

CONNECTING A SENQUIP DEVICE TO UDP OVER WIFI

1. Introduction

Senquip devices can connect to LTE or Wi-Fi networks and send data to the Senquip Portal via secure MQTT. Senquip devices can also send directly to 3rd party endpoint over UDP, MQTT, and HTTP. Certificates can be uploaded to secure MQTT and HTTP communications.

The use of UDP endpoints is not recommended on public networks due to security risks. UDP can however be useful for low overhead communications on secure Wi-Fi networks for communication with dashboards on closed networks, and for debugging.

This application note discusses the connection of a Senquip device to Wi-Fi using the embedded webserver and the Senquip Connect App. It will explain how to connect to a UDP endpoint and how to send custom messages to that endpoint.

For more information on Senquip devices, see the Senquip [ORB](#) and [QUAD](#) User Guides. For more information on scripting, see the Senquip [Scripting Guide](#).

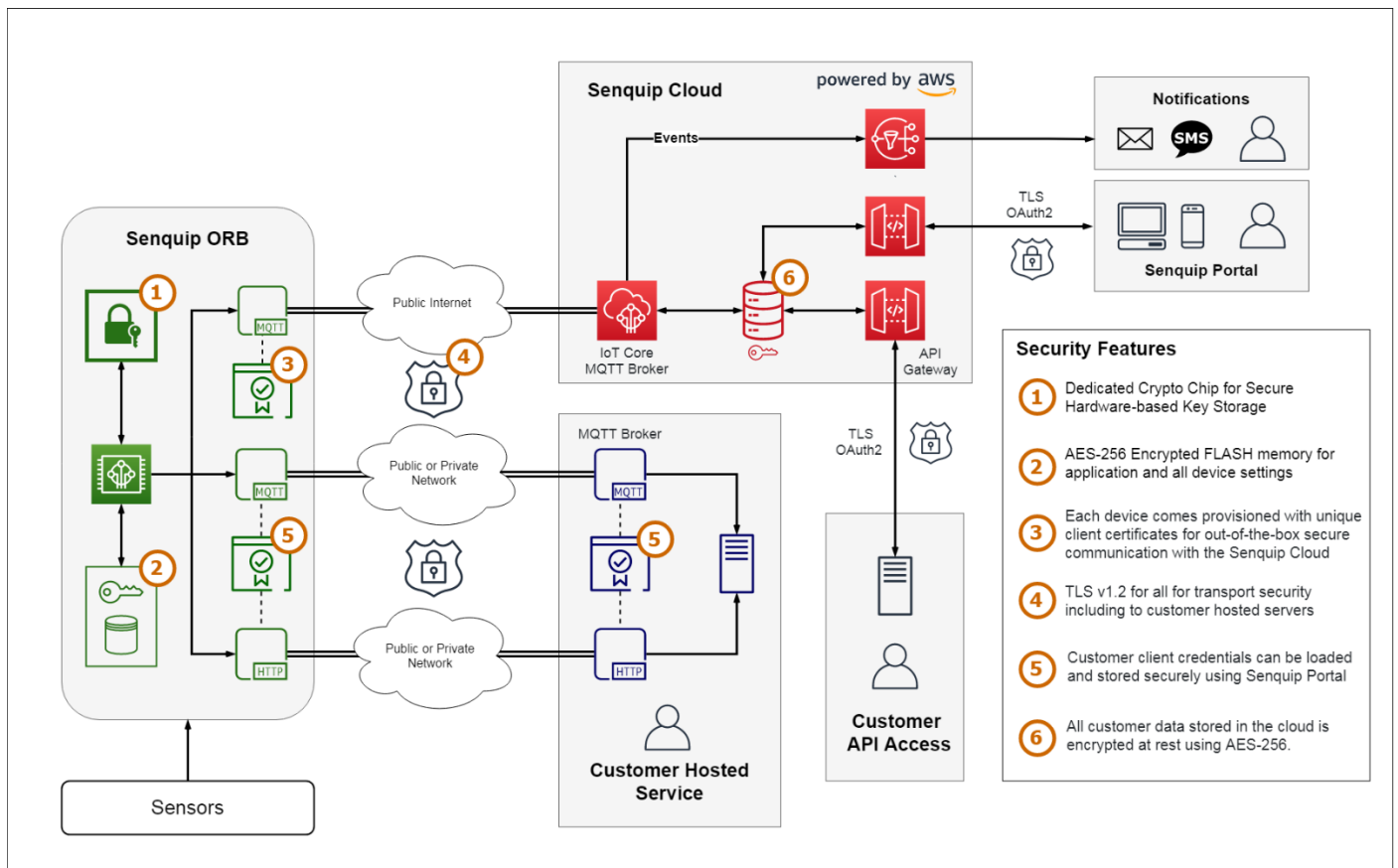


Figure 1 - Senquip Cloud Architecture

2. Connecting to a Wi-Fi Network

Senquip devices can all connect to 2.4GHz Wi-Fi networks. The Wi-Fi antenna is internal, and no external antennas are required. It is always recommended that Senquip devices are connected to a network as soon as possible, and that further configuration is then performed on the [Senquip Portal](#).

2.1. Using the Embedded Webserver

All Senquip devices have an embedded webserver that allows for configuration of settings, including Network settings. To enable the embedded webserver, place the Senquip device in setup mode by pressing the setup button. The green light will start to flash, and a Wi-Fi hotspot will be enabled. Using a phone, tablet, or computer, search for the device hotspot and connect using the Wi-Fi password. Open a browser and navigate to 192.168.4.1. Use the username “admin”, and the setup password to gain access to the device webserver.

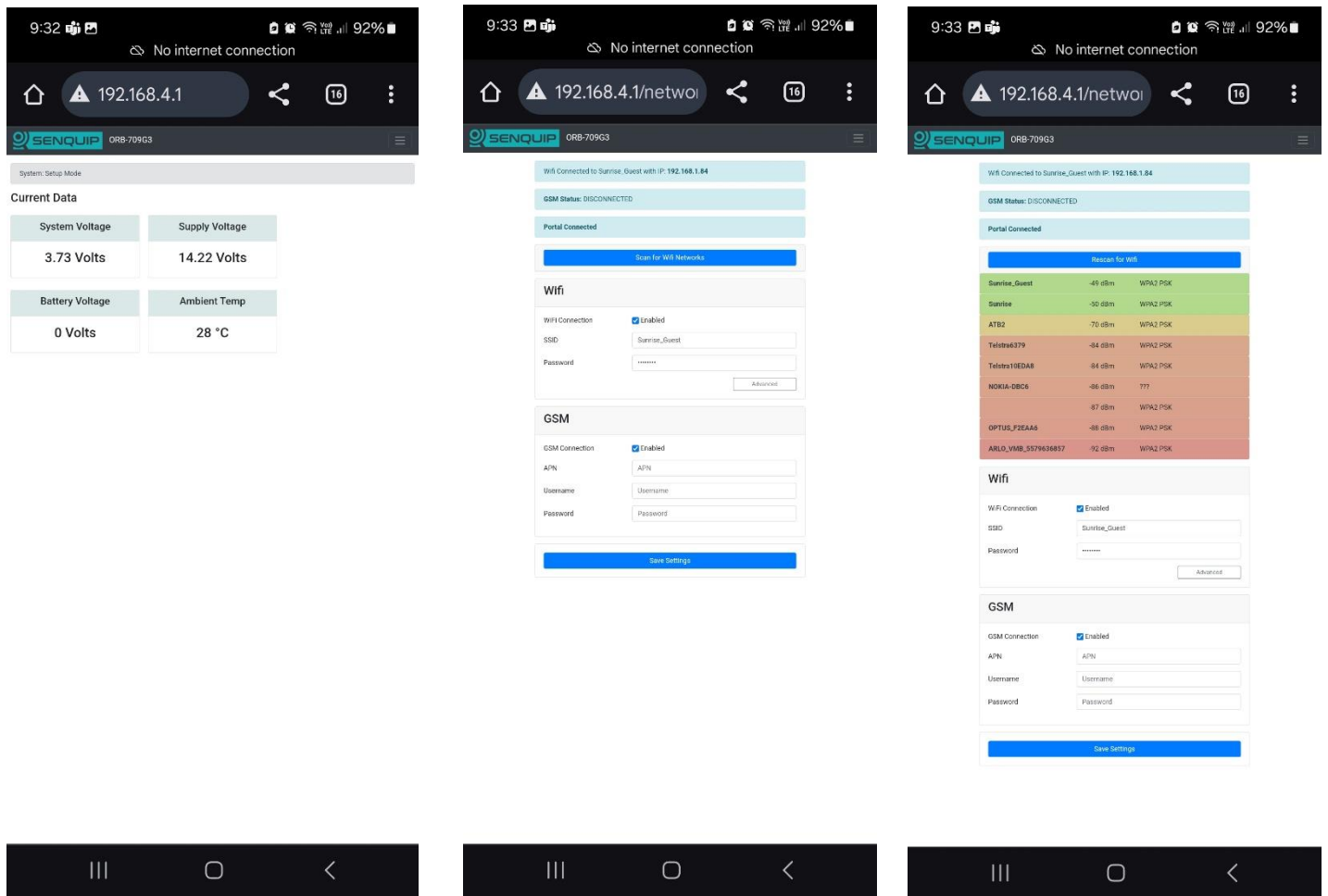


Figure 2 - Embedded Webserver

Use the menu option in the top right to navigate to *Settings>Network*. Enter the preferred Wi-Fi SSID and password or use the scan for networks to see available Wi-Fi networks. Press save. To exit setup mode, press the reset button on the Senquip device.

2.2. Using the Senquip Connect App

The Senquip Connect app is a mobile application that allows users to connect to nearby Senquip devices using Bluetooth to configure networks and for data viewing. The app is available for both Android and iOS devices running recent firmware. See the [Senquip Device Firmware Changelist](#) for more information on compatible firmware versions.

Network configuration via Bluetooth is available when the Senquip device is in setup mode. To enter setup mode, press the setup button on the device. The green light will start to flash. Open the Senquip Connect App and scan for available devices. Device in setup mode will allow connection for the purpose of network configuration. Press connect and enter the device setup password.

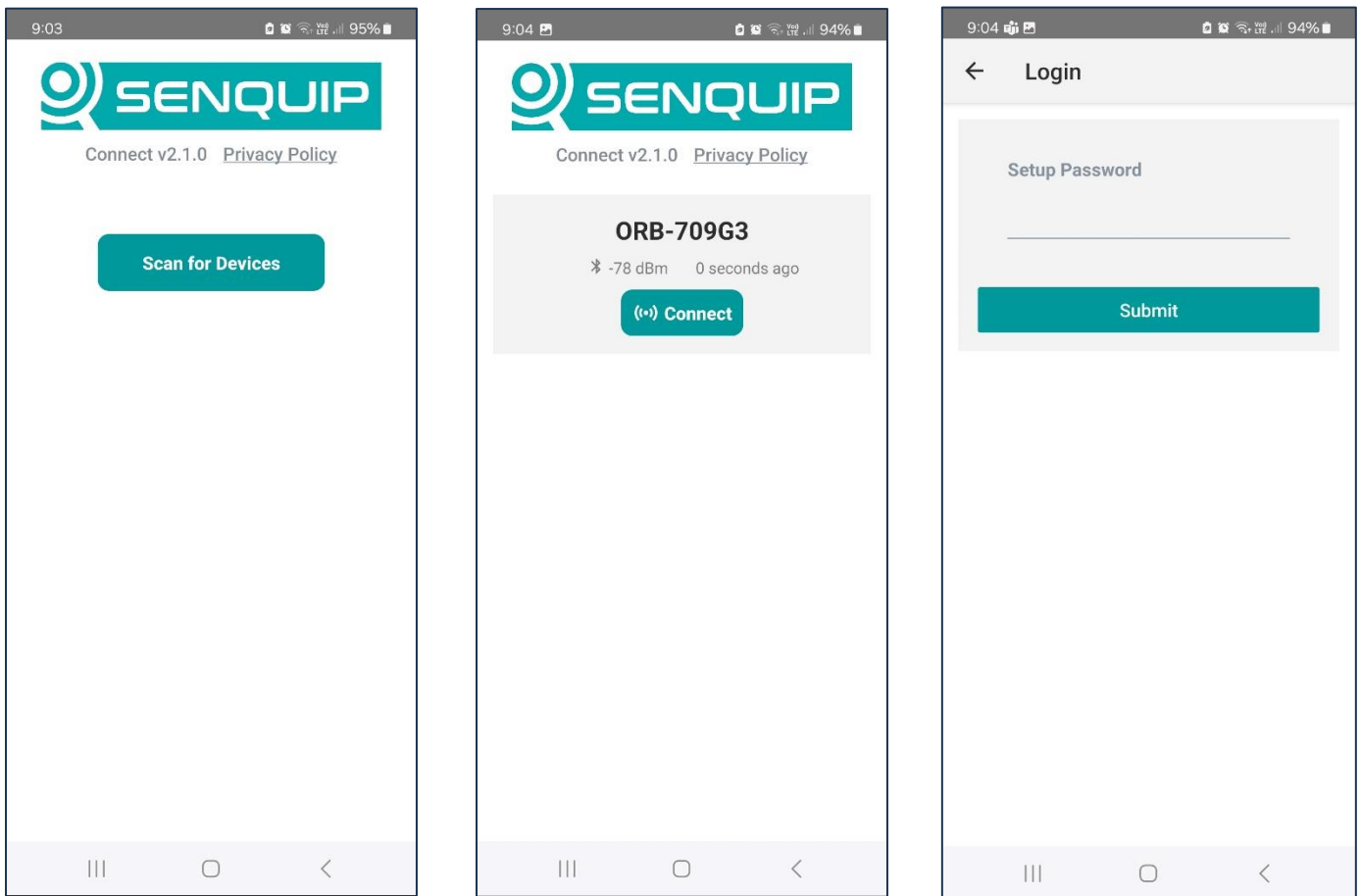


Figure 3 - Scan and connect to devices

To set the Wi-Fi options, select *Configure WiFi*, enable it and populate the SSID and Wi-Fi password and select *submit*. To exit setup mode on the device, press reset.

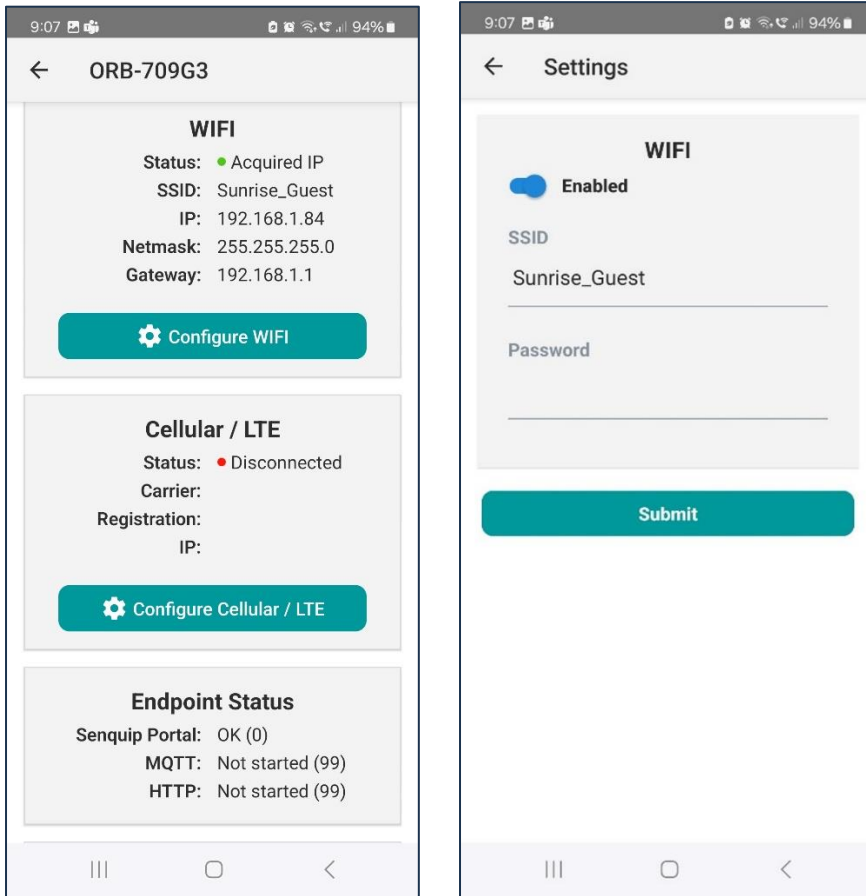


Figure 4 - Configure Wi-Fi

Now that the Senquip device is connected to a network, the orange light should go solid, and data should start arriving on the Senquip Portal. We will continue configuration of the UDP endpoint on the Senquip Portal.

3. Configuring a Simple UDP Endpoint

In this instance, the Senquip device will be a UDP client, and will send messages to a UDP server. In this application, we will use a UDP server called “Udp Client Server” by [NSauditor](#). The client server has determined that it has an IP address of 192.168.1.49. We have specified the port as 11000. The IP address field should be used if you want to send packets to the ORB. It will not be used in this application. Once you have entered your port number, press “Start Server” to begin.

Note that the Senquip device must be on the same network as the UDP server, and that there must be no firewalls blocking access to port 11000.



Figure 5 - UDP Client Server

We will now configure the Senquip device UDP endpoint to match the values specified in the server: Ip address 192.168.49 and port 11000. To configure the Senquip device, use the *Endpoint* tab on the device settings page.

Data Endpoints

Configuration via Senquip Portal	<input checked="" type="checkbox"/> Enabled
Send Data to Senquip Portal	<input checked="" type="checkbox"/> Enabled
Offline Buffer	<input type="checkbox"/> Enabled
Add Formatted Time	<input type="checkbox"/> Enabled
Report Network Info	<input checked="" type="checkbox"/> Enabled
Use Senquip Data Format	<input checked="" type="checkbox"/> Enabled

UDP

UDP Enabled

UDP Address

Figure 6: Senquip Device UDP Settings

Senquip devices send data to the Senquip Portal securely, using MQTT, and in JSON format. A standard Senquip data message in JSON format is shown below. By default, if an endpoint is enabled, this JSON packet will be sent to that endpoint on each base interval.

```
{
  "ts": 1608416763.9,
  "ambient": 25.83,
  "current1": 0.16,
  "angle": 1.7,
  "pitch": -0.9,
  "motion": 24,
  "wifi_ip": "192.168.1.178",
  "deviceid": "BD6AEDJQ2",
  "vreg": 8.98,
  "vsys": 4.09,
  "vbat": 0,
  "wifi_rssi": -32,
  "light": 1,
}
```

In Figure 7, we see the Senquip JSON data arriving at the UDP server.

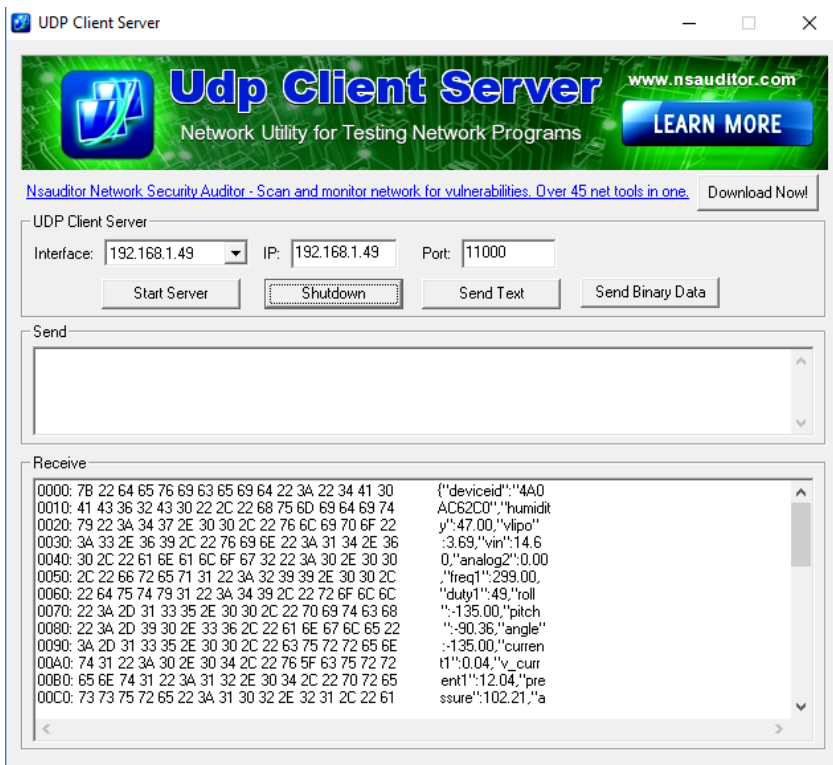


Figure 7 - Standard Senquip JSON Packet Arriving in UDP Server

4. Creating Custom Messages in a Script

Where the Senquip device is sending to a 3rd party dashboard, the standard Senquip JSON format data will not be compatible with what the endpoint is expecting. In this instance, a custom message will need to be generated and sent from within a script running on the Senquip device.

To disable the sending of the standard Senquip JSON message via 3rd part endpoints, the *Use Senquip Data Format* option in the endpoint settings should be unselected.

In a script, we generate a typical custom 3rd party data message and send it to our endpoint.

```
load('senquip.js');
load('api_endpoint.js');
load('api_config.js');

SQ.set_data_handler(function(data) {
  let obj = JSON.parse(data);

  let udp = "100,"+Cfg.get('device.name') + "\r\n";
  if (typeof obj.ambient === "number") {udp = udp + "211,"+JSON.stringify(obj.ambient) + "\r\n";}
  if (typeof obj.vin === "number")      {udp = udp + "212,"+JSON.stringify(obj.vin) + "\r\n";}

  UDP.send(udp, udp.length);
}, null);
```

Document Number APN0002	Revision 1.2	Prepared By NGB	Approved By NB
Title Connecting a Senquip Device to a UDP Server Over WiFi			Page 8 of 9

We include the config library to allow the use of the Cfg.get function that gets the device name. We then assemble a message, using measurements available in the data object, and escaped characters where required. The message is then sent to the configured endpoint using the UDP.send function. The data handler will run after each measurement cycle and so the message will be delivered at the base interval (assuming the transmit interval is 1).

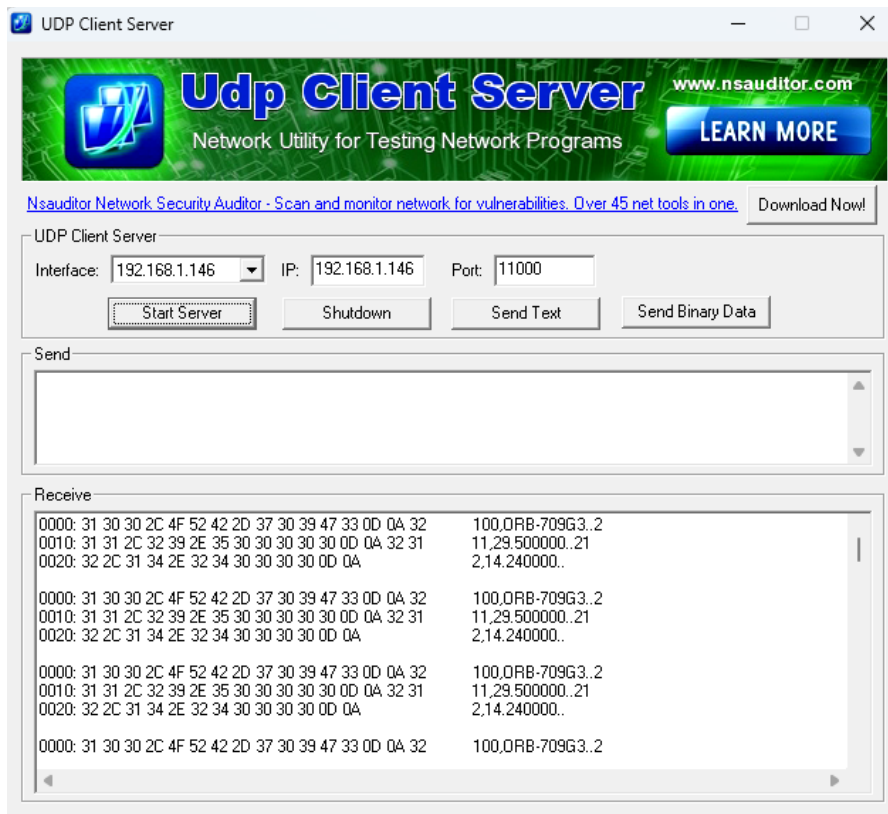


Figure 8 - Custom Message Arriving at Endpoint

For more details on how to write your own script, please see the [Senquip Scripting Guide](#).

5. Using UDP for Debugging Scripts

In the same way that scripts are used to create custom messages that are sent via UDP from within a script, debug messages can be generated in a script and sent to an endpoint to assist in debugging scripts. To facilitate this, a simple function that sends data to a debug log is created.

```
load('senquip.js');
load('api_endpoint.js');

function ulog(s) { UDP.send(s); }

ulog("Script Start");
```

Note: to be able to log to UDP, the UDP server needs to have had time to start. Messages sent before the UDP server has started will be lost. In the script above, it is likely that the UDP server will not have started by the time the "Script Start" messages it sent.

Document Number	Revision	Prepared By	Approved By
APN0002	1.2	NGB	NB
Title			Page
Connecting a Senquip Device to a UDP Server Over WiFi			9 of 9

To facilitate the logging of debug information, Senquip has created a simple Python [UDP server](#). Before using the Python script, change the server Ip address and port as required and set the endpoint on the Senquip device to match.

```
import logging
import socket

log = logging.getLogger('udp_server')


def udp_server(host='192.168.1.146', port=23):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

    log.info("Listening on udp %s:%s" % (host, port))
    s.bind((host, port))
    while True:
        (data, addr) = s.recvfrom(128*1024)
        yield data

FORMAT_CONS = '%(asctime)s %(name)-12s %(levelname)8s\t%(message)s'
logging.basicConfig(level=logging.DEBUG, format=FORMAT_CONS)

for data in udp_server():
    log.debug("%r" % (data,))
```

In Figure 9 we see the custom packet that we created in Section 4 being displayed as a log.



```
C:\Windows\System32\cmd.e  X  +  v
Microsoft Windows [Version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

C:\>py udp_server_v1.py
2024-09-17 11:26:15,889 udp_server INFO Listening on udp 192.168.1.146:23
2024-09-17 11:27:25,357 udp_server DEBUG b'100,ORB-709G3\r\n211,29.500000\r\n212,14.240000\r\n'
2024-09-17 11:27:35,274 udp_server DEBUG b'100,ORB-709G3\r\n211,29\r\n212,14.240000\r\n'
```

Figure 9 - UDP Logging Using Senquip Python Script

6. Conclusion

Connecting a Senquip device to a Wi-Fi network using either the webserver or Senquip Connect App is quick and easy. You can forward standard Senquip JSON messages to any server or SCADA system using UDP, HTTP/S or MQTT/S. Customised messages can be created and sent from within a script. UDP can be useful as a script debugging tool.