Doodle Labs Smart Radio Serial Interface Guide

Advanced Mesh Router for Resilient Private Wireless Networks

Introduction

The Smart Radio runs a customized version of Openwrt with enhancements related to the Doodle Labs Mesh Rider® technology. These enhancements are useful for applications requiring low-latency command-and-control transmission and high-throughput video - e.g. UAV and robotics.

The purpose of this guide is to aide a programmer in configuring the Smart Radio Serial interface. Communications over the Smart Radio's wireless interface are IP based, so it is necessary to relay packets from the network to the serial interface. This guide includes examples for how to send data from the serial interface of one radio to that of another radio, or to simply send packets directly over the network to the serial interface. The guide is organized in the following sections:

Hardware Setup

Doodle Labs currently offers four main hardware variants in its Smart Radio lineup. UART is available on all hardware variants. However, these hardware variants have different pinouts and you should refer to the documentation package of your specific hardware for pinout information. The hardware versions can be differentiated by the model number. In all cases, the UART is implemented as one TX line, one RX line, and GND.

Please note that -

1. The legacy models, RM-****-2H-*U have a USB host port on the auxiliary connector, and have no UART port. For the UART port, you require RM-****-2H-***S** model.

Connection Guide

UART Connector

- The UART connector is a 3-pin connector consisting of TX, RX, and GND. The TX and RX wires should be coupled closely with the GND wire for good signal integrity.
- Typically when two devices are connected over UART, the TX wire of the first device should be connected to the RX wire of the other device and vice versa.
- While most models use 3.3-V TTL signalling, the External Smart Radio uses RS232 signalling. Please consult the documentation package of your hardware before hooking up the UART.

If you are bridging the UART port to the Mesh Rider network, then you will also need to use the main network interface of your device. For -H, and -J hardware, this is ETH0, and for -K and -L hardware it is the USB data interface. For the remainder of the document we will use generic diagrams for the Smart Radio with one Ethernet port and one UART port.

Serial Interface Configuration

Start with one of the hardware setups in the section above. A generic setup is shown in Figure 1.

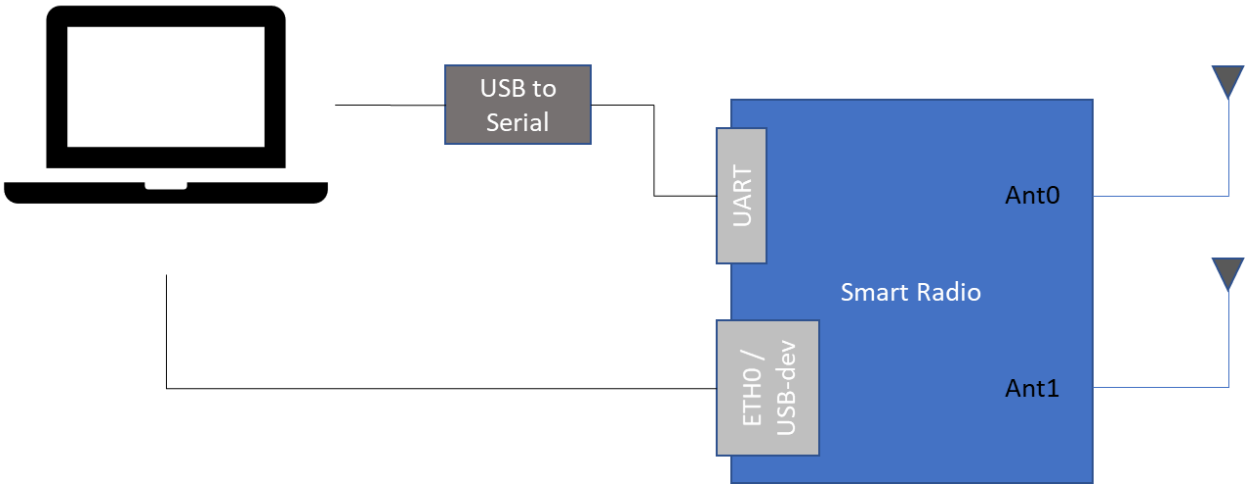

Figure 1 – Generic UART hardware connection

Configuring the UART port using the GUI

The GUI is strictly for configuring the UART port as a Serial to Network bridge. Open up a web browser and navigate to the IP address of the Smart Radio. From there, navigate to `Services → Serial Configuration`. A screenshot of the GUI is shown in Figure 2. For mobile applications, **we recommend using UDP transport** over TCP transport as it is more responsive.

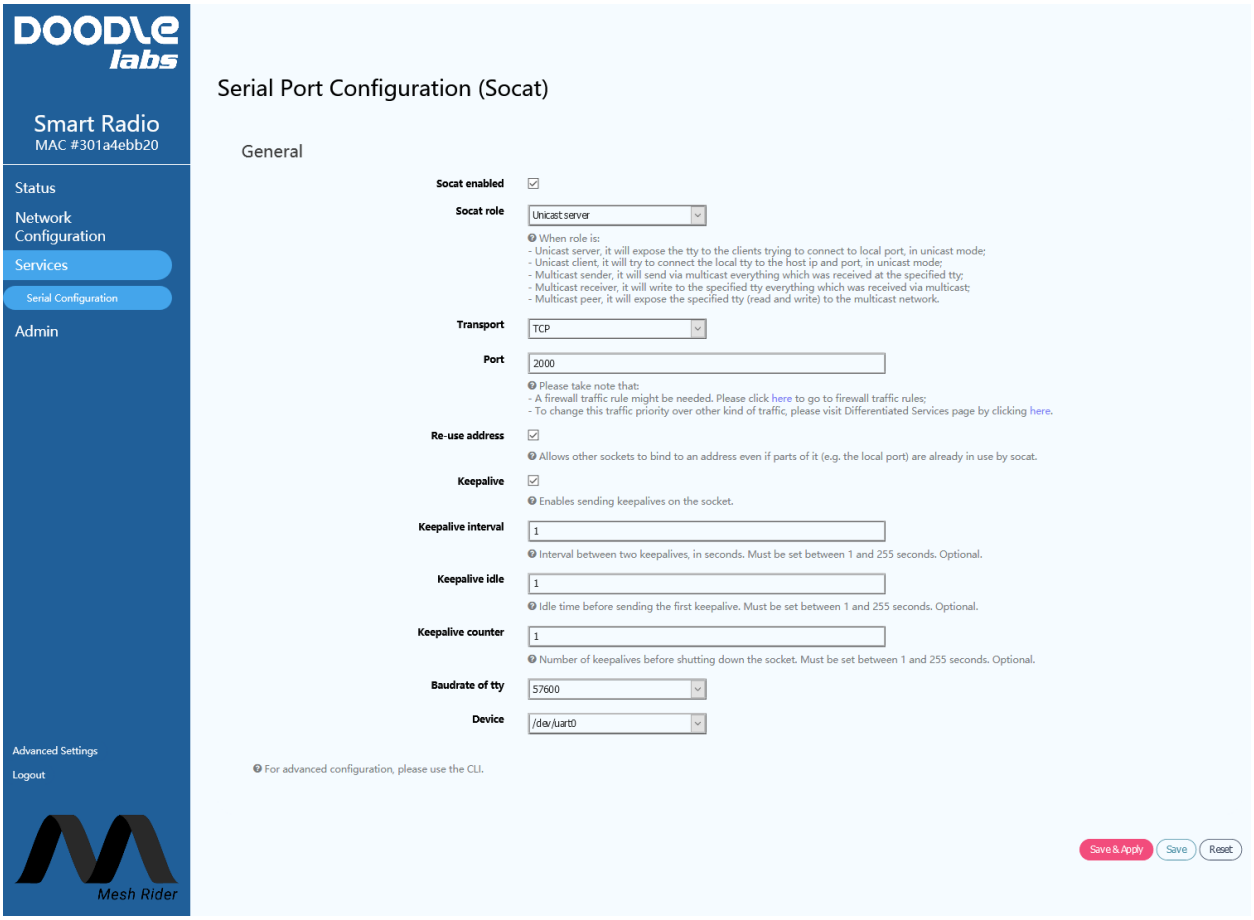

Figure 2 – UART Configuration GUI

When configuring the serial to network relay using the GUI, the program, `socat`, is used in the background. For unicast traffic, the relay should operate as either a server which listens on a particular network port for incoming connections, or a client which attempts to connect to a particular IP address and network port. For multicast traffic, the relay can either send, receive, or operate bi-directionally. **It will be necessary to open up the firewall at the desired network port, and a link to the firewall page is provided**. For multicast traffic, it is also necessary to update the routing table and a link is provided to the static route page. An example static route is shown in Figure 3.

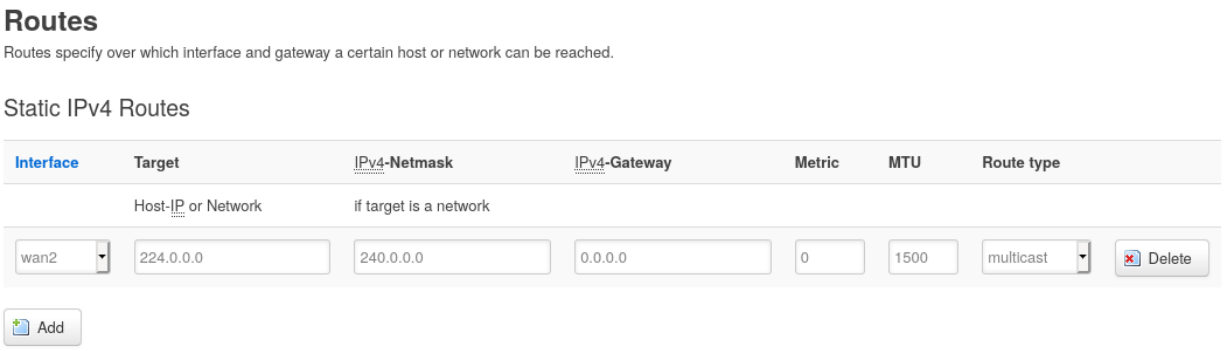

Figure 3 – Static route configuration for multicast traffic

Choose the device `/dev/uart0`. After making your changes, click `Save & Apply`. If you are familiar with `Socat` and want to use your own arguments, then you can input them using the checkbox at the bottom of the page.

The remainder of this section discusses configuration using the CLI.

Manually Configuration and Debugging the UART port

We recommend using the GUI for serial port configuration, but if you want an advanced configuration, then you may also use the CLI. This guide is not designed to be comprehensive, and if you plan on using the UART port manually, then you will need to have Linux expertise. We will discuss some of the tools which are built into the Smart Radio for you to use, and this section will also be useful as a debugging guide.

You can SSH into the radio by running

```
user@host-pc:~$ ssh root@<IP Address>
```

It may be necessary to update your PC's list of known hosts first.

```
user@host-pc:~$ ssh-keygen -R <IP Address>
```

The UART interface is attached to `/dev/uart0`.

Disable Socat

Before attempting to use the UART port manually, ensure that socat is disabled. SSH into the radio and run

```
root@smartradio:~# uci set socat.http.enable='0'
root@smartradio:~# uci commit
root@smartradio:~# /etc/init.d/socat restart
```

This can also be done in the web GUI.

STTY Configuration

The default UART settings can be changed using the `stty` program which is installed on the Smart Radio. An example usage is,

```
root@smartradio:~# stty -F /dev/ttyUSB0 115200 cs8 -cstopb -parenb
```

which sets the speed to 115,200 baud, 8 data bits, 1 stop bit and no parity.

(Please note that the Helix Smart Radio's UART port supports max 115,200 baud).

picocom

From the May 2021 firmware release onwards, `picocom` is installed by default on the Smart Radio. `picocom` is a terminal access program which can be used for general debugging of the UART port. You can invoke picocom with

```
root@smartradio:~# picocom -b 115200 /dev/uart0
```

You can exit picocom by holding down `CTRL`, and the hitting `a` and then `x`. For a full list of commands, just type `picocom`.

cat and echo

A single message can easily be sent on the UART port using `echo`

```
root@smartradio:~# echo -ne "Hello World\n" > /dev/ttyUSB0
```

or received on the UART port using `cat`

```
root@smartradio:~# cat < /dev/ttyUSB0
```

These programs use the settings applied by `stty` as defaults.

agetty

The program "agetty" can be used to open a serial console over the USB UART port. First access the Smart Radio over SSH, and then run

```
root@smartradio:~# agetty -8 115200 ttyUSB0 -n -l /bin/ash
```

socat and ser2net

`socat` and `ser2net` are installed on the Smart Radio, however, due to a `socat` bug, we strongy recommend that you use the web GUI for their configuration.

Startup Configuration

Openwrt uses the procd system for service startup, configuration and management [1]. To make a simple script which starts up on boot, and respawns when it closes, you can use the script below as a template.

```
root@smartradio:~# cat /etc/init.d/mystartupscript
#!/bin/sh /etc/rc.common
```

```
START=99
USE_PROCD=1

PROG=/opt/myscript.sh

start_service() {
        procd_open_instance
        procd_set_param command "$PROG"
        procd_set_param respawn 0 5 0
        procd_close_instance
}

stop_service() {
        checkproc=$(ps | grep "$PROG" | grep -v grep | awk '{print $1}')
        if [ ! -z "$checkproc" ]; then
          kill $checkproc
        fi
}
```

The `START` value indicates it's boot priority with 99 being the lowest (recommended). This template should be copied to the folder `/etc/init.d/` and enabled by running

```
root@smartradio:~# /etc/init.d/mystartupscript enable
root@smartradio:~# /etc/init.d/mystartupscript restart
```

At startup, the script `/opt/myscript.sh` will run. You can start, stop, restart, enable or disable the script using the `mystartupscript` init script.

References

[1] Openwrt procd init script parameters, https://openwrt.org/docs/guide-developer/procd-init-scripts, September 2021