Doodle Labs Smart Radio Serial Interface Guide

Advanced Mesh Router for 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 hardware versions can be differentiated by the order code. In -J hardware, the embedded version (-M) includes a UART port, while the external version (-E) includes RS232 level shifting. Example models include RM-5800-2J-XM and RM-4700-2J-XE. In -H hardware, models shipped with a -S order code, e.g. RM-915-2H-X**S**, include a UART port on the auxiliary connector. In all cases, the UART is implemented as one TX line, one RX line, and GND.

Pinouts

Doodle Labs currently offers four main hardware variants in its Smart Radio lineup – Embedded (-H and -J), External (-J only), and Pocketable (-J only). Figure 1 shows the location of the serial connector on the Embedded -H variant, and the pinout is included in Table 1.

© 2020 Doodle Labs. All rights reserved.

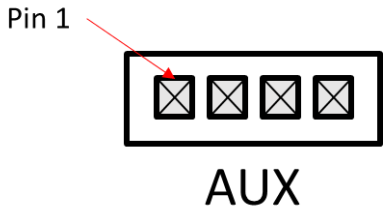

Figure 1 Embedded -H Pinout

Table 1 Embedded -H Pin Description

Pin Number	Direction	Voltage	Pin Description
1	I	+3.3-V	RFKill
2	-	-	GND
3	O	+3.3-V	UART_TX
4	I	+3.3-V	UART_RX

Figure 2 shows the location of the serial connector on the Embedded -J variant, and the pinout is included in Table 2. Note that if you are using the Eval Kit, then you can follow the guidelines for the External -J variant, but note that the voltage levels for the UART pins are +3.3-V.

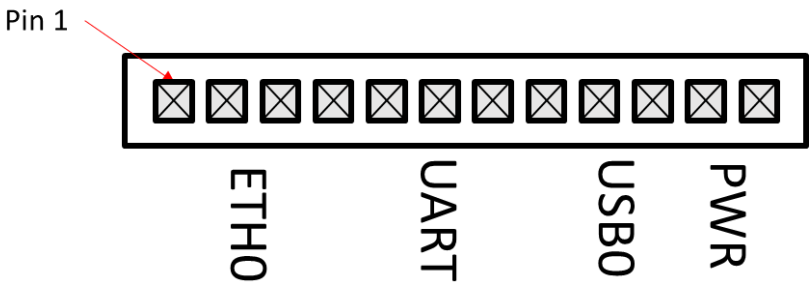

Figure 2 Embedded -J Pinout

Table 2 Embedded -J Pin Description

Pin Number	Direction	Voltage	Pin Description
1	O	Diff Signal	ETH0_TX-
2	O	Diff Signal	ETH0_TX+
3	I	Diff Signal	ETH0_RX-
4	I	Diff Signal	ETH0_RX+
5	-	GND	GND

© 2020 Doodle Labs. All rights reserved.

6	I	+3.3-V	UART_RX
7	O	+3.3-V	UART_TX
8	-	GND	GND
9	I/O	Diff Signal	USB0_DN
10	I/O	Diff Signal	USB0_DP
11	-	GND	GND
12	-	+6 V – +42 V	PWR

Figure 3 shows the location of the serial connector on the External -J variant, and the pinout is included in Table 3.

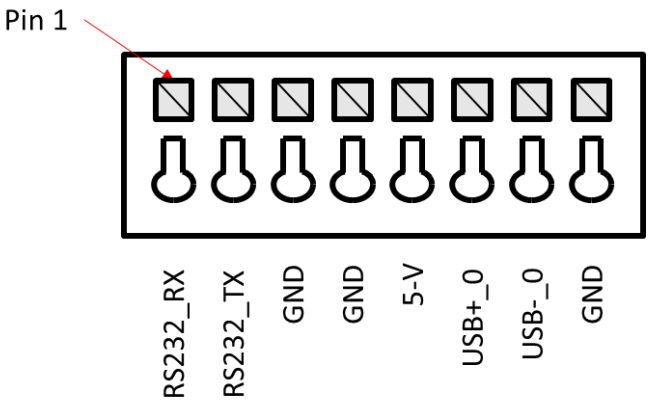

Figure 3 – External -J Pinout

Table 3 Embedded -J Pin Description

Pin Number	Direction	Voltage	Pin Description
1	I	-12 V - +12 V	RS232_RX
2	O	-12 V - +12 V	RS232_TX
3	-	GND	GND
4	-	GND	GND
5	O	+5 V	+5 V Output from Smart Radio
6	I/O	Diff Signal	USB0_DP
7	I/O	Diff Signal	USB0_DN
8	-	GND	GND

Figure 4 shows the location of the serial connector on the Pocketable -J variant, and the pinout is included in Table 4.

© 2020 Doodle Labs. All rights reserved.

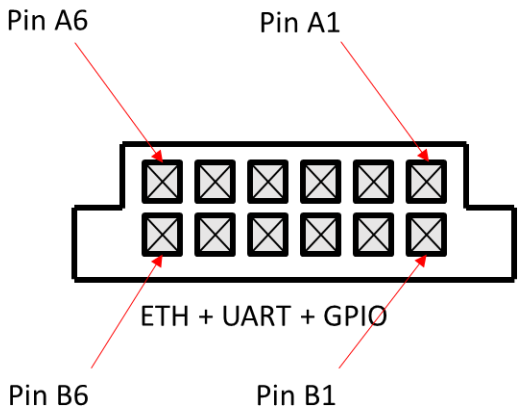

Figure 4 – Pocketable -J Pinout

Table 4 Pocketable -J Pin Description

Pin Number	Direction	Voltage	Pin Description
A1	-	GND	GND
A2	I	+3.3-V	UART_RX
A3	O	+3.3-V	UART_TX
A4	I/O	+3.3-V	GPIO1
A5	-	GND	GND
A6	-	-	N.C.
B1	I	Diff Signal	ETH1_RX+
B2	I	Diff Signal	ETH1_RX-
B3	O	Diff Signal	ETH1_TX+
B4	O	Diff Signal	ETH1_TX-
B5	-	-	N.C.
B6	-	-	N.C.

Connection Guide

Aside from the serial interface, you will also need to use the main Ethernet interface of your device (ETH0). For the -H and -J hardware variants, we will assume you are using the Evaluation Kits for your radios. Figures 5, 6 and 7 show how to setup the Smart Radio for quick evaluation of the UART port. For the remainder of the document we will use generic diagrams for the Smart Radio with one Ethernet port and one UART port. For the Pocketable hardware, you should use the USB DATA connector instead of the Ethernet connector.

© 2020 Doodle Labs. All rights reserved.

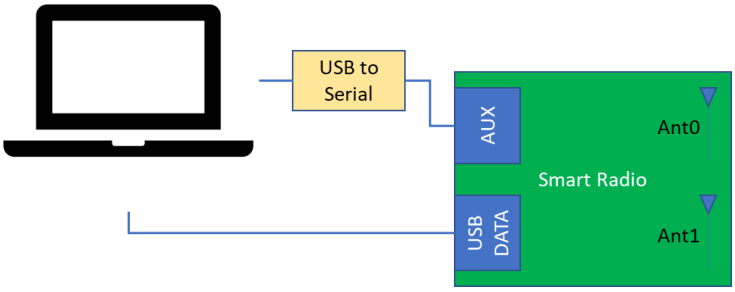

Figure 5 – -J Pocketable Hardware Variant

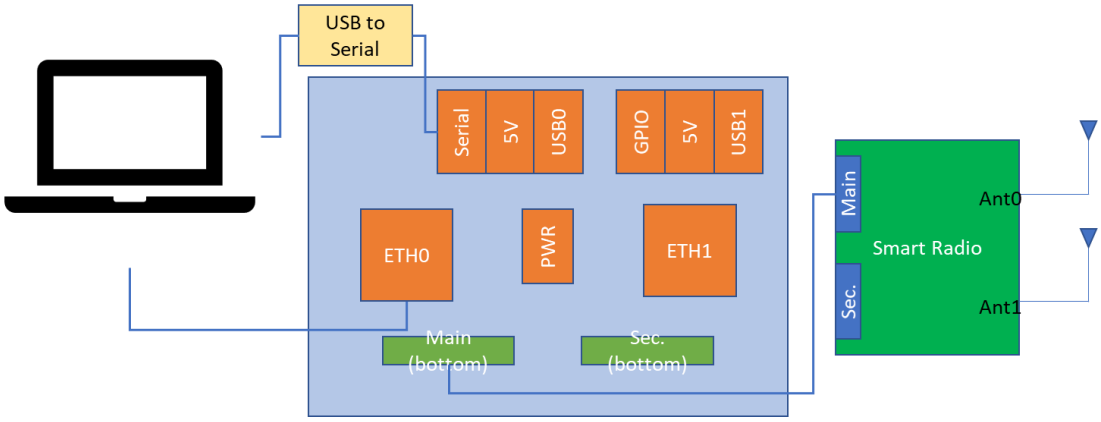

Figure 6 – -J Hardware Variant

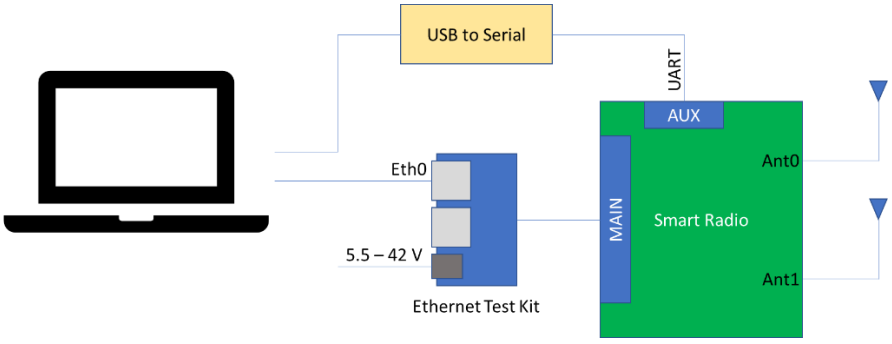

Figure 7 – -H Embedded Hardware Variant

© 2020 Doodle Labs. All rights reserved.

Serial Interface Configuration

Start with one of the hardware setups shown in Figures 5, 6, and 7. A generic setup is shown in Figure 8.

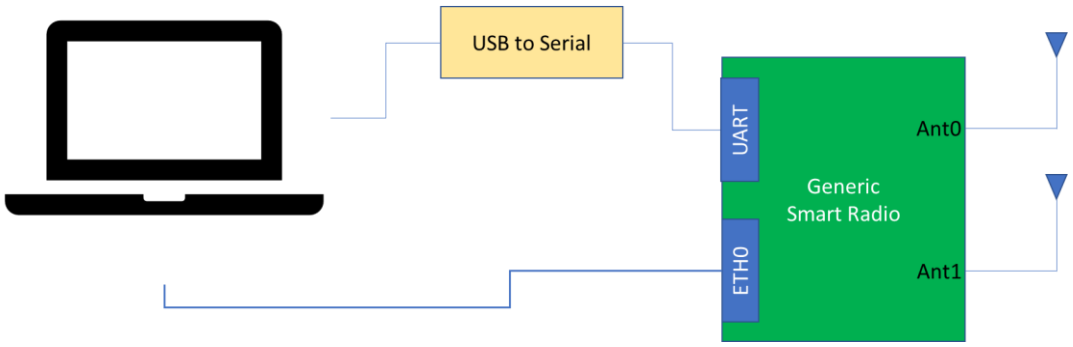

Figure 8 – Generic UART hardware connection

Configuring the UART port using the GUI

Open up a web browser and navigate to the IP address of the Smart Radio. From there, navigate to `Services → Socat Configuration`. A screenshot of the GUI is shown in Figure 9.

© 2020 Doodle Labs. All rights reserved.

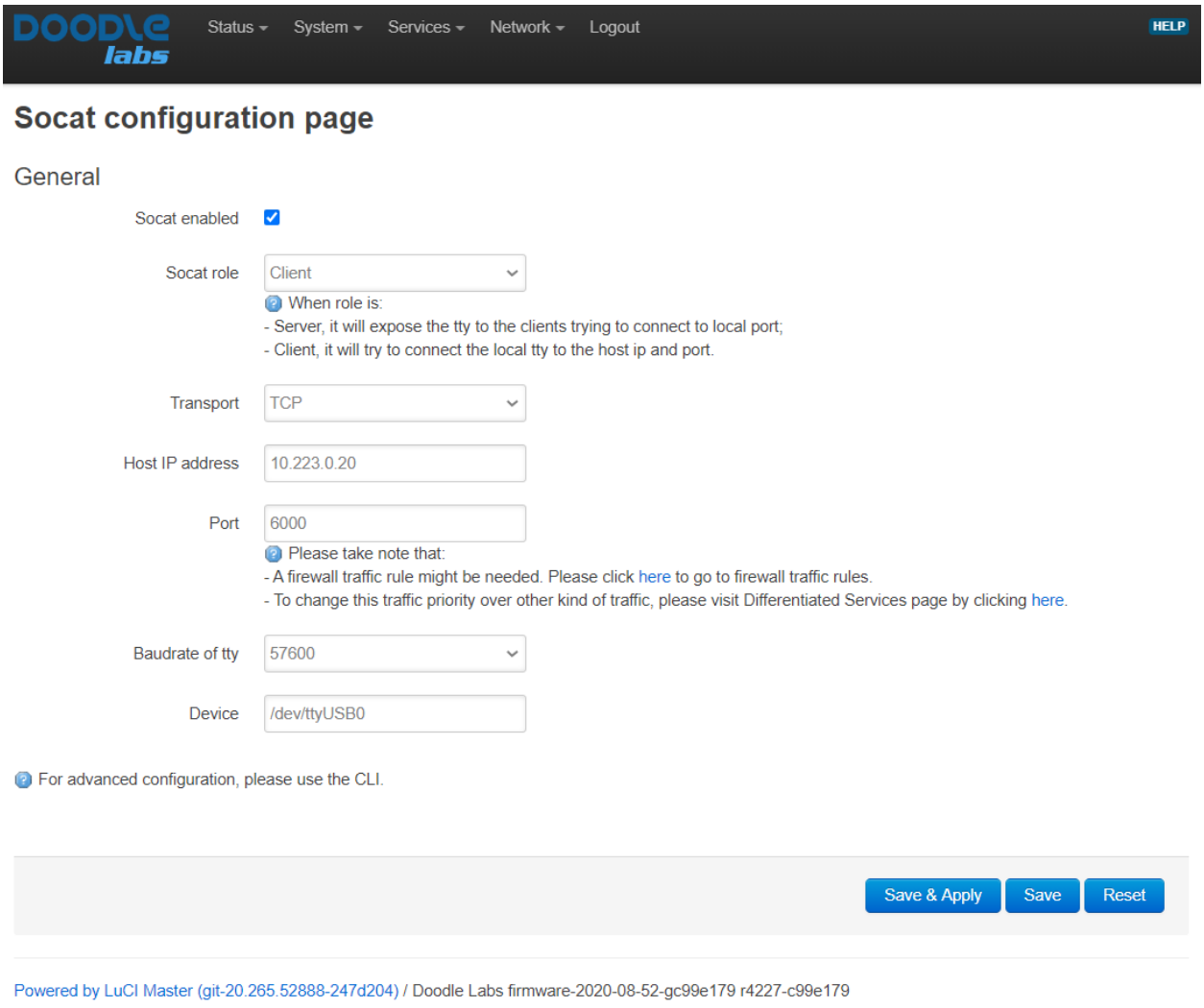

Figure 9 – UART Configuration GUI

When configuring the serial to network relay using the GUI, the program, `socat`, is used in the background. 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. The Device you choose depends on the hardware revision which you have (see the order code). -J hardware uses `/dev/ttyACM0`, and -H hardware uses `/dev/ttyUSB0`. After making your changes, click `Save & Apply`. The remainder of this section discusses configuration using the CLI.

Manually Configuring the UART port

We recommend using the GUI for serial port configuration, but if you want an advanced configuration, then it will be necessary to use the CLI.

© 2020 Doodle Labs. All rights reserved.

The current default serial port settings are 115200 baud rate, 8 data bits, no parity bit, and 1 stop bit. These settings can be changed using the `stty` program which is installed on the Smart Radio. When you are happy with the `stty` settings, you can modify the file `/etc/init.d/uart_defaults` to ensure that the settings are applied on boot-up. Note that if `ser2net` is used, then its serial settings will override those in the `uart_defaults` file once a connection is established.

First install a terminal emulator program like `minicom` in the host PC. Ensure that the TTY settings in the Smart Radio are the same as the settings in the host PC. You can use the program `stty` to modify configuration settings. Note that the UART port is accessed at `/dev/ttyUSB0` for -H hardware and `/dev/ttyACM0` for -J hardware.

Login to the device and update the UART settings to your liking using `stty`. The current default setting is configured as,

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

Testing the Configuration

At the host PC, start a minicom session with the following settings on the serial port: 115200 baud, 8N1, no hardware flow control, no software flow control. Make sure the serial device is the one connected to the Smart Radio. Back in the Smart Radio SSH session, send a command over UART with

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

"Hello World" should appear in the minicom session on the host PC. Next we can read input from the host PC. In the SSH session, type

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

Type "Hello World" in the host PC's minicom terminal. You should see it echoed in the SSH session of the Smart Radio. It is also possible to send an SSH remote command with the following example syntax

```
user@host-pc:~$ ssh root@<IP Address> 'cat < /dev/ttyUSB0'
```

Serial to Network Relay Configuration

Server Mode

In server mode, the Smart Radio opens up a network port and listens for an incoming connection. Therefore, the IP address of the client device is not required. ser2net is a good program for this task,and we can use ser2net to automatically forward network data to the UART port. This will replace the `echo` and `cat` commands above. As an example, the default configuration on the Smart Radio includes the following settings:

© 2020 Doodle Labs. All rights reserved.

```
2000:telnet:0:/dev/ttyUSB0:115200 NONE 1STOPBIT 8DATABITS
3000:raw:0:/dev/ttyUSB0:115200 NONE 1STOPBIT 8DATABITS
```

More information on ser2net can be found in its online documentation, and the example `ser2net.conf` file can be found in the Smart Radio at `/etc/ser2net.conf` after it is installed. ser2net is running in the Smart Radio by default with the above settings. Note that the serial port settings in the `ser2net.conf` file will override the settings in `/etc/init.d/uart_defaults` only if a connection is established with ser2net. To use UDP, modify the `ser2net.conf` file to add the line,

```
udp,3001:raw:0:/dev/ttyUSB0:115200 NONE 1STOPBIT 8DATABITS
```

After changing the configuration file, restart ser2net with

```
root@LEDE:~# /etc/init.d/ser2net restart
```

Next run SSH with port forwarding on the local host

```
user@host-pc:~$ netcat -C <IP Address> 3000
```

You can now send text between the minicom and netcat sessions.

Client Mode

In client mode, the Smart Radio sends packets to a host which should be listening on a particular network port. Therefore, the IP address of the server is required. ser2net is not capable of acting in client mode, but `socat` can be used instead.

`socat` is installed but not enabled by default since the IP address of the remote node is required. The settings can be changed in `/etc/config/socat`. For now, run `socat` with

```
root@LEDE:~# socat /dev/ttyUSB0,b115200,raw,echo=0 TCP:<IP Address>:3000
```

In this case, `<IP Address>` is the IP address of the remote smart radio. If you want to use UDP instead, use the following command

```
root@LEDE:~# socat /dev/ttyUSB0,b115200,raw,echo=0
UDP:10.223.141.25:3001,reuseaddr,sourceport=3001
```

Ruggedizing the commands

The instructions above establish basic means to receive and send UART messages. However, in your final application, you will find it necessary to extend the commands in order to ruggedize them (e.g. to allow multiple connections, or to recover from TCP disconnects). For example, you can modify the `ser2net` command to:

```
3000:raw:0:/dev/ttyUSB0:115200 NONE 1STOPBIT 8DATABITS max-connections=3
```

This allows for up to three incoming connections. You can use the following `socat` script to keep `socat` running after a TCP disconnect.

© 2020 Doodle Labs. All rights reserved.

```
root@LEDE:~# (while sleep 1; do
        socat /dev/ttyUSB0,b115200,raw,echo=0 TCP:<IP Address>:
3000,forever,reuseaddr >> /etc/logsocat.txt
)&
```

Testing the Serial to Network Relay

We can test the serial port either by sending packets directly over the network to the serial to network relay, or we can have a serial to network relay running on both radios and send data from UART port to the other.

PC to Smart Radio testing

At this point, we have setup out serial to network relay in either server or client mode. If the relay is in client mode, then you should start a server on your local PC for the client to connect to. We will use `netcat` for this.

```
user@remote-pc:~$ netcat -l -C 3000
```

Here the `-l` argument is added to tell netcat to listen on port 3000. If the relay is in server mode, the netcat can be used to connect to the relay with

```
user@remote-pc:~$ netcat -C <IP Address> 3000
```

In this case, `<IP Address>` is the IP address of Smart Radio running the relay.

Over the Air Messages

Assuming we have setup our Smart Radio as part of a mesh network of other Smart Radios, we can send messages wirelessly and have them appear on the UART port. Make hardware setup changes according to Figure 10.

© 2020 Doodle Labs. All rights reserved.

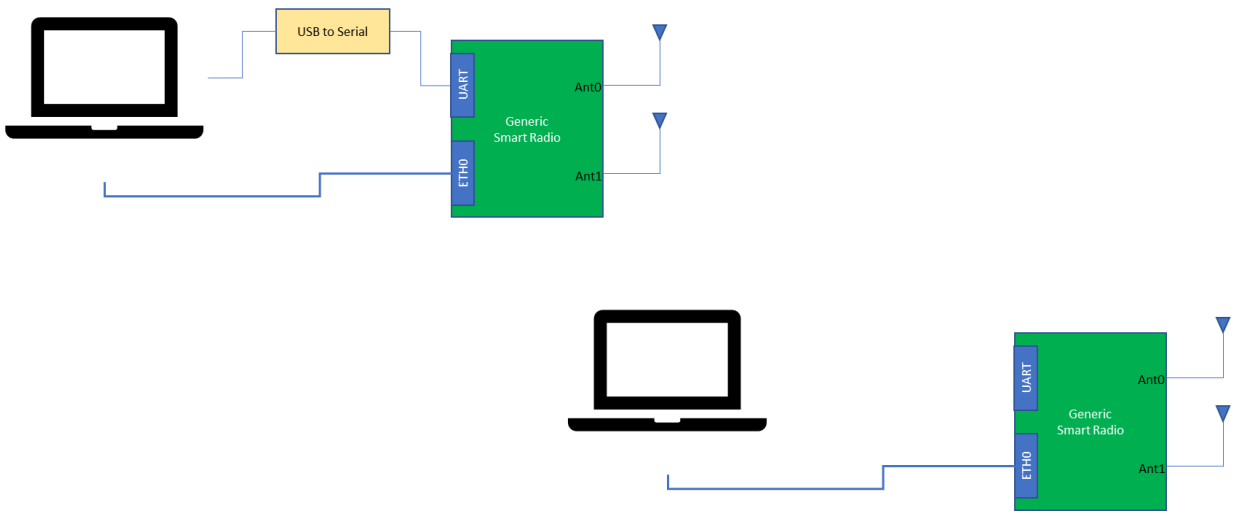

Figure 10 – Generic OTA Messaging Setup

As the entire mesh is setup as a transparent bridge, there is no difference in the settings or commands to send messages over the air.

UART to UART communications

As a last example, we will send messages from the UART port of Smart Radio 1 to the UART port of Smart Radio 2. We will continue with the previous setup and make modifications according to Figure 11.

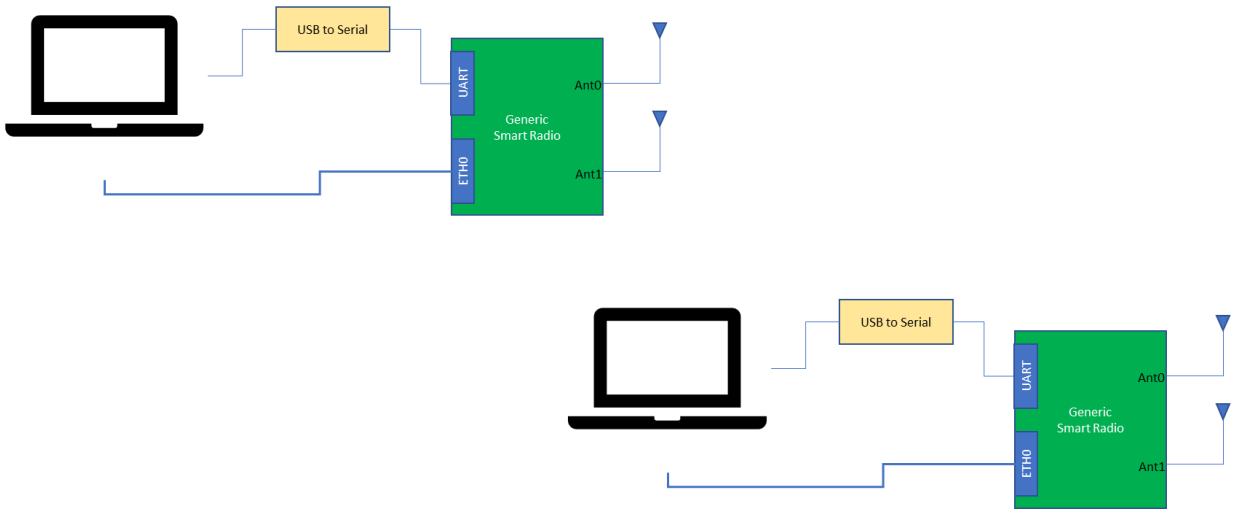

Figure 11 – Generic OTA UART to UART Setup

© 2020 Doodle Labs. All rights reserved.

Once again, the setup is the same as earlier except this time, we need two relays running. One of the relays should be a server and the other should be a client.

Linux Shell Access over Serial

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

You can embed this command in a script, and have it automatically restart by modifying inittab

```
bla:12345:respawn:/path/to/script
```

© 2020 Doodle Labs. All rights reserved.