Assignment 1

ASE-9476 Factory Information Systems

Juha Suntila, 256667
Aman Khan, 272541
Jarno Nieminen, 222712

Assignment objectives

The assignment objectives were to practice working with INICO S1000 industrial controller and learn how to operate it by using REST and SOAP protocols as platform NodeJS was used.

The purpose was to send the INICO S1000 to send events to the server created with NodeJS. Then the device sends time stamp, where the seconds part of the timestamp is picked up. Then the seconds are changed to 8-bit binary value. The eight outputs are supposed to be set on or off depending on the binary value zeros and ones. The INICO device sends the stamp in every 5 seconds and in every 5 seconds the outputs are also supposed to change.

Code description

- In the code, the first things were to define the modules to be used. Then the global variables were introduced.
- The next stage was to create the server to localhost. After that the subscribe events part of the code was created.
- After the subscribe event part worked, the focus was moved to get the timestamps.
- After the time stamps were successfully received, the body was parsed in a way that the seconds part of the body was separated.

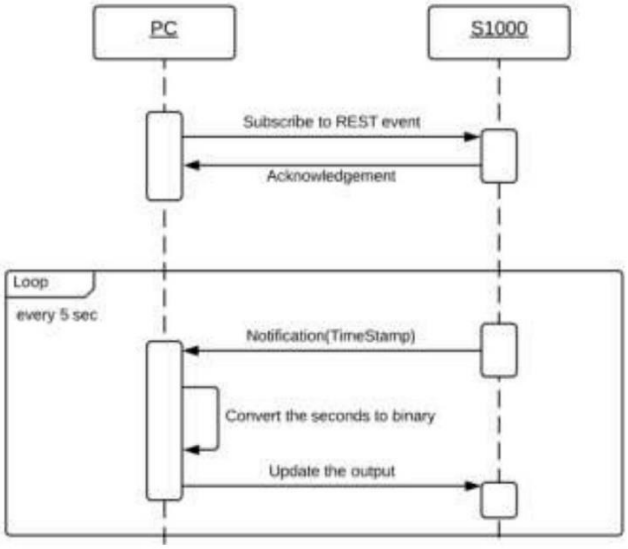

Figure 1: Sequence Diagram

- **Parsing the body**

  For parsing the message, the bodyParser module was used. By means of bodyParser the timestamp was picked from inside the payload, which was inside the body and inside the response message.The timeStamp was picked from the body. The time stamp was received in ISO format. The timestamp format looked like as follows when received properly: ” 2018-02-12T10:54:08.350 ”.At this point, the body parser part was done.

- The timeStamp was then fed to built-in javascript function called ”Date(),” which could convert the timestamp string into standard date format. From this format, it was then easy to pick up the seconds part by using ”getSeconds()” function. It returned the seconds reading in string format.Then the seconds string was converted to binary array and after that the binary array was converted to boolean array with size of eight bits.
- Then the boolean array was fed inside the body to be sent to change the output of the INICO controller.
- The server changes the outputs according to the time stamp every time the INICO controller sends the time stamp.

The code is added as attachment of this file with comments included. Also the console log is bellowed :

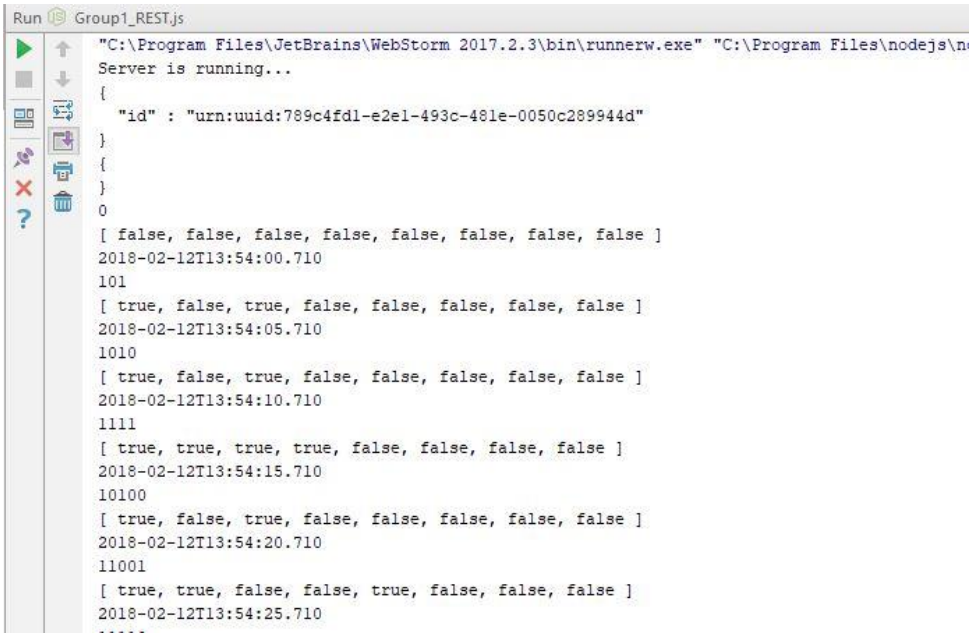

Figure 2: Console Log

An HTML interface is built for the application showing the number of received events, the status of the outputs and the current time in the S1000 RTU.

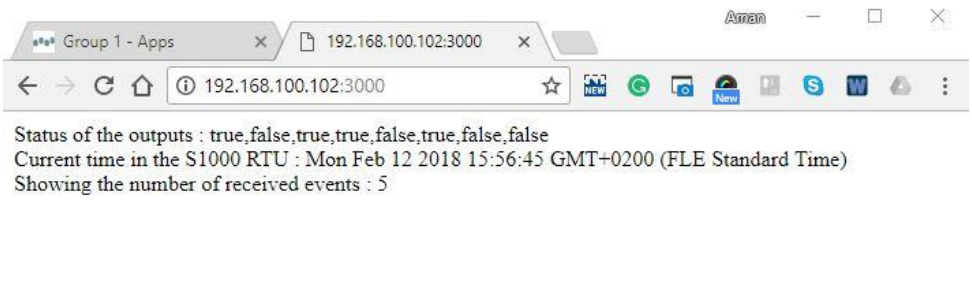

Figure 3: HTML interface

Challenges and limitations

Challenges were mostly related to nodeJS syntax and programming language.

- Body parser using gave a surprise that the seconds part really could not be picked up from the body by using it. Body parser just gave the timestamp entirely. So then other built-in function was needed to be used to get the actual seconds out of it.
- The body format to send the outputs with REST took some time to be realized. After it was realized that the body actually had to be built up by adding strings together with boolean values between, the output writing was successful.
- Then next challenge was to find out the correct syntax of the SOAP XML body. This required searching the INICO controller's inner variables and the WSDL file to show the proper syntax. After that the output changing went quite similar way as it went with REST.

Limitations:

- The INICO controller went to server overload mode several times (Figure4).

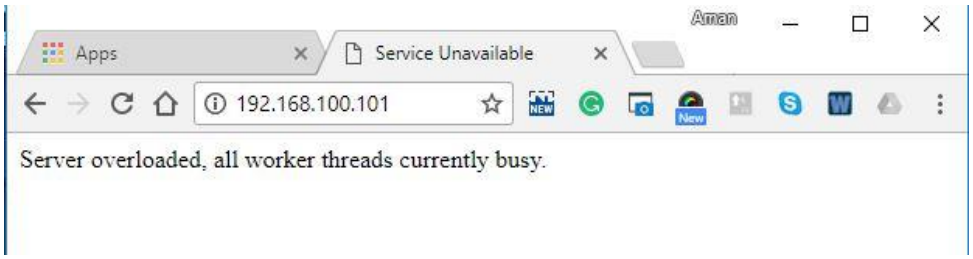

Figure 4: Server Overload

- Sometimes if the program was restarted for several times, it appeared that the controller gave as many timestamps as the restarts were done.
- After consecutive switching between Configuration mood and Run mood, some it gives the subscription id, but it did not give the timestamp or change the output.

Questions

Question 1: Describe the advantages and disadvantages of using HTTP-based protocols in industrial information systems? (Include in your response a reference to vertical and horizontal communications in the pyramid of automation)

- -HTTP based protocols make possible communication and control between many different devices in different OSI layers. So it is a very convenient way to implement different layers that don't need very quick reactions to data received.
- -The disadvantages are that the HTTP based protocol is not reliable enough to take full control of the process in industry, so the actual control needs to be done still in separate controller. In industry the fault in messages can cause severe failures in process and most often these failures may be extremely costly. Another thing in industry is that control should be quickly responding. The process state changes quickly and the controller must react in time of milliseconds. _This is not happening with HTTP.

Question2: Mention the importance of XSD when integrating systems

XSD defines the format the XML file should be when communicating. Program developer can use XSD file to check the application properties and by that means to create the program that can communicate successfully with the application. In XSD the limitations and freedoms of the input parameters are also defined. For example in XSD it is defined, whether you need to input strings or integers, in range of 0…100 or string of a maximum length of 100 characters etcetera.

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

Parsing is the process to extract exact information from massage. In node.js, using Express module, post request can be done and as a response. A enormous massage arrived including header, body, etc. From this kind of massive data, it is tough to find out and exactly retrieve the data we are interested. To easily retrieve the exact data, a npm module call ”Body-parser” can be used. It is an Express module middleware and every request first handled by this module.

In this assignment, INICO S1000 RTU sends a long response message after the subscription which contains the timestamp under payload in the body part of the message.

```
    _onPendingData: [Function: updateOutgoingData],
    req: [Circular],
    locals: {} },
  body:
   { id: 'RandomNumber',
     type: 'event',
     payload: { timeStamp: '2018-02-12T11:05:57.310' } },
  _body: true,
  length: undefined,
  read: [Function],
  route: Route { path: '/', stack: [ [Object] ], methods: { post: true } } }
{
}
```

Figure 5: Response message

To select the precisely the timestamp, we use the Body-parser module. Mainly the message body is in Json key/value pairs. So we use JSON body-parser and implementing the code and get the result.

```
app.post('/', function (req, res) {
    countOut=timeSta(req.body.payload.timeStamp) /*parsing timestamp from the body and calling timeSta function to get the seconds*/
    console.log(req.body.payload.timeStamp)
    res.send('POST request to homepage');
});
```

Figure 6: Implemented code

```
Server is running...
{
  "id" : "urn:uuid:8ed88192-b059-42ab-5dd3-0050c289944d"
}
10110
[ true, false, true, true, false, false, false, false ]
2018-02-12T11:47:22.310
{
}
11011
[ true, true, false, true, true, false, false, false ]
2018-02-12T11:47:27.310
```

Figure 7: Time Stamp

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

We are indeed using the event-driven architecture. The nodeJS server works asynchronously, and the events actually trigger the functions required. Otherwise, the server stays listening for events to come. In the assignment events are used in following spots:

- when the server sends the ”notifs,” the event is sent to the controller (in code ”var timenotifs”), and the controller returns a subscription id.
- when the controller is asked to start sending the time stamps (”var startEvents”)
- when the controller sends the time stamp to the server
- when the server sends the event to controller commanding it to change the state of the outputs (”var changeOutput”)

Question 5: Make a comparison between REST and SOAP (used in DPWS) Web Services.

We took observation of packets from the network monitor of the controller. However, it is hard to make any decision.The controller interface added some data whenever the Network tab pressed.

Web Services	Packets	Observation 1	Observation 2	Observation 3
REST	Sent	226	112	211
SOAP used in DPWS	Sent	173	163	183
REST	Received	338	276	358
SOAP used in DPWS	Received	311	299	300

By using Wireshark, for SOAP Web Services, the length of the send message for changing output is always 1095 bytes. For REST Web Services, the length of the send message for changing output is 332 bytes. So the SOAP web services have large length of data compared to REST Web Services.