음
    private String movieNm = "";    // 추천할 영화명
    private I2CBus bus; // I2C Bus 객체 생성
    private I2CDevice dev;  // I2C Device 객체 생성
    private I2CLCD lcd; // I2CLCD Class 객체 생성, 생성한 I2C Device 객체를 인자값으로 넘어감


    /* LCD_display 객체 생성자 */
    private LCD_display(String path, String value, CoapMediaType mediaType) {
        super(path, value, mediaType);

        try {
            bus = I2CFactory.getInstance(I2CBus.BUS_1); // I2C Bus 객체 생성
            dev = bus.getDevice(0x27);  // I2C Device 객체 생성
            lcd = new I2CLCD(dev);  // I2CLCD Class 객체 생성, 생성한 I2C Device 객체를 인자값으로 넘어감

            lcd.init(); // 초기화
            lcd.backlight(true);    // Back light on

            lcd.clear();    // LCD 디스플레이 화면 지우기
            lcd.display_string("Recommend Movie :", 1); // LCD 첫 번재 라인에 표시
            lcd.display_string(movieNm, 2); // LCD 두 번째 라인에 표시

            System.out.println(this.movieNm);
        } catch (UnsupportedBusNumberException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /* LCD_display 객체 생성자 */
    public LCD_display() {
        this("/lcd", "", CoapMediaType.text_plain); // 이 생성자는 매개변수가 없으므로 uri, 초기값, CoAP Media Type 값 설정

        try {
            bus = I2CFactory.getInstance(I2CBus.BUS_1); // I2C Bus 객체 생성
            dev = bus.getDevice(0x27);  // I2C Device 객체 생성
            lcd = new I2CLCD(dev);  // I2CLCD Class 객체 생성, 생성한 I2C Device 객체를 인자값으로 넘어감

            lcd.init(); // 초기화
            lcd.backlight(true);    // Back light on
        } catch (UnsupportedBusNumberException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

생성자 및 LCD 모듈 초기화

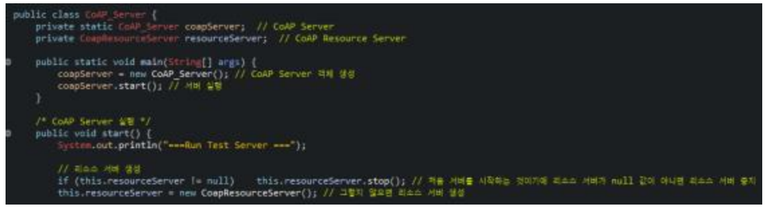
```
    /* LCD Display의 출력하는 값 변경 */
    @Override
    public synchronized boolean setValue(byte[] value) {
        this.movieNm = Encoder.ByteToString(value);

        lcd.clear();    // LCD 디스플레이 화면 지우기
        lcd.display_string("Recommend Movie :", 1); // LCD 첫 번재 라인에 표시
        lcd.display_string(movieNm, 2); // LCD 두 번째 라인에 표시

        System.out.println(this.movieNm);

        return true;
    }

    /* PUT 메시지 수신 시 */
    @Override
    public synchronized boolean put(byte[] data, CoapMediaType type) {
        return this.setValue(data);
    }
```

PUT 메시지 수신 시 LCD Display 출력 동작

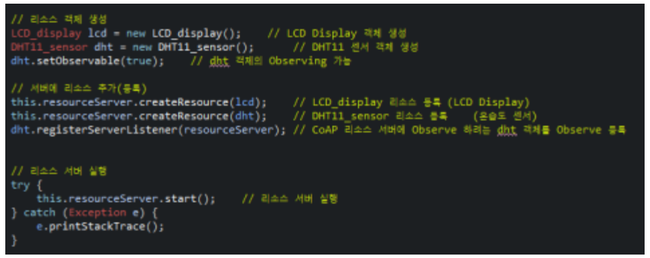
# 소프트웨어 구현

- CoAP_Server.java
  - 리소스 서버 생성 및 실행
  - 서버에 DHT11_sensor와 LCD_display 리소스 추가
  - DHT11_sensor는 Observe 옵션 활성화
    - 10초 간격으로 DHT11_sensor 값을 전송하도록 구현

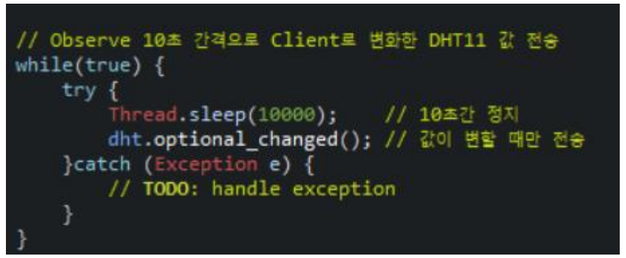
※ 소스 코드 주석 참고




서버 생성 및 실행




리소스 생성 및 등록, 리소스 서버 실행




DHT11_sensor의 Observe 동작 구현

# 소프트웨어 구현

- GUI_Client.java
  - JFrame으로 GUI 구현
  - CoAP Server의 IP 주소를 이용하여 연결
  - Display에 추천 영화 출력
  - DHT11 Observe 버튼, 감정 입력 부분, LCD 모듈로 전송하는 Input 버튼 추가

※ 소스 코드 주석 참고

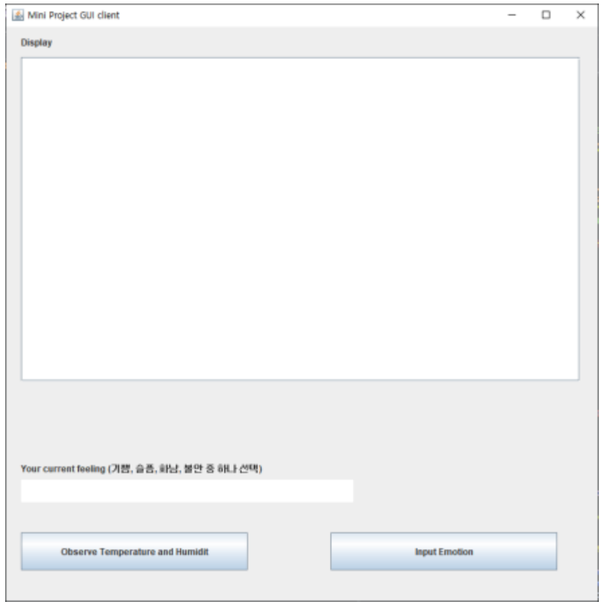
```
public class GUI_Client extends JFrame implements CoapClient{
    // Button 객체 생성
    JButton btn_obs = new JButton("Observe Temperature and Humidit");   // 온습도 값을 Observe 하는 버튼
    JButton btn_input = new JButton("Input Emotion");   // 감정을 입력하면 그 값을 전송하는 버튼

    // Label 및 TextArea 객체 생성
    JLabel payload_label = new JLabel("Your current feeling (기쁨, 슬픔, 화남, 불안 중 하나 선택)");   // 감정 입력의 Label
    JTextArea payload_text = new JTextArea("", 1,1);    // 감정을 입력하는 부분 스크롤바 없음
    JTextArea display_text = new JTextArea();   // 감정과 다기 상태에 따른 추천된 영화를 출력하는 부분
    JScrollPane display_text_jp  = new JScrollPane(display_text);   // 스크롤바 있음
    JLabel display_label = new JLabel("Display");   // Display Label

    CoapClientChannel clientChannel = null;

    /* GUI_Client 생성자 */
    public GUI_Client (String serverAddress, int serverPort) {
        //제목 설정
        super("Mini Project GUI client");

        //레이아웃 설정
        this.setLayout(null);
        String sAddress = serverAddress;    // 서버 IP주소
        int sPort = serverPort; // 서버 포트번호

        // CoAP 통신을 위한 CoapChannelManager 객체 생성
        CoapChannelManager channelManager = BasicCoapChannelManager.getInstance();

        // Client와 Server 연결
        try {
            clientChannel = channelManager.connect(this, InetAddress.getByName(sAddress), sPort);   // client를 서버 IP와 Port에 연결
        } catch (UnknownHostException e) {
            e.printStackTrace();
            System.exit(-1);    // 프로세스 강제 종료
        }

        // clientChannel 설정이 안되어 있으면 종료
        if (null == clientChannel) {
            return;
        }


        // Button 위치 설정
        btn_obs.setBounds(20, 670, 300, 50);
        btn_input.setBounds(430, 670, 300, 50);
```

```
public static void main(String[] args){
    // 프레임 열기
    GUI_Client gui = new GUI_Client("192.168.0.9", CoapConstants.COAP_DEFAULT_PORT);    // 192.168.0.9 IP 주소를 가진 CoAP 서버와 통신
}
```

GUI 구성 요소 생성 및 CoAP Server와 연결

```
// payload_label 위치 및 글꼴 설정
payload_label.setBounds(20, 570, 350, 30);
payload_text.setBounds(20, 600, 440, 30);
payload_text.setFont(new Font("arian", Font.BOLD, 15));

// display_label 위치 및 글꼴 설정
display_label.setBounds(20, 10, 100, 20);
display_text.setLineWrap(true);
display_text.setFont(new Font("arian", Font.BOLD, 15));
display_text_jp.setBounds(20, 40, 740, 430);

// Button 및 Label들을 화면창에 추가
this.add(btn_obs);
this.add(btn_input);
this.add(payload_label);
this.add(payload_text);
this.add(display_text_jp);
this.add(display_label);

// 프레임 크기 지정
this.setSize(800, 800);

// 프레임 보이기
this.setVisible(true);

//swing에만 있는 X버튼 클릭시 종료
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

JFrame에 각 구성 요소 위치 지정 및 추가




GUI_Client 실행 결과 모습

# 소프트웨어 구현

- GUI_Client.java
  - Observe 버튼 클릭 시, Token 값은 "ObToken"으로 설정하고 Sequence Number는 0부터 시작하도록 설정, /dht로 GET Request 메시지 전송
  - Input 버튼 클릭 시, 입력된 감정에 따른 추천 영화를 /lcd로 PUT Request 메시지 전송

※ 소스 코드 주석 참고

```
/* Observe 버튼 클릭 시 이벤트 정의 */
btn_obs.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String path = "/dht";   // URI 경로 (온습도 센서)

        // request 메시지 설정
        CoapRequest request = clientChannel.createRequest(CoapRequestCode.GET, path, true); // /dht 경로로 메소드는 GET, CON 메시지로 요청
        request.setToken(Encoder.StringToByte("ObToken"));  // Observe Token 값 설정
        request.setObserveOption(0);    // Sequence Number 0부터 시작

        clientChannel.sendMessage(request);     // request 메시지 전송

        // display 부분에 해당 텍스트를 추가
        display_text.append(System.lineSeparator());    // 줄 바꿈
        display_text.append("Observe the Conditions of the Atmosphere");
        display_text.append(System.lineSeparator());    // 줄 바꿈
    }
});
```

Observe 버튼 클릭 시 동작 구현

```
/* Input 버튼 클릭 시 이벤트 정의 */
btn_input.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String path = "/lcd";   // URI 경로 (LCD Display 장치)
        String payload = RecommendMovieList.emotionRecommend(payload_text.getText());   // request로 보낼 payload 값 : 입력한 기분에 따른 추천 영화

        // 잘못 입력된 payload 값일 때
        if (payload.equals("Incorrectly Inputted")) {
            display_text.append(System.lineSeparator());    // 줄 바꿈
            display_text.append(payload);   // Display 부분에 잘못 입력되었다는 것을 출력
            display_text.append(System.lineSeparator());    // 줄 바꿈

            return;
        }

        // request 메시지 설정
        CoapRequest request = clientChannel.createRequest(CoapRequestCode.PUT, path, true); // /lcd 경로로 메소드는 PUT, CON 메시지로 요청
        request.setPayload(new CoapData(payload, CoapMediaType.text_plain));    // payload로 보낼 값과 타입 설정

        clientChannel.sendMessage(request);     // request 메시지 전송

        // display 부분에 해당 텍스트를 추가
        display_text.append(System.lineSeparator());    // 줄 바꿈
        display_text.append("Recommended Movie (" + payload_text.getText() + ") : " + payload); // "Recommended Movie (Emotion) : Movie name" 형태로 출력
        display_text.append(System.lineSeparator());    // 줄 바꿈
    }
});
```

Input 버튼 클릭 시 동작 구현

# 소프트웨어 구현

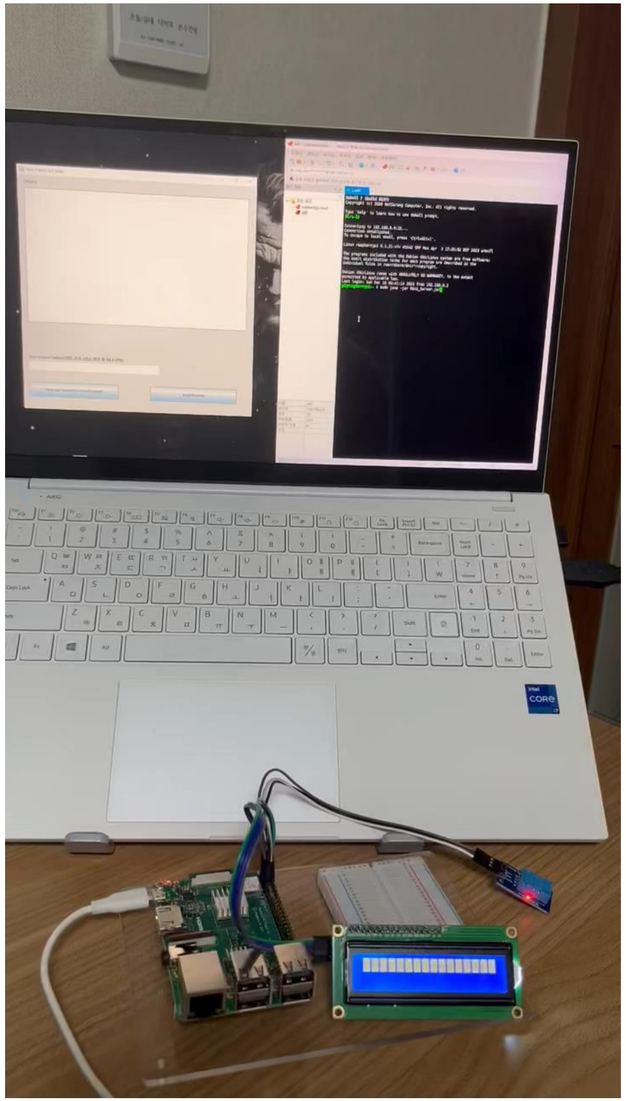
- GUI_Client.java
  - Observe 응답 메시지 수신 시, 수신받은 메시지의 Payload 값을 해당 추천 영화로 변환하여 PUT 메시지로 LCD 모듈로 전송

※ 소스 코드 주석 참고

```java
/* Server에서 온 Observe 응답 메시지 처리 */
@Override
public void onResponse(CoapClientChannel channel, CoapResponse response) {
    if (Encoder.ByteToString(response.getToken()).equals("ObToken")) {  // 응답 메시지의 Token 값이 Observe Token 값인 "ObToken"이면
        controlLCD(Encoder.ByteToString(response.getPayload()));    // 받은 응답 값(대기 상태)에 따른 추천 영화를 LCD Display 센서로 전송
    }
}

/* LCD Display 센서로 전송 */
public void controlLCD(String state) {
    // 대기 상태 값에 오류가 있으면 건너뛰기
    if (state.equals("Error")) {
        return;
    }

    // URI 경로가 /lcd로 put 메소드의 CON 요청
    CoapRequest request = clientChannel.createRequest(CoapRequestCode.PUT, "/lcd", true);   // /lcd 경로로 메소드는 PUT, CON 메시지로 요청
    request.setPayload(new CoapData(RecommendMovieList.weatRecommend(state), CoapMediaType.text_plain));    // payload 값은 대기 상태에 따른 추천 영화

    clientChannel.sendMessage(request);     // request 메시지 전송

    displayRequest(request, state); // 추천 영화와 대기 상태를 표시
}
```

Observe 응답 메시지 수신 시 동작 구현

# Demo Video



# References


1. 손민경. (2022). 감정표현불능증에 대한 영화치료프로그램의 효과: 비전캠프에 참가한 병사들을 대상으로. 인문사회 21, 13(3), 863-876.

2. 임애련. (2021). COVID-19로 인한 비대면 상호작용적 영화심리치료 수업의 효과성 연구: 감성지능과 공감능력. 한국산학기술학회 논문지, 22(2), 57-66

3. 임베디드 시스템 7주차 실습 pdf

4. 임베디드 시스템 4주차 이론 Open Source Hardware pdf

5. The Pi4J Project Raspberry Pi -3B+

6. CoapResourceServer 클래스에 대한 설명

7. CoapChannelManager 클래스에 대한 설명

8. CoapClientChannel 클래스에 대한 설명

# 감사합니다

질문이 있다면 말씀해주세요.