APPLICATION NOTE

SCA11H DATALOGGER SAMPLE CODE

General Description

This document describes a simple method for logging data output of the SCA11H BCG sensor node. The method is described based on Python example code, which is included at the end of this document.

The example code logs the data output of the BCG node in to a .txt file. It can log both BCG and raw data outputs. The code functions only when the BCG is working in local mode. For cloud mode data logging, see Flask server documentation.

The code includes a SSDP search function (ssdpSearch()) that will search the current network for any connected nodes. It is described further in the SSDP discovery documentation provided elsewhere.

Data format

BCG data

```
20,68,16,47,143,2021,1,595,456,0(CR)
(CR,LF)
21,68,16,47,143,1055,1,595,456,0(CR)
(CR,LF)
```

Raw data

```
025a,025a,025b,025b,025b,025c,025c,025c,
```

Example Python code

```
# 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.


# Python 3.x

import msvcrt       # Windows only!
import socket
import time
import datetime

REPEAT = 10

def ssdpSearch():
    print("Starting SSDP Search. 10 seconds.")
    UDP_IP = '<broadcast>'
    UDP_PORT = 2000
    UDP_MESSAGE = '{"type":"SCS-DISCOVER","hostname":"Host-SCS"}'
    networks = socket.gethostbyname_ex(socket.gethostname())[2] # Find all networks (i.e, wifi, wired)
    sockets = []
    for net in networks:    # Connect to all networks
        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)     # UDP
        sock.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)  # Allow broadcast
        sock.bind((net, UDP_PORT))  # Connect
        sock.settimeout(1.0)        # Set timeout (if no answer when reading)
        sockets.append(sock)        # Save "sock" to sockets
    timeStart = time.time()
    devices = []
    print('Found devices:')
    time.sleep(0.1)
    while time.time() - timeStart < REPEAT:
        for sock in sockets:
            try:
                sock.sendto(UDP_MESSAGE.encode(), (UDP_IP, UDP_PORT))
                data, addr = sock.recvfrom(1024)
                data = data.decode()
                data = data[1:].split(',')
                if data[0] == '"type":"SCS-NOTIFY"':    # Only accept correct responses
                    oldDevice = 0
                    # print(data)
                    for dev in devices:
                        if dev[0] == data[1]:
                            oldDevice = 1
                    if not oldDevice:
                        devices.append([data[1],data[2]])   # Save found devices
                        print('\t' + data[1] + ' ' + data[2])
            except:
                1
        time.sleep(0.2)
    if not len(devices):
        print('\tNo devices found.')
    print('')
    for sock in sockets:
        sock.close()

def readLine(s):
    # Function to read status from BCG data
    line = s.recv(1024).decode()
    return line


def main():
    print('BCG Data Logger\nLogs either raw data or BCG algorithm data to a file \
depending on configured mode.')
    # Open file
    IP = input('Insert IP address (empty for SSDP): ')
    while len(IP) == 0:
        ssdpSearch()
        IP = input('Insert IP address (empty for SSDP): ')
    PORT = 8080
    filename = 'logged_data_' + str(datetime.datetime.now().strftime('%Y-%m-%d_%H-%M-%S')) + '.txt'

    fid = open(filename,'w')
    print('')
    print('Starting to read data. Press \"ctrl+c\" to quit.')
    while True:
        try:
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            s.settimeout(10)
            s.connect((IP, PORT))
            while True:     #
                data = readLine(s)
                print(data,)
                fid.write(data)
        except (KeyboardInterrupt, SystemExit):
            print('Exiting program.')
            fid.close()
            break
        except (socket.timeout):
            print('Timed out, reconnecting.')
        except socket.error as msg:
            print(msg)
            print('Trying to reconnect.')

if __name__ == '__main__':
    main()
```

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