

APPLICATION NOTE



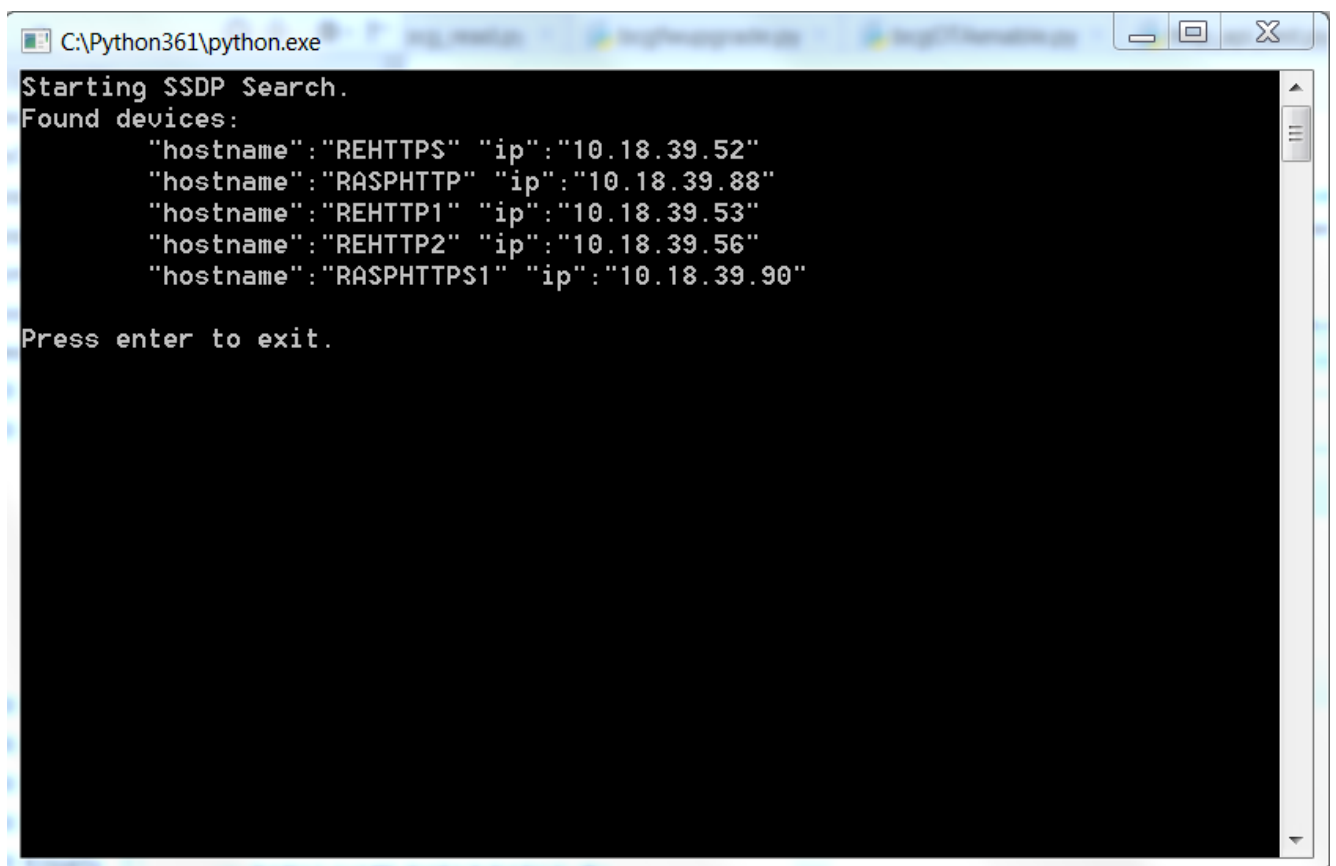
SCA11H SSDP DISCOVERY SAMPLE CODE

General Description

This document describes a simple SSDP (*Simple Service Discovery Protocol*) discovery function that can be used to search a local network for connected SCA11H BCG sensor nodes. The method is described based on Python example code, which is included at the end of this document.

The example code is a SSDP search function that uses BCG node specific messages in order to search for only BCG nodes connected to the network. The function repeats itself 10 times and returns the IP addresses and node ID's of the nodes connected to the current local network.

Example code run:



```
C:\Python361\python.exe
Starting SSDP Search.
Found devices:
  "hostname": "REHTTPS" "ip": "10.18.39.52"
  "hostname": "RASPHTTP" "ip": "10.18.39.88"
  "hostname": "REHTTP1" "ip": "10.18.39.53"
  "hostname": "REHTTP2" "ip": "10.18.39.56"
  "hostname": "RASPHTTPS1" "ip": "10.18.39.90"

Press enter to exit.
```

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 (tested with 3.5)

import socket
import time

REPEAT = 10

def ssdpSearch():
    print("Starting SSDP Search.")
    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()

if __name__ == '__main__':
    ssdpSearch()
    try:
        input("Press enter to exit.")
    except KeyboardInterrupt:
        pass

```

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