## 2.1.3 Access, control and reconfiguration

This chapter focuses on how the user is able to accesses our testbed and what is the technology behind it. Starting from sensor network towards the server side is the sensor network which is based on Atmel ZigBit modules. This is a multihop network so the gateway is able to communicate also with the nodes which are not in its wireless range. For the purpose of communication between sensor network and our server on JSI we developed a new protocol (Figure 1) which was inspired by HTTP protocol and is simple enough for a fast implementation on VESNAs. The protocol defines two requests GET and POST which are understood by every VESNA. The GET is used for “safe” requests which do not change the state of the system and POST for “unsafe” requests which change the state of the system. The response is considered to be in binary format although it is normally in text format expect the spectrum sensing data are binary. Every response ends with OK\r\n and this is how we recognize the end of the response.

GET:  
    GET resource?arg1=val1&arg2=val2&...&argN=varN\r\n  
  
    resource: abstract resource identifier

examples: - firmare/version

- sensors/temperature

arg1: parameter 1 name

val1: value of parameter 1

…

argN: parameter N name

valN: value of parameter N

POST:  
    POST resource?arguments\r\n  
    Length=len\r\n  
    <data, having len bytes length>\r\n  
    crc=crc\_value\r\n  
  
    resource: abstract resource, for example: firmware  
    arguments: arguments given to the handler of POST

example: 2.34/firmware

len: length of the data that will be written to the specific

resource

data: possibly binary data, to be transmitted

crc\_value: value of CRC calculated on all the previous content

    except the line starting with crc=;

value represented as an unsigned decimal number

Responses from the coordinator have the general form:  
  
    <response to a specific request>\r\n  
    OK\r\n

Figure 1: Resource access protocol

The protocol includes simple and efficient error handling mechanism (Figure 2). There are two types of errors the first is JUNK-INPUT this is the more common situation when someone mistypes the resource name and the parser on the node do not recognize it. After this response the parser on the node expects 5 new lines which resets the parser after that someone could try to access the resource again. The second type of error is CORRUPTED-DATA this means that CRC check did not succeed thus we can conclude that the error happened some were on the line between the infrastructure and gateway. The last situation will occur with very low probability.

<output from node, description of error>\r\n

<JUNK-INPUT or CORRUPTED-DATA>\r\n

\r\n

OK\r\n

Figure 2: Error handling

The protocol is designed as client-server. In our case the servers are sensor nodes and the client is the infrastructure. Before we can access the resources the gateway has to establish connection with the infrastructure. This is done by establishing a secure SSL encrypted socket with the infrastructure. The gateway has an Ethernet module embedded on the expansion board which is used to connect the gateway to the internet. The Ethernet module is configured to get the IP address from DHCP server and then automatically trays to setup an encrypted SSL socket with one of the SSL servers listening on specific port located on the infrastructure. Once the connection has been established one could access any resource on any of the nodes. Resources pre-prepared on the nodes includes features like: Remote reprogramming, start spectrum sensing, collecting spectrum data, configure nodes as transmitters, configure frequency band, … Figure 3 depicts a complete schematic of our testbad. We already described the establishment of the connection between the nodes to gateway and gateway to infrastructure. We also mentioned that the connection is secure. In the next part we will describe how the user is able to access our testbed and what happens behind the scene.



Figure 3: LOG-a-TEC schematic

There are two different ways to communicate with the LOG-a-TEC testbed. First is by using the web portal (Figure 4) and the second one is to write custom code using exposed HTTP API. The web portal offers full functionality to the user and HTTP API allows more freedom to the developer/experimenter. The web app is split in two parts first part is Google maps where we can observe nodes locations and the second part is used to interact with the testbed. Features of the second part of the web app are:

* Choosing the cluster where three clusters are available one on IJS and two in Logatec first the industrial zone cluster and second the city center cluster. Each cluster needs its own SSL server listening on the infrastructure and by changing the cluster we make a switch to different SSL server.
* The experiment can be described in a simple text file in which we specify the GET and POST requests which form our experiment. The commands are separated by an empty line.
* All the requests and all responses are stored in the request-response log file thus we collect the spectrum sensing data in format that corresponds to CREW common data format.
* There is also an option to send single request to the sensor network. As mentioned above there are two methods GET and POST.
* The last part of the portal is remote reprogramming. There is an option to select a binary file from users local environment and upload it to the server where a special script, written to send the file to the gateway, cuts the file into packets of 512 bytes. Each packet gets a header of 4 bytes containing the serial number of the packet and the footer of 4 bytes with the CRC of the packet. The finished packet is 520 bytes long and gets transferred to the gateway which forwards the packet to the end node. The ZigBit packets were therefore extended by an additional layer which can handle packets of this size. The packets are stored on an SD card which is devided into several slots each able to store one firmware image. After the transfer is complete the node is set to boot the firmware from the requested SD card slot and rebooted. After a reboot the new firmware is loaded in the flash and started.
* Reset SSL connection button is a safety button in case something unexpected happen

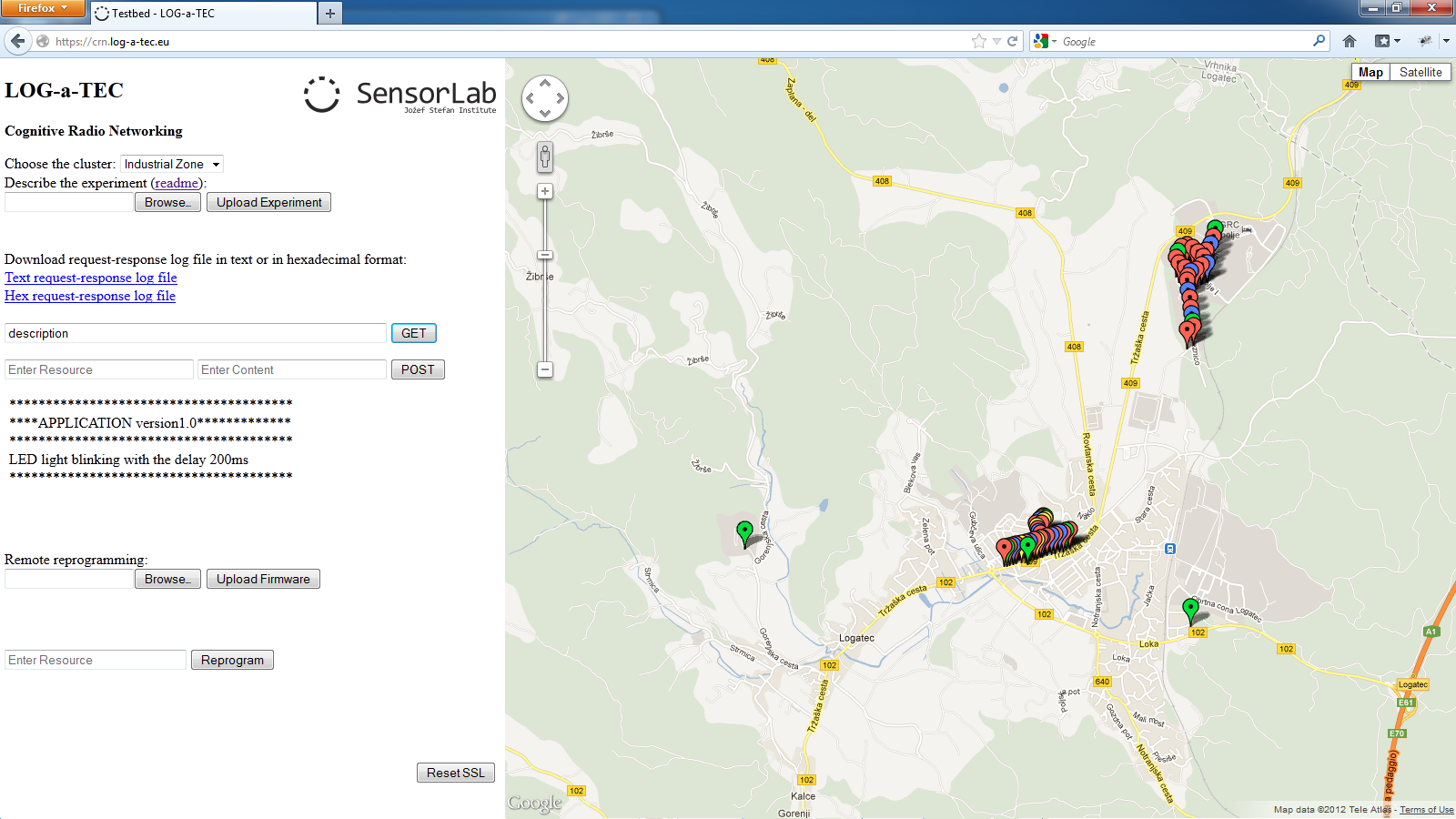


Figure 4: LOG-a-TEC Testbed web portal

The other way of using the LOG-a-TEC testbed is by calling the HTTP API. The call to the API has to meet the specified form for GET (Figure 5) and POST (Figure 6) request.

https://crn.log-a-tec.eu/communicator?cluster=”port”&method=get&resource=”resource”

Figure 5: HTTP API GET request

To make a GET or POST request we have to make a call to handler called “communicator” located on the web server on LOG-a-TEC infrastructure with the domain name crn.log-a-tec.eu. The call is made over secure HTTPS encrypted socket with authentication required. The parameters of the call have to be:

* Cluster – this corresponds to the SSL port which the cluster gateway is connected to. The JSI cluster is connected to the port 9501, Industrial Zone to 10001 and City Center to 10002.
* The second parameter is method which can be GET or POST
* GET requests last parameter is resource. This corresponds to the resource name located on the target node from one of the clusters.
* The POST request includes also content where we specify the reconfiguration parameters for the nodes

Requests that do not meet the form will be rejected.

https://crn.log-a-tec.eu/communicator?cluster=”port”&method=post&resource=”resource”&content=”content”

Figure 6: HTTP API POST request

We can access the portal from [www.log-a-tec.eu](http://www.log-a-tec.eu/). Hire is the link which points to the portal where we need to first authenticate to be able to enter. For now we can manually add the authentication parameters for interested experimenters although we are preparing an automated subscription system with scheduler for the experimenters.