임베디드 시스템

jCoAP Open Source 실습 2

Inhyeok Kang

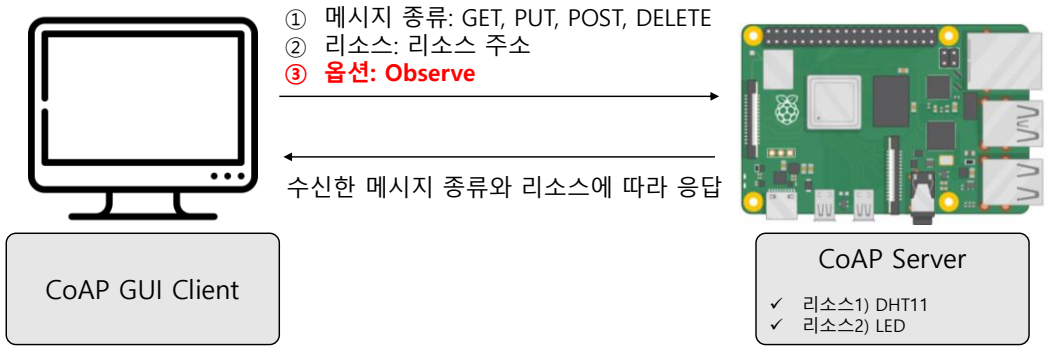
M23522@hallym.ac.kr

연구실: 공학관 1321호

Contents

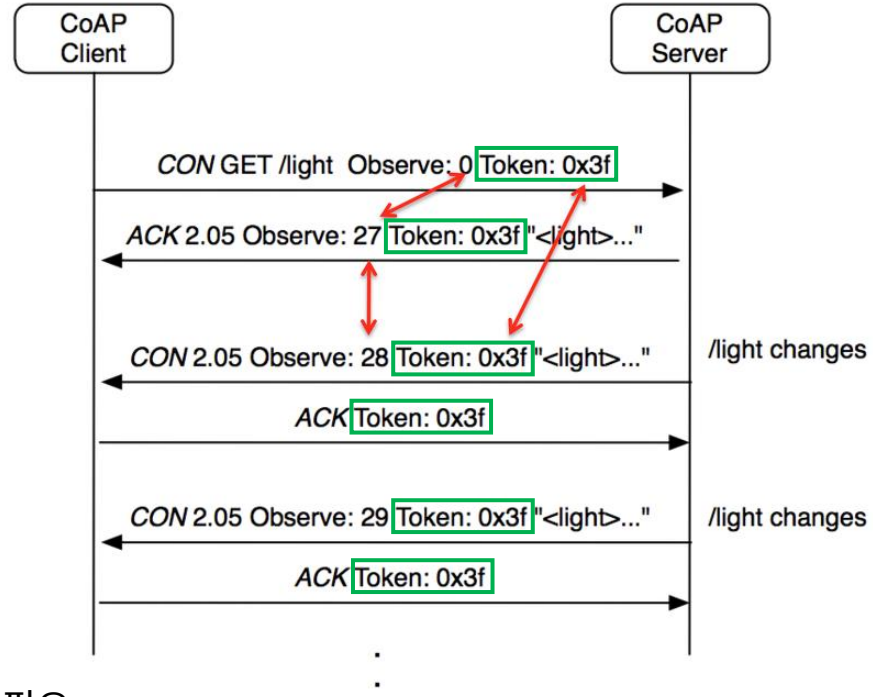
CoAP

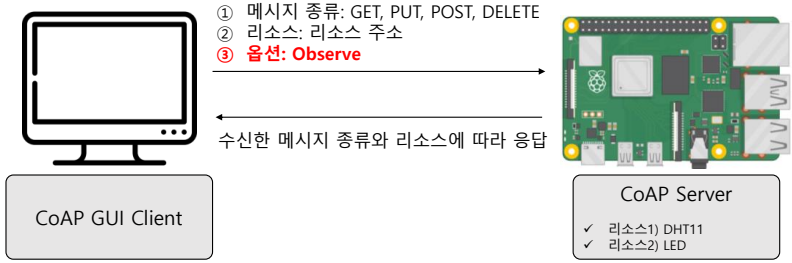
- Constrained Application Protocol (CoAP)
  - CoAP는 사물인터넷과 같이 대역폭이 제한된 통신 환경에 최적화되어 개발된 Representational State Transfer (REST) 기반의 경량 메시지 전송 프로토콜임
  - CPU, 메모리, 통신 Bandwidth 등이 제한된(Constrained) 기기를 위한 Application Protocol
  - CoAP는 RESTful 기반의 프로토콜이므로 기존의 HTTP (Hypertext Transfer Protocol) 웹 프로토콜과의 연동이 쉬움
  - 자원 관리를 위해 HTTP와 동일하게 **GET, PUT, POST, DELETE**의 메소드를 사용함
  - 자원 발견(Resource Discovery), 멀티캐스트 지원, 비동기 트랜잭션 요청 및 응답 등을 지원함
  - **Sever – Client 구조**



CoAP

- CoAP(Constrained Application Protocol)
  - 리소스/자원(Resource)
    - ✓ 센서, 액추에이터, 사용자 정보 등 CoAP Client가 사용할 수 있는 자원임
    - ✓ 일반적으로 각각의 센서 또는 액추에이터가 하나의 리소스로 정의됨
    - ✓ 사용자 이름, 서버의 상태 등 다양한 형태의 리소스가 존재할 수 있음
  - Observe option
    - ✓ 리소스의 상태가 변경될 때마다 알림을 받고자 할 때 사용
    - ✓ Observer가 여러 리소스에 관심이 있는 경우, 모든 리소스에 대해 개별적인 등록 필요





Request

- CoAP Server의 리소스 조회 → GET 요청
- CoAP Server의 리소스 변경 → PUT 요청
- Observe GET (주기적으로 리소스 값 수신)

Response

- 다수의 리소스 관리
- CoAP Client의 요청에 따른 응답 수행
- Observe를 요청한 CoAP Client에게 주기적으로 리소스 값 전송

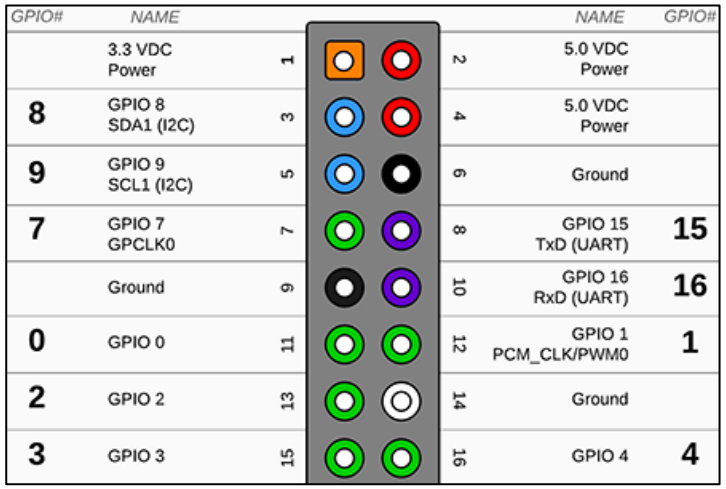
DHT11 및 LED 연결

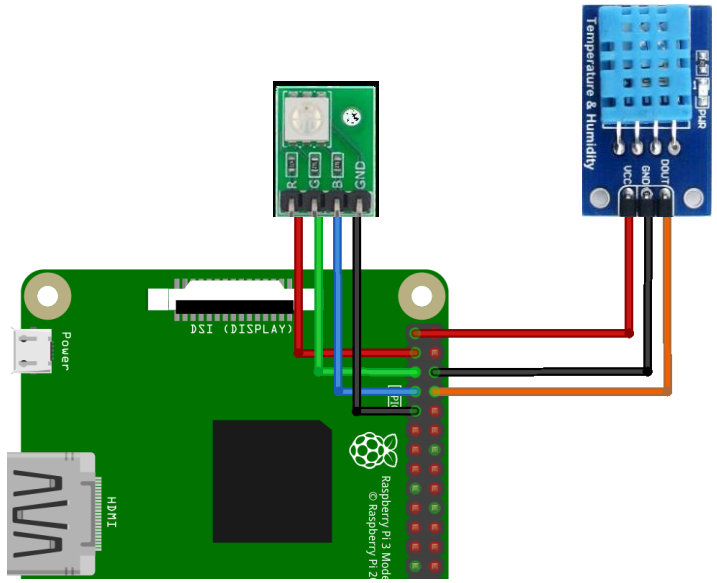
- 2. 구성품 연결
  - **[LED 센서]**
    - ✓ **[점프 와이어(F/F)]**로 연결

LED 센서 모듈	GPIO Pins
R	8
G	9
B	7
VCC	Ground

  - **[DHT11 온습도 센서]**
    - ✓ **[점프 와이어(F/F)]**로 연결

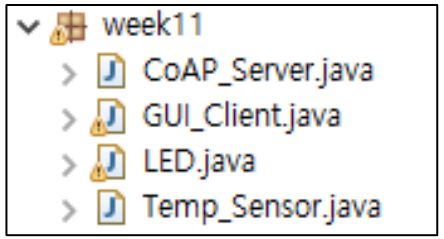
DHT11 온습도 센서	GPIO Pins
DOUT	15
GND	Ground
VCC	3.3 VDC





CoAP Server & Client 실행 및 결과

- 1. CoAP_Server.java 소스 코드
  - (a): CoapResourceServer에 observe하려는 resource (temp_sensor)를 등록함
    - ✓ registerServerListener 메서드 사용
      - ✓ CoapResourceServer 객체에 특정 리소스를 등록하여 해당 리소스에 대해 서버가 이벤트를 감지하고 처리할 수 있도록 함



CoAP_Server.java → 라즈베리 파이에서 실행

GUI_Client.java → PC에서 실행

LED.java → CoAP_Server.java에서 객체로 사용

Temp_Sesnor.java → CoAP_Server.java에서 객체로 사용

```java
package week11;

import org.ws4d.coap.core.rest.CoapResourceServer;

public class CoAP_Server {
    private static CoAP_Server coapServer;
    private CoapResourceServer resourceServer;

    public static void main(String[] args) {
        coapServer = new CoAP_Server();
        coapServer.start();
    }

    public void start() {
        System.out.println("===Run CoAP Server ===");
        // create server
        if (this.resourceServer != null)    this.resourceServer.stop();
        this.resourceServer = new CoapResourceServer();

        // initialize resource
        LED led = new LED();
        Temp_Sensor temp_sensor = new Temp_Sensor();
        // CoapResourceServer에 observe하려는 resource 등록
```

(a)

CoAP Server & Client 실행 및 결과

- 1. CoAP_Server.java 소스 코드
  - Observe option 구현
    - ✓ 주기적으로 resource 값을 전송함

```
        // add resource to server
        this.resourceServer.createResource(temp_sensor);
        this.resourceServer.createResource(led);

        // run the server
        try {
            this.resourceServer.start();
        } catch (Exception e) {
            e.printStackTrace();
        }

        while (true) {
            try {
                Thread.sleep(5000); // observe 주기
                temp_sensor.changed(); // resource의 변화를 알리기 위해 사용됨
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

CoAP Server & Client 실행 및 결과

- 2. GUI_Client.java 소스 코드
  - (b): Observe button 생성
    - ✓ JButton 활용

```
package week11;

import java.awt.FlowLayout;

public class GUI_Client extends JFrame implements CoapClient{
    private static final boolean exitAfterResponse = false;
    JButton btn_get = new JButton("GET");
    JButton btn_post = new JButton("POST");
    JButton btn_put = new JButton("PUT");
    JButton btn_delete = new JButton("DELETE");

                    (b)

    JLabel path_label = new JLabel("Path");
    JTextArea path_text = new JTextArea("/.well-known/core", 1,1);//스크롤바 없음
    JLabel payload_label = new JLabel("Payload");
    JTextArea payload_text = new JTextArea("", 1,1);//스크롤바 없음
    JTextArea display_text = new JTextArea();
    JScrollPane display_text_jp  = new JScrollPane(display_text);
    JLabel display_label = new JLabel("Display");

    CoapClientChannel clientChannel = null;
```

CoAP Server & Client 실행 및 결과

- 2. GUI_Client.java 소스 코드

```
//btn
btn_get.setBounds(20, 670, 100, 50);
btn_put.setBounds(130, 670, 100, 50);
btn_post.setBounds(240, 670, 100, 50);
btn_delete.setBounds(350, 670, 100, 50);
btn_observe.setBounds(460, 670, 100, 50);

btn_get.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        String path = path_text.getText();
        String payload = payload_text.getText();
        CoapRequest request = clientChannel.createRequest(CoapRequestCode.GET, path, true);
        displayRequest(request);
        clientChannel.sendMessage(request);
    }
});
```

Observe button
좌표 및 크기 설정

CoAP Server & Client 실행 및 결과

- 2. GUI_Client.java 소스 코드
  - (c): Observe button에 Action Listener 추가
    - ✓ 기존에 구현되어 있는 get/put/post/delete button의 addActionListener를 참고하여 구현

```
btn_delete.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        String path = path_text.getText();
        String payload = payload_text.getText();
        CoapRequest request = clientChannel.createRequest(CoapRequestCode.DELETE, path, true);
        displayRequest(request);
        clientChannel.sendMessage(request);
    }
});
```

(c)

CoAP Server & Client 실행 및 결과

- 2. GUI_Client.java 소스 코드

```
this.add(btn_get);
this.add(btn_post);
this.add(btn_put);
this.add(btn_delete);
this.add(btn_observe);
this.add(path_text);
this.add(path_label);
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);
```

GUI에 Observe button 추가

CoAP Server & Client 실행 및 결과

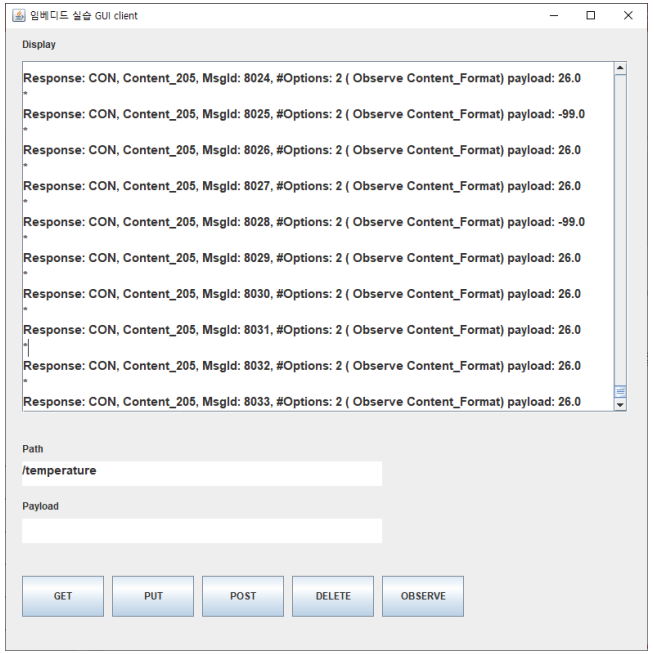
- 3. JAR 파일 생성 후 XFTP를 통해 Raspberry Pi로 전송
- 4. Raspberry Pi에서 JAR 파일 실행(CoAP Server 실행)
  - **sudo java -jar server.jar**

```
pi@raspberrypi:~ $ sudo java -jar Server.jar
===Run Test Server ===
ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console. Se
t system property 'log4j2.debug' to show Log4j2 internal initialization logging.
```

CoAP Server & Client 실행 및 결과

- 5. Eclipse에서 GUI_Client 실행(CoAP Client 실행)
  - Path: /.well-known/core 입력 → GET 버튼 클릭 → Display 창에서 등록된 리소스 확인
  - Path: /temperature 입력 → OBSERVE 버튼 클릭 → Display 창에서 주기적인 온도 값 수신 확인

CoAP Client 동작하지 않을 시, CoAP Client 재실행
(CoAP Server보다 Client를 먼저 실행했기 때문)



감사합니다

Thank You