임베디드 시스템

접근 객체 인식 기반 신호등 시스템

20205236 빅데이터전공 이채현

배경

- 보행 신호에도 불구하고 횡단보도를 지나가는 차량
- 교통사고 사망자 중 보행자 933명으로 전체 사망자의 34.1% → OECD 평균과 비교해 1.9배 (국토교통부)
- https://news.jtbc.co.kr/article/article.aspx?news_id=NB12151771

[르포+] 이름만 보행자 우선도로?…차 '쌩쌩' 위협받는 보행자 안전

[JTBC] 입력 2023-11-11 09:06 | 수정 2023-11-12 10:24

안내 › JTBC 뉴스는 여러분의 생생한 제보를 기다리고 있습니다.

9일 강남역 11번출구 뒤쪽 골목. 보행자 우선도로지만, 차와 뒤섞여 보행자들은 겨우 길을 지나가고 있다. 〈사진=이지현 기자〉

배경 및 제안 아이디어

배경 — **보행자 우선도로**에서도 **안전을 위협**받는 보행자들

기존 방법 — 보행자가 **손을 들어** 차량에 **건너겠다는 의지 표현**

한계점 — 보행자의 걸음 속도를 **기다려주지 않고 차량이 지나감**

제안 아이디어

- **차량 신호가 초록불로 유지**하다 보행자가 접근하면 **보행자를 인식**하고 차량 신호 변화를 위한 **보행자 스위치 활성화** 및 **차량 신호가 변화**
- 차량 신호를 제어함으로써 신호위반 단속 및 보행자의 안전 보장

시스템 구조

PIR Sensor

- 접근 객체 감지
- Observe Option을 통한 주기적인 모니터링 활성화

Switch

- 버튼 눌림 시 차량 신호 변화

LCD Display Module

- 차량 신호 상태에 따라 문구 노출

LED Sensor Module

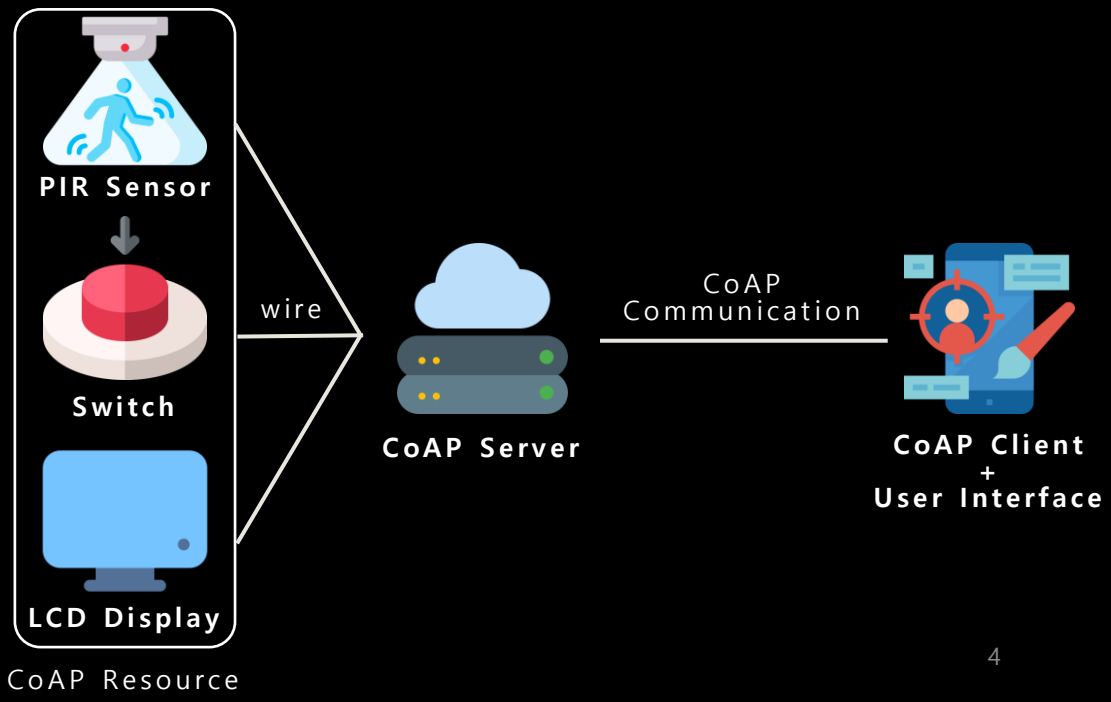
- 차량 신호 표현

CoAP Server

- CoAP 통신을 사용한 Resource 관리

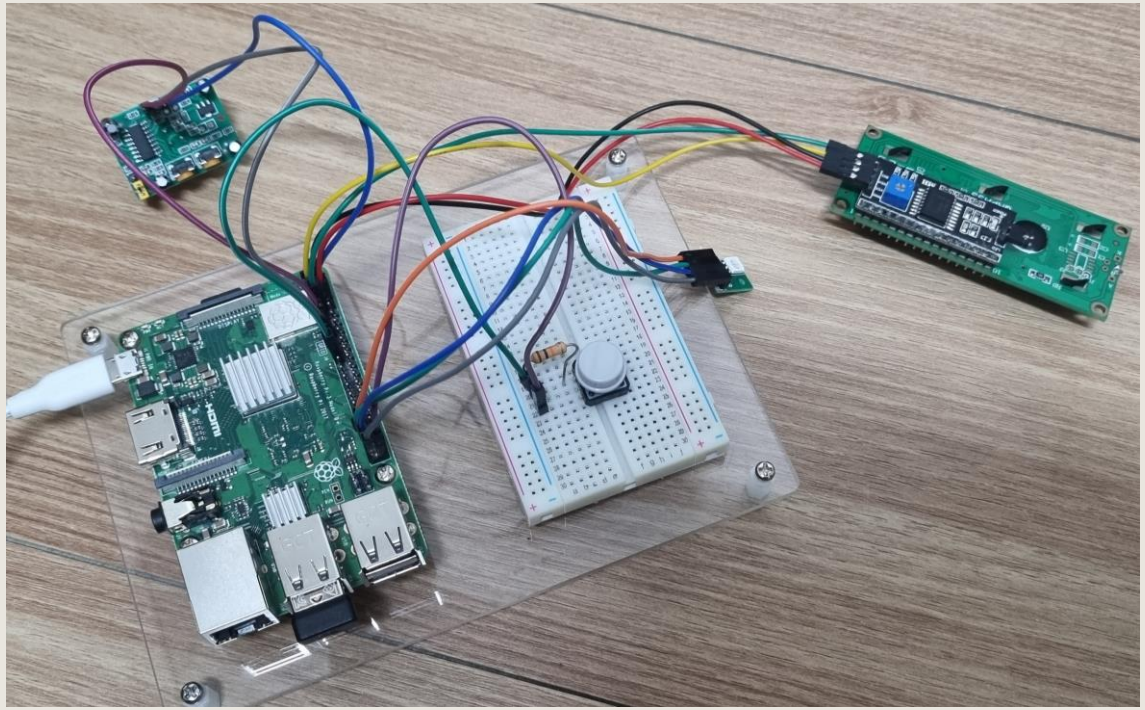
CoAP Client/UI

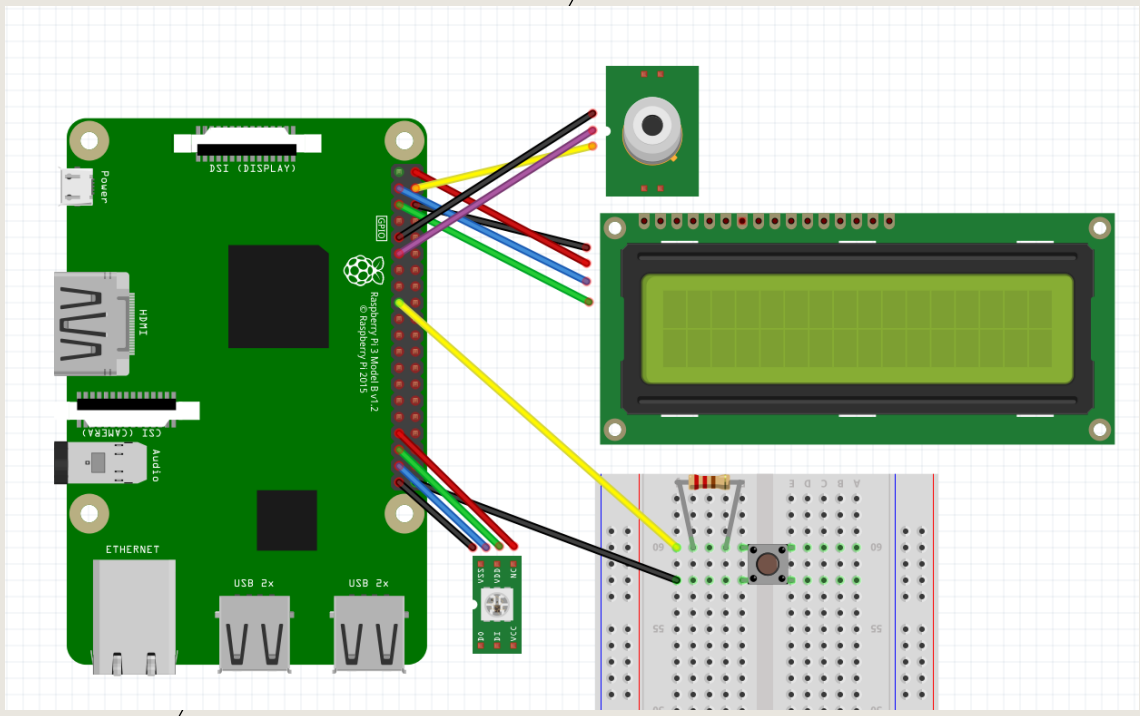
- Client와 UI 통합
- 차량 신호 상태 모니터링



하드웨어

- **PIR Sensor** : Ground, GPIO 00, 5.0 VDC
- **LCD Display Module** : Ground, 5.0 VDC, I2C SDA(GPIO 08), I2C SCL(GPIO 09)
- **LED Sensor Module** : Ground, R(GPIO 23), G(GPIO 24), B(GPIO 25)
- **Switch** : 막대저항(10KΩ), GPIO 29, 3.3 VDC





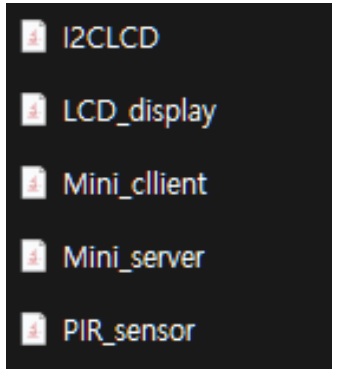
소프트웨어

JAVA

- 자바 기반 구현

다음 소스코드 포함

- **I2CLCD** : LCD 모듈 사용 라이브러리
- **LCD_display** : LCD 모듈 CoAP Resource
- **Mini_client** : CoAP Client
- **Mini_server** : CoAP Server
- **PIR_sensor** : PIR 센서 CoAP Resource



소프트웨어

- LCD_display

```
public class LCD_display extends BasicCoapResource{
    private String msg = "null";
    I2CDevice _device = null;
    I2CLCD _lcd = null;

    private LCD_display(String path, String value, CoapMediaType mediaType)
        super(path, value, mediaType);
        try {
            I2CBus bus = I2CFactory.getInstance(I2CBus.BUS_1);
            _device = bus.getDevice(0x27);
            _lcd = new I2CLCD(_device);
            _lcd.init();
            _lcd.backlight(true);
            _lcd.clear();
            _lcd.display_string("Press The Button", 1);
            _lcd.display_string(value, 1);
        } catch (Exception ex) {
            System.out.println(ex.toString());
        }
    }
```

LCD 모듈 초기화

- 생성자에서 LCD 모듈을 사용하기 위한 라이브러리 객체 선언

```
@Override
public synchronized boolean setValue(byte[] value) {
    this.msg = Encoder.ByteToString(value);
    _lcd.clear();
    _lcd.display_string("Press The Button", 1);
    _lcd.clear();
    _lcd.display_string(msg, 1);
    return true;
}
```

PUT 메시지 수신 시 동작 정의

- 수신한 메시지의 Payload를 LCD의 1번째 Line에 표시

소프트웨어

- PIR_sensor

```
public class PIR_sensor extends BasicCoapResource{
    private String state = "Green Light";

    GpioController gpio = GpioFactory.getInstance();
    // PIR Sensor
    GpioPinDigitalInput pir = gpio.provisionDigitalInputPin(RaspiPin.GPIO_00);
    // Button
    GpioPinDigitalInput btn = gpio.provisionDigitalInputPin(RaspiPin.GPIO_29);
    // LED
    GpioPinDigitalOutput r_led = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_23, PinState.LOW);
    GpioPinDigitalOutput g_led = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_24, PinState.LOW);
    GpioPinDigitalOutput b_led = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_25, PinState.LOW);
```

클래스 변수로 pi4j 객체 선언

- 클래스 변수로 PIR sensor 값, Switch 값, LED sensor 값을 읽어올 수 있는 pi4j 객체 선언

```
public void SetLED() {

    if (this.state.equals("Red Light")) {
        r_led.high();
        g_led.low();
        b_led.low();
    } else if (this.state.equals("Green Light")) {
        r_led.low();
        g_led.high();
        b_led.low();
    } else if (this.state.equals("Yellow Light")) {
        r_led.high();
        g_led.high();
        b_led.low();
    }
}
```

LED sensor 동작

- 상태에 따라 LED sensor의 색상 변화

소프트웨어

- PIR_sensor

```java
@Override
public synchronized CoapData get(List<CoapMediaType> mediaTypesAccepted) {
    boolean pir_state = pir.isHigh();   // PIR 센서 감지를 확인하는 변수
    boolean btn_pressed = btn.isHigh(); // 버튼 눌림을 확인하는 변수

    if (pir_state == true) {
        System.out.println("Person O");
        if (btn_pressed == true) {
            try {
                // Blue Light --> 3 Sec
                this.state = "Yellow Light";
                this.changed(this.state);
                SetLED();
                Thread.sleep(3000);

                // Red Light --> 5 Sec
                this.state = "Red Light";
                this.changed(this.state);
                SetLED();
                Thread.sleep(5000);

                this.state = "Green Light";
                this.changed(this.state);
                SetLED();
            } catch (Exception e) {

            }
        }
    }
    else {
        System.out.println("Person X");
        this.state = "Green Light";
        this.changed(this.state);
        SetLED();
    }

    return new CoapData(Encoder.StringToByte(this.state), CoapMediaType.text_plain);
}
```

GET 메시지 수신 시 동작 정의

- PIR sensor를 통해 보행자를 인식하고, Switch를 통해 LED 변화
- 보행자가 인식되고 Switch가 눌리면, Yellow(3초) → Red(5초) → Green으로 LED 색상 변화
- 결과를 Payload에 표시하여 응답

소프트웨어

- Mini_server

```
// initialize resource
LCD_display lcd = new LCD_display();
PIR_sensor pir = new PIR_sensor();

pir.setObservable(true);

// add resource to server
this.resourceServer.createResource(lcd);
this.resourceServer.createResource(pir);

pir.registerServerListener(resourceServer);
```

Resource 추가 및 PIR sensor

Observe 옵션 활성화

- Server에 PIR senso와 LCD display Resource 추가
- PIR sensor Observe 옵션 활성화

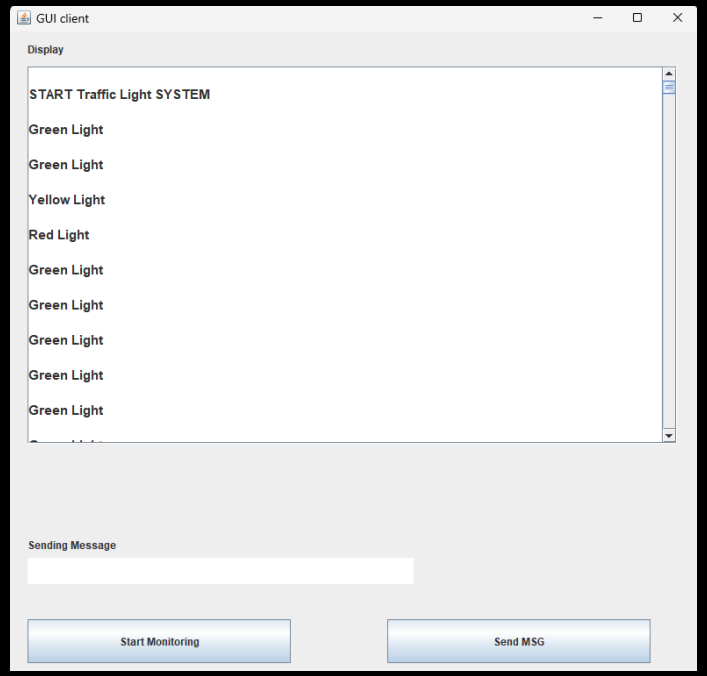
```
// Sensor values are delivered to the observed client once per second
while(true) {
    try {
        Thread.sleep(1000);
    }catch (Exception e) {
        // TODO: handle exception
    }
    pir.changed();
}
```

PIR sensor의 Observe 동작 구현

- 1초에 1번씩 PIR sensor 값 전송

소프트웨어

- Mini_client



모니터링을 위한 UI

- Start Monitoring을 클릭하면 현재 차량 신호의 상태를 1초마다 출력

```
public void chainging_Light(String msg) {
    if(msg.equals("Yellow Light")) {
        display_text.append(System.lineSeparator());
        display_text.append("Yellow Light");
        display_text.append(System.lineSeparator());
        CoapRequest request = clientChannel.createRequest(CoapRequestCode.PUT, "/lcd", true);
        request.setPayload(new CoapData("Please Wait", CoapMediaType.text_plain));
        displayRequest(request);
        clientChannel.sendMessage(request);
    }
    else if (msg.equals("Red Light")){
        display_text.append(System.lineSeparator());
        display_text.append("Red Light");
        display_text.append(System.lineSeparator());
        CoapRequest request = clientChannel.createRequest(CoapRequestCode.PUT, "/lcd", true);
        request.setPayload(new CoapData("Please Cross", CoapMediaType.text_plain));
        displayRequest(request);
        clientChannel.sendMessage(request);
    }
    else {
        display_text.append(System.lineSeparator());
        display_text.append("Green Light");
        display_text.append(System.lineSeparator());
        CoapRequest request = clientChannel.createRequest(CoapRequestCode.PUT, "/lcd", true);
        request.setPayload(new CoapData("Press The Button", CoapMediaType.text_plain));
        displayRequest(request);
        clientChannel.sendMessage(request);
    }
}
```

차량 신호 상태

- 차량 신호 상태를 출력
- 차량 신호 상태에 따라 보행자를 위한 LCD 출력
- **Green** : Press the Button / **Yellow** : Please Wait / **Red** : Please Cross

소프트웨어

- Mini_client

```
public class LCD_display extends BasicCoapResource{
    private String msg = "null";
    I2CDevice _device = null;
    I2CLCD _lcd = null;

    private LCD_display(String path, String value, CoapMediaType mediaType)
        super(path, value, mediaType);
        try {
            I2CBus bus = I2CFactory.getInstance(I2CBus.BUS_1);
            _device = bus.getDevice(0x27);
            _lcd = new I2CLCD(_device);
            _lcd.init();
            _lcd.backlight(true);
            _lcd.clear();
            _lcd.display_string("Press The Button", 1);
            _lcd.display_string(value, 1);
        } catch (Exception ex) {
            System.out.println(ex.toString());
        }
    }
```

모니터링을 위한 UI

- 생성자에서 LCD 모듈을 사용하기 위한 라이브러리 객체 선언

```
@Override
public synchronized boolean setValue(byte[] value) {
    this.msg = Encoder.ByteToString(value);
    _lcd.clear();
    _lcd.display_string("Press The Button", 1);
    _lcd.clear();
    _lcd.display_string(msg, 1);
    return true;
}
```

PUT 메시지 수신 시 동작 정의

- 수신한 메시지의 Payload를 LCD의 1번째 Line에 표시

DEMO 영상

- https://youtu.be/8BoGLeq0kkE

- 보행자가 없는 경우 차량 신호 초록불 유지 및 LCD에 Press The Button 출력
- PIR Sensor로 보행자 인식
- 보행자가 인식될 경우에만 Switch 활성화
- Switch를 누르면 차량 신호 변화 : 초록불 및 LCD에 Press the Button→ 노란불(3초) 및 LCD에 Please Wait → 빨간불(5초) 및 LCD에 Please Across → 초록불 및 LCD에 Press the Button

REFERENCE

- Flaticon
- Fritzing
- JTBC

감사합니다.