Assignment I – DPWS and REST deployment

Dharmendra Sharma – 272534
Junaid Iqbal - 272536
Ronal Bejarano – 272544

Objective

Implement a web-based application by using DPWS, SOAP over a Node.JS based application able to subscribe, listen and execute HTTP events in a basic network established with a INCO-1000 PLC.

Code description

The code is developed in two Javascript files. REST message implementation is available on mainREST.js and SOAP implementation on main SOAP.js. The difference between both approaches is based only in the payload of the POST HTTP request from the server to the PLC, aiming to change the values of its digital outputs. All other code lines are equal. There is no specific reason for this approach, only than initially the development of the script started to be coded by separate.

REST: (JSON payload)

```
/******************************************************/
/*Attaching the required NPM PACKAGES                 */
/******************************************************/

var request = require('request');           // HTTP
var express = require('express');           // Web framework
var app = express();
var bodyParser = require('body-parser');    //Allows parse
var path = require('path');                 //Discover path for file references

/******************************************************/
/*Desired server host IP and port                     */
/******************************************************/

host = '192.168.100.107';
port = 4444;

/******************************************************/
/*body parser dependencies and serving static files   */
/******************************************************/

app.use(bodyParser.urlencoded({
    extended: true
}));
app.use(bodyParser.json());
app.use(express.static(path.join(__dirname, 'files')));

/******************************************************/
/*JSON variable for frontend                          */
/******************************************************/

var plcData={};
plcData.time="00000";
plcData.activityStatus="None";
plcData.eventCounts=0;
plcData.outputs={"op0":"Ivory","op1":"Ivory","op2":"Ivory","op3":"Ivory","op4":"Ivory","op5":"Ivory","op6":"Ivory","op7":"Ivory"};

/******************************************************/
/*HTTP request options                                */
/******************************************************/

var options_req = {
    body: {"destUrl":"http://192.168.100.107:4444"}, // Javascript object payload
```

```
to tell PLC about the source of incoming request
    json: true,
    url: "",
    headers: {
        'Content-Type': 'application/json'
    }
};

/*****************************************************/
/*Generic function for HTTP post                     */
/*****************************************************/

function post_request() {
    request.post(options_req, function (err, res, body) {
        if (err) {
            console.log('Error :', err);
        }
        //console.log(' RESPONSE OF POST REQUEST :', JSON.stringify(body)); //
Console log of the response - For diagnostics
    });
};

/*****************************************************/
/*function to call delete request - Explored but not used
/*****************************************************/
/*function delete_request() {
    request.delete(options_req, function (err, res, body) {
        if (err) {
            console.log('Error :', err);
            return;
        }
        console.log(' RESPONSE OF DELETE REQUEST :', JSON.stringify(body)); // Just
printing the return message as we post req.
    });
};
options_req.url="http://192.168.100.106/rest/events/time/notifs";
delete_request();                                                    // delete
request if subscriptions already exists
*/

options_req.url="http://192.168.100.106/rest/events/time/notifs";   // Initial
request to subscribe for iterative notifications from PLC
post_request();

options_req.url="http://192.168.100.106/rest/services/startEvents";
post_request();                                                      //  Initial
trigger event to start the notifications

/***********************************************************************************
************************************************/
/*GET request handler from server's root address - Enables the Home site to be
accessed through a web browser by server ip:port address   */
/***********************************************************************************
************************************************/

app.get('/', function (req, res) {
    res.sendFile(__dirname + "/" + "files/Home.html");
    //console.log("METHOD: GET");                                    // For
diagnostics
});

/*****************************************************/
/*GET request handler (from ajax, front end)         */
/*****************************************************/

app.get('/frontend', function(req, res){
    var obj = {};
    res.send(plcData);                                           // Sends a JSON
object with data to front end for visualization
    /*
    console.log('GET REQ. FROM AJAX ');                         // For diagnostics
    console.log('body: ' + JSON.stringify(req.body))            // For diagnostics
    console.log('RESPONSE PAYLOAD TO THE AJAX REQ.\N',plcData); // For diagnostics
    */
});

/*****************************************************/
/*POST request (from PLC) handler                    */
/*****************************************************/

app.post('/', function (req, res) {
        plcData.eventCounts+=1;                             // received POST
request event counter
        var body = req.body;
        var receivedTime=body.payload.timeStamp;            // Extracting timeStamp
        plcData.time=receivedTime;
        var timeSec=receivedTime[17]+receivedTime[18];      // extracting the
'seconds' of the timeStamp
        var out=d2b_array(timeSec);                         // binary outputs
framing from time 'seconds'
        res.writeHead(200, {'Content-Type': 'text/html'});  // response back to the
'POST' request from the PLC
        res.end();
        //console.log("METHOD: POST REQ. FROM PLC\n");     // For diagnostics
    /*****************************************************/
    /*REST POST REQUEST TO CHANGE OUTPUT OF PLC          */
    /*****************************************************/
        request.post({                                      // REST post request to
change the outputs of the PLC
            body:
{"state0":out[0],"state1":out[1],"state2":out[2],"state3":out[3],"state4":out[4],"s
tate5":out[5],"state6":out[6],"state7":out[7]}, // Javascript object payload
            json: true,
            url: "http://192.168.100.106/rest/services/changeOutput", // URI for
request
            headers: {
                'Content-Type': 'application/json'
            }
        }, function (err, res, body) {
            if (err) {
                console.log('Error :', err);
                return;
            }
            //console.log(' REST REQUEST`S RESPONSE :', JSON.stringify(body)); //
Just printing the return message as we post req. - For diagnostics
        });
});
/*****************************************************/
/*Server listener                                    */
/*****************************************************/

app.listen(port, host, function () {
console.log('Server is listening on http://192.168.100.107:4444\n')});

/*********************************************************************************
*************************/
/*Function to convert 'seconds' on integer to binary vector and re format from
'1/0' to 'true/false' and
'GreenYellow/Ivory' for the PLC and HTML style interface respectively
/*********************************************************************************
************************/

function d2b_array(dec)
{
    var bin=[];
    var binHTML=[];
    for(i=0;i<8;i++)
    {
        var a=dec&1;    // '&' (logical and) is better to use then '&' for modulus
of positive binary numbers.
        if(a==1)
        {
            bin[7-i]=true;
            binHTML[7-i]="GreenYellow ";
        }
        else
        {
            bin[7-i]=false;
            binHTML[7-i]="Ivory";
        }
        dec=dec/2;
    }
    i=0;                //Javascript object for LED 'color' HTML style for front
end
    for (var key in plcData.outputs) {
        if (plcData.outputs.hasOwnProperty(key)) {
            plcData.outputs[key]=binHTML[7-i];
            i++;
        }
    }
    return bin;
}

/*********************************************************************************
**/
/*HTTP Get request to PLC at regular interval to monitor the PLC status*/
/*********************************************************************************
**/

setInterval(function () {
    options_req.url="http://192.168.100.106/rest/events/time/notifs";   // get
request to subscribe to notifications
    options_req.timeout=900;                                            // small
post request timeout for quick disconnection detection
    request.get(options_req, function (error, response, Body) {
        if (error) {                                                    // no
response
            plcData.activityStatus="PLC not available";
            //console.log('Error :', error); // For diagnostics
            return;
        }
        if(JSON.stringify(Body.children)==='{}') {                      // Empty -
No response
            plcData.activityStatus="Subscribing...";
            options_req.url="http://192.168.100.106/rest/events/time/notifs";   //
Attempt to re subscribe again
            post_request();
            options_req.url="http://192.168.100.106/rest/services/startEvents"; //
Triggers events
            post_request();
        }
        if(JSON.stringify(Body.children)!=='{}') {
            plcData.activityStatus="PLC active";
        }
        //console.log('DEVICE STATUS:', JSON.stringify(Body)); // Just printing the
return message as we post req. - For diagnostics
    });
},1000); // Update rate of HTML front end: 1 second
```

SOAP: (XML payload)

```
    /*****************************************************/
    /*SOAP POST REQUEST                                  */
    /*****************************************************/
    request.post({                                            // REST post
request to change the outputs of the PLC
        headers: {                                            // header as
per instructions
            'accept':
'text/html,application/xhtml+xml,application/xml,text/xml;q=0.9,*/*;q=0.8',
            'accept-encoding': 'none',
            'accept-charset': 'utf-8',
            'connection': 'close',
            'host': '192.168.100.107:80',
            'content-type': 'application/xml'
        },
        url: "http://192.168.100.106:80/dpws/WS01",           // uri of the
soap destination
        body: "<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>\n" + // soap xml
request body
        "<s12:Envelope\n" +
        "\txmlns:s12=\"http://www.w3.org/2003/05/soap-envelope\"\n" +
        "\txmlns:wsa=\"http://schemas.xmlsoap.org/ws/2004/08/addressing\">\n" +
        "<s12:Header>\n" +

"<wsa:Action>http://www.tut.fi/fast/Assignment/UpdateOutputs_Request</wsa:Action>\n
" +
        "</s12:Header>\n" +
        "<s12:Body xmlns:tns=\"http://www.tut.fi/fast/Assignment\">\n" +    //
targetNameSpace
        "\t\t<tns:Outputs>\n" +
        "<tns:output0>"+out[0]+"</tns:output0>\n" +
        "<tns:output1>"+out[1]+"</tns:output1>\n" +
        "<tns:output2>"+out[2]+"</tns:output2>\n" +
        "<tns:output3>"+out[3]+"</tns:output3>\n" +
        "<tns:output4>"+out[4]+"</tns:output4>\n" +
        "<tns:output5>"+out[5]+"</tns:output5>\n" +
        "<tns:output6>"+out[6]+"</tns:output6>\n" +
        "<tns:output7>"+out[7]+"</tns:output7>\n" +
        "</tns:Outputs>\n" +
        "</s12:Body>\n" +
        "</s12:Envelope>"                   // soap message payload end
    }, function (err) {
        if (err) {
            console.log('Error :', err);
            return;
        }
    });
});
```

Challenges and limitations

Understanding properly the format contained on the WSDL was the first challenge. It was an obstacle to obtain the right XML syntax for the SOAP request. To overcome this, a XML sample message was generated by Boomerang SOAP client by reading the WSDL file, but the result was not successful since the message required 3 tags and one of them was part of the body (tns). Merging the results of the WSDL interpretation from boomerang and the recommendations from the assignment description was possible to find out the proper message syntax.

Initially the code was developed by using the NPM http module, but later it was changed by express.

The use of socket io was not achieved before the implementation of iterative ajax request to connect frontend and backend.

It is still unclear why the PLC loses the subscription when ethernet cable is disconnected and reconnected.

Questions

1. **Describe the advantages and disadvantages of using HTTP-based protocols in industrial information systems?**

   HTTP Protocols are quite popular in industry environment and reasons behind its popularity are: IP addressing is quite viable. It is quite flexible for all kind of applications. In automation pyramid, the horizontal communication contains data for logical and control instructions. While vertical communication is for different managerial levels. On vertical level, HTTP is used for flow of data from control systems to level 3 information systems and on horizontal layer, HTTP helps in plc to plc communication etc. By using firewalls, it is quite easy between layers to divide the layer in segments. HTTP protocols allow communication systems to transfer data at request, and the communication is real time. Multiple clients can be connected to one server at a time and they can run in parallel. Reusability and interoperate ability of webservices make HTTP quite popular. HTTP address destination or source device based on IP address, and it is quite secure and independent messaging system. UDDI allows cross platform programming. REST webservices are light, generic and are open to content type. While SOAP webservices are quite secure. HTTP is quite reliable and allows the exchange of metadata. Since, HTTP communication takes place over TCP connections, so different ports can be used for communication. Some security issue may arise with *http* but *https* may be used as its replacement for secured services.

2. **Mention the importance of XSD when integrating systems**

   XML Schema Definition, specifies the formal definition of an element in XML. It can be used to describe objects characteristics and attributes. When integrating systems, XSD can be used to write human readable documentation of XML. XSD tools can be used to produce readable HTML. XSD based XML documents can be treated as programming objects. XSD is based on XML, So, it does not require parsers.

3. Explain what is parsing. In which part of the assignment do you use it?

Parsing is the conversion of a text into more useful format. Like in XML, a parser can be used to convert characters into attributes. The actual definition of parsing is: to break data into small chunks so, they can be stored or manipulated, [1].

In the assignment we parse incoming request bodies in a middleware before using handlers, available under the '*req.body*' property. For '*express*' it is used in the following way-

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({
  extended: true
}));
app.use(bodyParser.json());
```

This parser accepts any Unicode ('ISO-_' in this case) encoding of the '*body*'. A new '*body*' object containing the parsed data is populated on the '*request*' object after the middleware (i.e. *req.body*).

4. In this assignment, are you using an Event Driven Architecture? In which part? Justify

In the assignment, the server (using express) is developed to handle the incoming events from PLC and front end. Whenever the PLC's status POST request arrives at the server, it receives and process it (based on event, no need of continuous polling/continuous monitoring).

Same for GET request handler from '/frontend' the server handles the request and sends the JSON visualization payload data.

The above events are highlighted by blue in the descriptive diagram below.

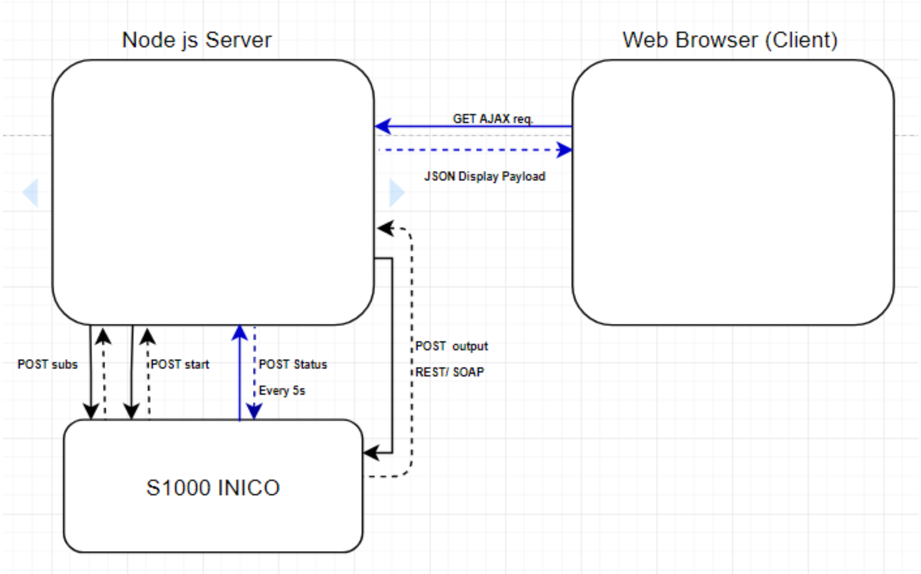

Figure 1: Events flow of the system

5. **Make a comparison between REST and SOAP (used in DPWS) Web Services. Generate a table with packets sent and received.**

REST	**SOAP**
Is an architectural style.	Is an XML based method.
Calls services by URL.	Calls services by RPC method.
Used XML or JSON for communication.	Uses WSDL for communication
Uses only HTTP protocol.	Can use other protocol.
Performance is great compared to SOAP.	Not that great performance.
Human Readable	Difficult to read for humans.

This is the data till 5 statuses received. It's not clear how to monitor.
REST:

NETWORK INTERFACE STATISTICS

Packets sent:	351
Packets received:	436

SOAP:

Packets sent:	321
Packets received:	438

Conclusions

- Web services implemented on industrial applications represent important advantages on flexibility and ease of access for client devices. Information flows can be optimized by event driven data exchange.
- Network management and libraries on Node.JS represented a technical challenge for implementation. Dependence on external libraries make this kind of implementations prone to compatibility issues.
- Deployment of web based automation and control applications can enhance the interoperability of multivendor applications in the near future.