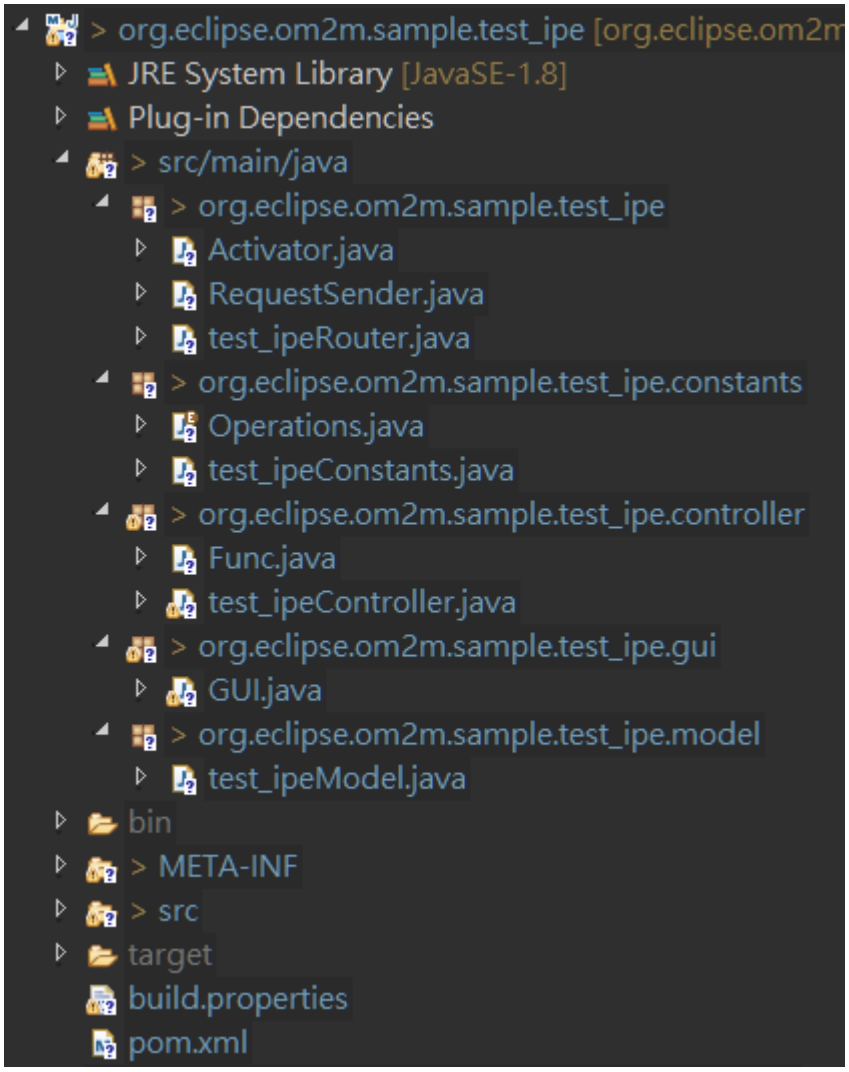
物聯網核心網路技術 **LAB5**

姓名：陳維倫　　學號：**N16094409**　系所:機械碩一

目標:在 mn-cse 建立一個具有 GUI 功能的 IPE，可以讓使用者調節冷氣溫度、風速和開關，當冷氣關閉時就不能調整溫度及風速，其中風速限制為 1~n (n 可以自己設定)，溫度不限。

步驟解釋:

先使用 Eclipse 建立 IPE Bundle，並建立 Packages 及 Files 如下圖



Activate.java

```
package org.eclipse.om2m.sample.test_ipe;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.eclipse.om2m.core.service.CseService;
import org.eclipse.om2m.interworking.service.InterworkingService;
import org.eclipse.om2m.sample.test_ipe.constants.test_ipeConstants;
import org.eclipse.om2m.sample.test_ipe.controller.Func;
import org.eclipse.om2m.sample.test_ipe.controller.test_ipeController;
import org.eclipse.om2m.sample.test_ipe.gui.GUI;
```

```
import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;
import org.osgi.framework.ServiceReference;
import org.osgi.util.tracker.ServiceTracker;

public class Activator implements BundleActivator {
    /** Logger */
    private static Log logger = LogFactory.getLog(Activator.class);
    /** SCL service tracker */
    private ServiceTracker<Object, Object> cseServiceTracker;

    public void start(BundleContext bundleContext) throws Exception {
        logger.info("Register IpeService..");

    bundleContext.registerService(InterworkingService.class.getName(),
new test_ipeRouter(), null);
        logger.info("IpeService is registered.");
        cseServiceTracker = new ServiceTracker<Object,
Object>(bundleContext, CseService.class.getName(), null) {
            public void removedService(ServiceReference<Object>
reference, Object service) {
                logger.info("CseService removed");
            }

            public Object addingService(ServiceReference<Object>
reference) {
                logger.info("CseService discovered");
                CseService cseService = (CseService)
this.context.getService(reference);
                test_ipeController.setCse(cseService);
                // create Resource in mn-cse
                Func.createResources(test_ipeConstants.AE_NAME,
test_ipeConstants.POA);
                GUI.init();
                return cseService;
            }
        };
        cseServiceTracker.open();
```

```
    }

    public void stop(BundleContext bundleContext) throws Exception {
        try {
            logger.info("Stop test_ipe");
            // do something
            GUI.stop();
        } catch (Exception e) {
            logger.error("Stop test_ipe error", e);
        }
    }
}
```

RequestSender.java

```
package org.eclipse.om2m.sample.test_ipe;

import java.math.BigInteger;

import org.eclipse.om2m.commons.constants.Constants;
import org.eclipse.om2m.commons.constants.MimeMediaType;
import org.eclipse.om2m.commons.constants.Operation;
import org.eclipse.om2m.commons.constants.ResourceType;
import org.eclipse.om2m.commons.resource.AE;
import org.eclipse.om2m.commons.resource.Container;
```

```java
import org.eclipse.om2m.commons.resource.ContentInstance;
import org.eclipse.om2m.commons.resource.RequestPrimitive;
import org.eclipse.om2m.commons.resource.Resource;
import org.eclipse.om2m.commons.resource.ResponsePrimitive;
import org.eclipse.om2m.sample.test_ipe.controller.test_ipeController;

public class RequestSender {

    /**
     * Private constructor to avoid creation of this object
     */
    private RequestSender() {
    }

    public static ResponsePrimitive createResource(String targetId,
Resource resource, int resourceType) {
        RequestPrimitive request = new RequestPrimitive();
        request.setFrom(Constants.ADMIN_REQUESTING_ENTITY);
        request.setTo(targetId);
        request.setResourceType(BigInteger.valueOf(resourceType));
        request.setRequestContentType(MimeMediaType.OBJ);
        request.setReturnContentType(MimeMediaType.OBJ);
        request.setContent(resource);
        request.setOperation(Operation.CREATE);
        return test_ipeController.CSE.doRequest(request);
    }

    public static ResponsePrimitive createAE(AE resource) {
        return createResource("/" + Constants.CSE_ID, resource,
ResourceType.AE);
    }

    public static ResponsePrimitive createContainer(String targetId,
Container resource) {
        return createResource(targetId, resource,
ResourceType.CONTAINER);
    }
```

```
    public static ResponsePrimitive createContentInstance(String
targetId, ContentInstance resource) {
        return createResource(targetId, resource,
ResourceType.CONTENT_INSTANCE);
    }

    public static ResponsePrimitive getRequest(String targetId) {
        RequestPrimitive request = new RequestPrimitive();
        request.setFrom(Constants.ADMIN_REQUESTING_ENTITY);
        request.setTo(targetId);
        request.setReturnContentType(MimeMediaType.OBJ);
        request.setOperation(Operation.RETRIEVE);
        request.setRequestContentType(MimeMediaType.OBJ);
        return test_ipeController.CSE.doRequest(request);
    }
}
```

test_ipeRouter.java

```
package org.eclipse.om2m.sample.test_ipe;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.eclipse.om2m.commons.constants.ResponseStatusCode;
import org.eclipse.om2m.commons.exceptions.BadRequestException;
import org.eclipse.om2m.commons.resource.RequestPrimitive;
import org.eclipse.om2m.commons.resource.ResponsePrimitive;
import org.eclipse.om2m.interworking.service.InterworkingService;
import org.eclipse.om2m.sample.test_ipe.constants.test_ipeConstants;
```

```
import org.eclipse.om2m.sample.test_ipe.controller.test_ipeController;

public class test_ipeRouter implements InterworkingService {

    private static Log LOGGER = LogFactory.getLog(test_ipeRouter.class);

    @Override
    public ResponsePrimitive doExecute(RequestPrimitive request) {
        // handle the user command
        ResponsePrimitive response = new ResponsePrimitive(request);
        if (request.getQueryStrings().containsKey("op")) {
            String operation =
request.getQueryStrings().get("op").get(0);
            String value = null;
            if (request.getQueryStrings().containsKey("value")) {
                value = request.getQueryStrings().get("value").get(0);
            }
            LOGGER.info("Received request in test_ipe: op=" + operation
+ " ; value=" + value);
            switch (operation) {
            case "off":
                test_ipeController.setAirConOFF();
                response.setResponseStatusCode(ResponseStatusCode.OK);
                break;
            case "on":
                test_ipeController.setAirConON();
                response.setResponseStatusCode(ResponseStatusCode.OK);
                break;
            case "set_temp":
                if (value.equals("plus") || value.equals("minus")) {
                    if (test_ipeController.setTemp(value))

    response.setResponseStatusCode(ResponseStatusCode.OK);
                    else

    response.setResponseStatusCode(ResponseStatusCode.BAD_REQUEST);
                } else {
```

```
    response.setResponseStatusCode(ResponseStatusCode.BAD_REQUEST);
                }
                break;
            case "set_fan":
                if (value.equals("plus") || value.equals("minus")) {
                    if (test_ipeController.setFan(value))
    response.setResponseStatusCode(ResponseStatusCode.OK);
                    else
    response.setResponseStatusCode(ResponseStatusCode.BAD_REQUEST);
                } else {
    response.setResponseStatusCode(ResponseStatusCode.BAD_REQUEST);
                }
                break;

            case "get_state":
                String content = test_ipeController.getAirConState();
                response.setContent(content);
                response.setResponseStatusCode(ResponseStatusCode.OK);
                break;
            default:
                throw new BadRequestException();
            }
        }
        if (response.getResponseStatusCode() == null) {
    response.setResponseStatusCode(ResponseStatusCode.BAD_REQUEST);
        }
        return response;
    }

    @Override
    public String getAPOCPath() {
        return test_ipeConstants.POA;
    }
```

```
}
```

Operations.java

```
package org.eclipse.om2m.sample.test_ipe.constants;

import org.eclipse.om2m.commons.exceptions.BadRequestException;

/**
 * Represent a operation
 *
 */
public enum Operations {
```

```
    GET_STATE("getState"), GET_STATE_DIRECT("getStateDirect"),
SET_ON("setOn"), SET_OFF("setOff"), TOGGLE("toggle"),
    ALL_ON("allOn"), ALL_OFF("allOff"), ALL_TOGGLE("allToggle");

    private final String value;

    private Operations(final String value) {
        this.value = value;
    }

    public String toString() {
        return value;
    }

    public String getValue() {
        return value;
    }

    /**
     * Return the operation from the string
     *
     * @param operation
     * @return
     */
    public static Operations getOperationFromString(String operation) {
        for (Operations op : values()) {
            if (op.getValue().equals(operation)) {
                return op;
            }
        }
        throw new BadRequestException("Unknow Operation");
    }
}
```

Test_ipeConstants.java

```
package org.eclipse.om2m.sample.test_ipe.constants;

import org.eclipse.om2m.commons.constants.Constants;

public class test_ipeConstants {

    private test_ipeConstants() {
    }
```

```
    public static final String POA = "test_ipe";
    public static final String DATA = "DATA";
    public static final String AE_NAME = "Air_Conditioner";
    public static final String QUERY_STRING_OP = "op";

    public static String CSE_ID = "/" + Constants.CSE_ID;
    public static String CSE_PREFIX = CSE_ID + "/" + Constants.CSE_NAME;
}
```

Func.java

```
package org.eclipse.om2m.sample.test_ipe.controller;

import java.math.BigInteger;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.eclipse.om2m.commons.constants.ResponseStatusCode;
import org.eclipse.om2m.commons.resource.AE;
import org.eclipse.om2m.commons.resource.Container;
import org.eclipse.om2m.commons.resource.ResponsePrimitive;
```

```
import org.eclipse.om2m.sample.test_ipe.RequestSender;
import org.eclipse.om2m.sample.test_ipe.constants.test_ipeConstants;

public class Func {

    private static Log LOGGER = LogFactory.getLog(Func.class);

    public static void createResources(String appId, String poa) {
        // Create the Application resource
        Container container = new Container();
        container.getLabels().add("air_conditioner");
        container.setMaxNrOfInstances(BigInteger.valueOf(0));

        AE ae = new AE();
        ae.setRequestReachability(true);
        ae.getPointOfAccess().add(poa);
        ae.setAppID(appId);
        ae.setName(appId);

        ResponsePrimitive response = RequestSender.createAE(ae);
        // Create Application sub-resources only if application not yet
created

    if(response.getResponseStatusCode().equals(ResponseStatusCode.CREATE
D)) {
            container = new Container();
            container.setMaxNrOfInstances(BigInteger.valueOf(10));
            // Create STATE container sub-resource
            container.setName(test_ipeConstants.DATA);

    LOGGER.info(RequestSender.createContainer(response.getLocation(),
container));
        }
    }
}
```

Test_ipeController.java

```
package org.eclipse.om2m.sample.test_ipe.controller;

import java.util.ArrayList;
import java.util.List;

import org.eclipse.om2m.commons.constants.MimeMediaType;
import org.eclipse.om2m.commons.obix.Bool;
import org.eclipse.om2m.commons.obix.Obj;
import org.eclipse.om2m.commons.obix.Str;
```

```
import org.eclipse.om2m.commons.obix.io.ObixEncoder;
import org.eclipse.om2m.commons.resource.ContentInstance;
import org.eclipse.om2m.core.service.CseService;
import org.eclipse.om2m.sample.test_ipe.RequestSender;
import org.eclipse.om2m.sample.test_ipe.constants.test_ipeConstants;
import org.eclipse.om2m.sample.test_ipe.model.test_ipeModel;

public class test_ipeController {

    public static CseService CSE;
    protected static String AE_ID;
    private static Observer GUIOBSERVER = null;

    public static void setGUIOBSERVER(Observer obs) {
        GUIOBSERVER = obs;
    }

    public static interface Observer {
        void StateChange(String msg);
    }

    private static void createDATA() {
        // notify GUI
        notifyObserver(test_ipeController.getAirConState());
        // Send the information to the CSE
        String targetID = test_ipeConstants.CSE_PREFIX + "/" +
test_ipeConstants.AE_NAME + "/" + test_ipeConstants.DATA;
        ContentInstance cin = new ContentInstance();
        Obj obj = new Obj();
        obj.add(new Bool("on/off", test_ipeModel.getAirConValue()));
        obj.add(new Str("Temperature",
String.valueOf(test_ipeModel.getAirConTemp())));
        obj.add(new Str("Fan",
String.valueOf(test_ipeModel.getAirConFan())));
        cin.setContent(ObixEncoder.toString(obj));
        cin.setContentInfo(MimeMediaType.OBIX + ":" +
MimeMediaType.ENCOD_PLAIN);
        RequestSender.createContentInstance(targetID, cin);
```

```
    }

    private static void notifyObserver(final String msg) {
        new Thread() {
            @Override
            public void run() {
                GUIOBSERVER.StateChange(msg);
            }
        }.start();
    }

    public static void setAirConON() {
        // Set the value in the "real world" model
        test_ipeModel.setAirConON();
        // Send the information to the CSE
        createDATA();
    }

    public static void setAirConOFF() {
        // Set the value in the "real world" model
        test_ipeModel.setAirConOFF();
        // Send the information to the CSE
        createDATA();
    }

    public static void switchAirCon() {
        // Set the value in the "real world" model
        if (test_ipeModel.getAirConValue() == false) {
            test_ipeController.setAirConON();
        } else {
            test_ipeController.setAirConOFF();
        }
    }

    public static String getAirConState() {
        String state = String.valueOf(test_ipeModel.getAirConValue());
        state += ',' + String.valueOf(test_ipeModel.getAirConTemp());
        state += ',' + String.valueOf(test_ipeModel.getAirConFan());
```

```
        return state;
    }

    public static boolean setTemp(String PM) {
        boolean res = false;
        if (PM.equals("plus")) {
            // Set the value in the "real world" model
            res = test_ipeModel.setTempPlus();
        } else if (PM.equals("minus")) {
            // Set the value in the "real world" model
            res = test_ipeModel.setTempMinus();
        }
        createDATA();
        return res;
    }

    public static boolean setFan(String PM) {
        boolean res = false;
        if (PM.equals("plus")) {
            // Set the value in the "real world" model
            res = test_ipeModel.setFanPlus();
        } else if (PM.equals("minus")) {
            // Set the value in the "real world" model
            res = test_ipeModel.setFanMinus();
        }
        createDATA();
        return res;
    }

    public static void setCse(CseService cse) {
        CSE = cse;
    }

}
```

Test_ipeModel.java

```
package org.eclipse.om2m.sample.test_ipe.model;

public class test_ipeModel {

    private static boolean AIRCON = false;
    private static int AIRCON_temp = 25;
    private static int AIRCON_fan = 1;

    private test_ipeModel() {
```

```
    }

    // Sets the direct current Air Conditioner state

    public static void setAirConON() {
        AIRCON = true;
    }

    public static void setAirConOFF() {
        AIRCON = false;
    }

    public static boolean setTempPlus() {
        if (AIRCON == false)
            return false;
        AIRCON_temp++;
        return true;
    }

    public static boolean setTempMinus() {
        if (AIRCON == false)
            return false;
        AIRCON_temp--;
        return true;
    }

    public static boolean setFanPlus() {
        if (AIRCON_fan == 3 || AIRCON == false)
            return false;
        else {
            AIRCON_fan++;
            return true;
        }
    }

    public static boolean setFanMinus() {
        if (AIRCON_fan == 1 || AIRCON == false)
            return false;
```

```
        else {
            AIRCON_fan--;
            return true;
        }
    }

    // Gets the direct current Air Conditioner state

    public static boolean getAirConValue() {
        return AIRCON;
    }

    public static int getAirConTemp() {
        return AIRCON_temp;
    }

    public static int getAirConFan() {
        return AIRCON_fan;
    }
}
```

GUI.java

```
package org.eclipse.om2m.sample.test_ipe.gui;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.EventQueue;
import java.awt.Font;

import javax.swing.ImageIcon;
import javax.swing.JButton;
```

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JSplitPane;
import javax.swing.border.EmptyBorder;
import javax.swing.border.LineBorder;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.eclipse.om2m.sample.test_ipe.controller.test_ipeController;
import org.osgi.framework.FrameworkUtil;

public class GUI extends JFrame {
    /**
     * automatically generated
     */
    private static final long serialVersionUID = 53230770754627227l8L;
    /** Logger */
    static Log LOGGER = LogFactory.getLog(GUI.class);
    /** GUI Frame */
    static GUI frame;
    /** GUI Content Panel */
    private JPanel contentPanel;
    /** AIR_ON Icon */
    static ImageIcon iconAirON = new
ImageIcon(FrameworkUtil.getBundle(GUI.class).getResource("images/Btn_ON.
png"));
    /** AIR_OFF Icon */
    static ImageIcon iconAirOFF = new
ImageIcon(FrameworkUtil.getBundle(GUI.class).getResource("images/Btn_OFF
.png"));
    /** GUI observer */
    static test_ipeController.Observer obs;
    /** switch button */
    static JButton AirButton = new JButton();
    /** LAMP_1 LABEL */
    static JLabel lbltempGraph = new JLabel("tempGraph");
    static JLabel lblfanGraph = new JLabel("fanGraph");
```

```
    /**
     * Initiate The GUI.
     */
    public static void init() {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    frame = new GUI();
                    frame.setVisible(true);
                } catch (Exception e) {
                    LOGGER.error("GUI init Error", e);
                }
            }
        });
    }

    /**
     * Stop the GUI.
     */
    public static void stop() {
        // SampleModel.deleteObserver(lampObserver);
        frame.setVisible(false);
        frame.dispose();
    }

    /**
     * Creates the frame.
     */
    public GUI() {
        setLocationByPlatform(true);
        setVisible(false);
        setResizable(false);
        setTitle("DCNLab test IPE");
        setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        java.awt.Dimension screenSize =
java.awt.Toolkit.getDefaultToolkit().getScreenSize();
```

```
        setBounds((screenSize.width - 500) / 2, (screenSize.height -
570) / 2, 497, 570);

        lbltempGraph.setFont(new Font("Vani", Font.BOLD | Font.ITALIC,
30));
        lblfanGraph.setFont(new Font("Vani", Font.BOLD | Font.ITALIC,
30));

        contentPanel = new JPanel();
        contentPanel.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPanel);
        contentPanel.setLayout(null);

        // temperature control
        JPanel panel_temp = new JPanel();
        panel_temp.setBounds(10, 5, 319, 260);
        panel_temp.setBorder(new LineBorder(new Color(0, 0, 0), 1,
true));
        contentPanel.add(panel_temp);

        JSplitPane splitPane_graph_temp = new JSplitPane();
        panel_temp.add(splitPane_graph_temp);

        JSplitPane splitPane_temp = new JSplitPane();
        splitPane_temp.setOrientation(JSplitPane.VERTICAL_SPLIT);
        splitPane_graph_temp.setLeftComponent(lbltempGraph);
        splitPane_graph_temp.setRightComponent(splitPane_temp);
        panel_temp.add(splitPane_temp);

        JButton btntempPLUS = new JButton("tempPLUS");
        splitPane_temp.setLeftComponent(btntempPLUS);
        // Listener of btntempPLUS
        btntempPLUS.addActionListener(new
java.awt.event.ActionListener() {
            // btntempPLUS Button Clicked
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                new Thread() {
```

```
                    public void run() {
                        test_ipeController.setTemp("plus");
                    }
                }.start();
            }
        });

        JButton btntempMINUS = new JButton("tempMINUS");
        splitPane_temp.setRightComponent(btntempMINUS);
        // Listener of btntempMINUS
        btntempMINUS.addActionListener(new
java.awt.event.ActionListener() {
            // btntempMINUS Button Clicked
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                new Thread() {
                    public void run() {
                        test_ipeController.setTemp("minus");
                    }
                }.start();
            }
        });

        // fan control
        JPanel panel_fan = new JPanel();
        panel_fan.setBounds(10, 271, 319, 260);
        panel_fan.setBorder(new LineBorder(new Color(0, 0, 0), 1,
true));
        contentPanel.add(panel_fan);

        JSplitPane splitPane_graph_fan = new JSplitPane();
        panel_fan.add(splitPane_graph_fan);

        JSplitPane splitPane_fan = new JSplitPane();
        splitPane_fan.setOrientation(JSplitPane.VERTICAL_SPLIT);
        splitPane_graph_fan.setLeftComponent(lblfanGraph);
        splitPane_graph_fan.setRightComponent(splitPane_fan);
        panel_fan.add(splitPane_fan);
```

```
        JButton btnfanPLUS = new JButton("fanPLUS");
        splitPane_fan.setLeftComponent(btnfanPLUS);
        // Listener of btnfanPLUS
        btnfanPLUS.addActionListener(new
java.awt.event.ActionListener() {
            // btnfanPLUS Button Clicked
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                new Thread() {
                    public void run() {
                        test_ipeController.setFan("plus");
                    }
                }.start();
            }
        });

        JButton btnfanMINUS = new JButton("fanMINUS");
        splitPane_fan.setRightComponent(btnfanMINUS);
        // Listener of btnfanMINUS
        btnfanMINUS.addActionListener(new
java.awt.event.ActionListener() {
            // btnfanMINUS Button Clicked
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                new Thread() {
                    public void run() {
                        test_ipeController.setFan("minus");
                    }
                }.start();
            }
        });

        // Air switch
        AirButton.setOpaque(false);
        AirButton.setIcon(iconAirOFF);
        AirButton.setBounds(339, 190, 145, 168);
        contentPanel.add(AirButton);
```

```
        AirButton.setMinimumSize(new Dimension(30, 23));
        AirButton.setMaximumSize(new Dimension(30, 23));
        AirButton.setPreferredSize(new Dimension(30, 23));

        JLabel labelSwitchAll = new JLabel("ON/OFF");
        labelSwitchAll.setAutoscrolls(true);
        labelSwitchAll.setFont(new Font("Vani", Font.BOLD |
Font.ITALIC, 14));
        labelSwitchAll.setFocusCycleRoot(true);
        labelSwitchAll.setBorder(null);
        labelSwitchAll.setBounds(371, 369, 85, 29);
        contentPanel.add(labelSwitchAll);
        // Listener of AirButton
        AirButton.addActionListener(new java.awt.event.ActionListener()
{
            // Switch Button Clicked
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                // Change all lamps states
                new Thread() {
                    public void run() {
                        // Send switch all request to create a content
with the current State
                        test_ipeController.switchAirCon();
                    }
                }.start();
            }
        });

        obs = new test_ipeController.Observer() {
            @Override
            public void StateChange(String msg) {
                // TODO Auto-generated method stub
                String sp[] = msg.split(",");
                setLabel(Boolean.parseBoolean(sp[0]), sp[1], sp[2]);
            }
        };
        test_ipeController.setGUIOBSERVER(obs);
```

```
    }

    public static void setLabel(boolean newState, String temp, String
speed) {
        if (newState == true) {
            AirButton.setIcon(iconAirON);
        } else {
            AirButton.setIcon(iconAirOFF);
        }
        lbltempGraph.setText(temp);
        lblfanGraph.setText(speed);
    }

    /*
     * main function for test
     */
    public static void main(String[] args) {
        System.out.println("test");
        GUI.init();
    }
}
```

建立一個 MN-AE(名稱: **org.eclipse.om2m.sample.test_ipe**)在 MN-CSE 底下，其中 ipeRouter.java 及 ipeController.java 各別將 GUI.java 介面變動的內容處理後儲存到 MN-CSE/mn-name/Air_Conditioner/DATA 中且建立新的 Content Instance，透過 NODE-RED(相當於一個 AE)訂閱此 Container(Air_Conditioner)，將底下新增的內容 notify 到 POA 的路徑，POA 的路徑為(http://127.0.0.1:1880/AirCon) ，送到 AE 後在 AE 內進行處理後顯示在 Node-Red Debug 中。