APPLICATION NOTE

SCA11H FLASK SERVER

General Description

This document describes a simple Flask server. It can be used as a cloud server on a PC where SCA11H BCG sensor node can be connected to. The method is described in the Python example code included at the end of this document. The code requires the user to have Flask server v0.11 or newer.

The code has two simple functions: logging of BCG data to a .txt file, and reading and changing the calibration parameters through the over-the-air (OTA) functionality. The latter requires that the Automatic Firmware Upgrade option is enabled in node configuration. Raw data is not supported in cloud mode.

The server automatically logs received data to loggedData_<unixtime>.txt in the root folder. The old parameters provided by the node, when OTA message is triggered, along with the new randomized parameters that are being sent to the node, are logged in to parameterSync_<unixtime>.txt.

Example Python code

```
# Python >3.5 and Flask >0.11 required

# THIS SOFTWARE IS PROVIDED BY MURATA "AS IS" AND
# ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
# LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
# FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
# MURATA BE LIABLE FOR ANY DIRECT, INDIRECT,
# INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
# (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
# SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
# HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
# STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
# IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
# POSSIBILITY OF SUCH DAMAGE.

from flask import Flask, request, Response, make_response
from werkzeug.serving import WSGIRequestHandler
from werkzeug.datastructures import Headers
import random, time, string

HOST = '0.0.0.0'  # Use host 0.0.0.0 to allow access from all devices
PORT = 80

app = Flask(__name__)
WSGIRequestHandler.protocol_version = "HTTP/1.1"
dataLogFilename = 'loggedData_' + str(round(time.time())) + '.txt'
paramsLogFilename = 'parameterSync_' + str(round(time.time())) + '.txt'

#Root page
@app.route('/')
def index():
    return 'Flask server is running!'

#/data/push/ folder, accepts only POST messages
#Receives data and logs it to a file
@app.route('/data/push/', methods=['POST'])
def receive_data():
    dataLogFid = open(dataLogFilename,'a')
    dataLogFid.write(request.data.decode())
    dataLogFid.close()
    return 'done'

def newparams():
    #Calculate random new calibration parameters, replace with own if so preferred
    var_level_1 = random.randrange(1000,9999,1)
    var_level_2 = random.randrange(100,800,1)
    stroke_vol = random.randrange(2600,6000,1)
    tentative_stroke_vol = 0
    signal_range = round(stroke_vol / 2.6)
    to_micro_g = 7
    return([var_level_1,var_level_2,stroke_vol,tentative_stroke_vol,signal_range,to_micro_g])

#/firmware/device/<mac>/ page, accepts GET and POST
#Implements OTA update function but utilizes only parameter updates
#Logs old parameters sent by BCG Sensor Node and responses with random parameters
#Can be modified to return address to downloadable firmware file etc.
@app.route('/firmware/device/<mac>/', methods=['GET','POST'])
def show_device_mac(mac):
    #Log old parameters sent by BCG Sensor Node
    paramsLogFid = open(paramsLogFilename,'a')
    t = time.strftime("%a, %d %b %Y %H:%M:%S +0000", time.gmtime())
    paramsLogFid.write(t + '\n')
    content = request.json
    oldParameters = content['pars']
    paramsLogFid.write(mac + ' old Parameters: ' + oldParameters + '\n')
    #Create new parameters
    newparameters = newparams()
    #Create response message
    strResp = '{\"pars\":\"'
    strResp += str(newparameters[0]) + ','
    strResp += str(newparameters[1]) + ','
    strResp += str(newparameters[2]) + ','
    strResp += str(newparameters[3]) + ','
    strResp += str(newparameters[4]) + ','
    strResp += str(newparameters[5]) + '\"}'
    #Log new parameters
    paramsLogFid.write(mac + ' new Parameters: ' + strResp[9:-2] + '\n')
    #Create chunked response
    respData = '{:X}\r\n{:s}\r\n0\r\n\r\n'.format(len(strResp),strResp)
    resp = make_response()
    resp.mimetype = 'application/json'
```

```
    resp.data = (respData)
    resp.headers.set('Content-Type', 'application/json')
    resp.headers.set('Transfer-Encoding' , 'chunked')
    resp.headers.set('Content-length',None) #Ensure no 'Content-length' is sent with chunked!
    paramsLogFid.close()
    return(resp)

if __name__ == "__main__":
    app.debug = True    #Set debug mode on/off, debug mode is insecure!
    app.run(HOST,PORT)
```

Rev.	Date	Change Description
1	9-June-17	First version of document.