RDS-TMC Spoofing using GNU Radio

Dimitrios Symeonidis
Security Technology Assessment Unit (G06), Via Fermi 2749, 21027 Ispra (VA), Italy
dimitrios.symeonidis@jrc.ec.europa.eu

***Abstract* – Encryption in the RDS-TMC standard is optional. Using a USRP mainboard, a BasicTX daughterboard and the GNU Radio toolkit running on a Linux platform, we demonstrate the possibility to spoof an RDS-TMC transmission and thereby convince COTS GPS-based navigators to change the calculated route.**

***Index terms* – RDS-TMC Spoofing using GNU Radio**

I. Disclaimer

This article presents a proof-of-concept implementation based on open-source hardware and software that demonstrates a security vulnerability in a commercial, widely deployed type of ITS (Intelligent Transportation System).

Any type of transmission in the FM broadcast spectrum is regulated in most – if not all – countries; transmission without a license is a criminal offence in most countries and it is strongly discouraged by the author.

It is by no means the intention of the author to assist or encourage others to engage in criminal, disruptive activities. Instead, the purpose and aspiration of this article is to raise awareness on the issue and motivate the people and the organisations with the power to do so to migrate to a more modern and secure protocol.

II. Introduction

The RDS (Radio Data System) protocol (IEC standard 62106) is transmitted on the 3nd harmonic of a stereophonic FM broadcast's pilot tone. Its most common application is transmission of radio station and current song names. However, an additional protocol transmitted inside an RDS stream is gaining popularity with the plummeting price of GPS navigation solutions; this is the TMC (Traffic Message Channel) protocol (ISO standard 14819).

TMC messages contain information about traffic jams, detours etc., which is used by suitable GPS receivers to re-calculate routes around these traffic jams with the ultimate goal of minimising the time-to-destination. Not all GPS receivers are capable of receiving and decoding TMC signals, but their number is rising.

TMC information is often transmitted by government radio stations, but increasingly also by private companies. Private companies then tend to encrypt the data stream and charge a fee for the ability to decrypt it, while governmental transmissions are usually unencrypted and free of charge.

In any case, no method exists today to authenticate an RDS-TMC transmission. We believe this to be a likely target for spoofing, be it in the context of a prank, as well as for criminal or even terrorist purposes.

In order to increase awareness on the topic, we developed a tool with the capacity to transmit arbitrary RDS-TMC information and demonstrated that a COTS GPS navigation solution with an integrated RDS-TMC receiver would actually change its calculated route when tuned to our transmission.

III. The RDS and TMC standards

A. History

The RDS protocol was initially released by the EBU (European Broadcasting Union) in 1984, then as CENELEC (European Committee for Electrotechnical Standardization) standard EN 50067, initially released in 1990, then 1992, then 1998. In 1999 it became international standard IEC 62106 (International Electrotechnical Commission): „*Specification of the Radio Data System (RDS) for VHF/FM sound broadcasting in the frequency range from 87,5 To 108,0 MHz*“.

The RDS logo is copyright of the RDS Forum

The RDS protocol was inspired by the German ARI (Autofahrer-Rundfunk-Informationssystem), which operated in West Germany since 1974 and was superseded by RDS. ARI carried traffic announcements (voice) on the 57 kHz subcarrier, just like RDS. In fact, the ARI and RDS transmissions co-existed peacefully until 2005, when the German ARD network seized ARI transmission on March 1st.

Contrary to RDS, which is a mature and well-established protocol, the TMC protocol is rather new, with the first adoptions in 1997-98 (Germany, Italy) and most western countries starting TMC transmissions in the mid-to-late '00s.

RDS-TMC was originally described in 1997-98 in CEN (Comité Européen de Normalisation, or European Committee for Standardisation) Pre-Standard ENV12313. It is superseded since 2003 by ISO standard 14819 „*Traffic and Traveller Information (TTI) – TTI messages via traffic message coding*“, composed of 3 parts:

1. ISO 14819-1: *Coding protocol for Radio Data System – Traffic Message Channel (RDS-TMC) using ALERT-C*
2. ISO 14819-2: *Event and information codes for Radio Data System – Traffic Message Channel (RDS-TMC)*
3. ISO 14819-3: *Location referencing for ALERT-C*

The event codes in ISO 14819-2 (and the preceding ENV12313-2) are compatible with those in ENV 13106:2000 „Road transport and traffic telematics - DATEX traffic and travel data dictionary“.

B. Physical Layer

FM Radio is broadcast (in most countries) between 87.5 and 108MHz with 200kHz of channel spacing, while maximum frequency deviation of the FM modulator is ±75kHz.

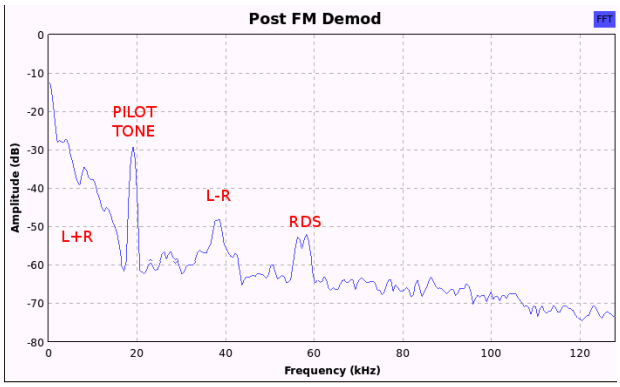


Figure 1: Spectrum of an FM broadcast, after FM demodulation

The lower 15kHz of the demodulated FM signal carries the monophonic audio signal (Left+Right), which was designed like this for reasons of backwards compatibility with mono FM receivers. Then at 19kHz there's the "pilot tone", transmitted at approx. 10% the overall modulation level. Between 23 and 53kHz, modulated onto the pilot tone's 2nd harmonic (38kHz) and phase-locked to the pilot tone there's the stereophonic audio signal (Left-Right). This type of transmission is called double-sideband suppressed carrier (DSB-SC). Finally, between 55 and 59kHz there's the RDS signal modulated onto the pilot tone's 3rd harmonic (57kHz), with the carrier yet again phase-locked to the pilot tone.

The RDS signal is a BPSK-modulated, differentially-encoded 1187.5 bits-per-second data stream. The 1187.5bps rate is one-sixteenth of the pilot tone's 19kHz.

C. Data-Link Layer

One RDS "group" (frame) is 104 bits long, comprised of 4 "blocks" of 26 bits each. A block includes the "infoword" (16 bits) and the "checkword" (10 bits). There are 32 group types (0-15, A or B) and the decoding of the infowords depends on the group type. This structure is illustrated in Figure 2:

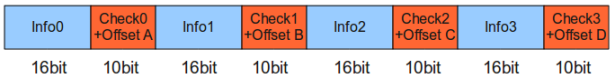


Figure 2: Structure of an RDS group

The benefits of the checkword are:

1. group and block synchronisation (framing)
2. basic error protection; the code can detect all single and double bit errors in a block, as well as any single error burst spanning 10 bits or less and 99.9% of longer error bursts.
3. forward error correction; it can correct any single error burst spanning 5 bits or less.

On transmission, the checkword is calculated using a modified shortened cyclic code as follows:

a) the infoword is shifted left 10 bits and then divided modulo(2) by 0x5B9 (equivalent to the generator polynomial: $x^{10}+x^8+x^7+x^5+x^4+x^3+1$)

b) the result of the above operation is XOR'ed with a 10-bit long „offset word“, which depends on the block number and whether the group type is A or B.

The checkword is then appended to the corresponding infoword and the block is transmitted m.s.b.-first (most significant bit).

On reception (syndrome decoding), xxxxx

D. The RDS-TMC protocol

The TMC protocol digitally and silently carries information about roadworks, weather and traffic incidents. It is transmitted inside an RDS signal as group type 8A. RDS-TMC messages can span a single or multiple groups (up to five), with multi-group messages carrying not only incident information but also diversion advice. Each RDS-TMC group is sent at least twice in succession (“Immediate Repetition”).

A single-group RDS-TMC message will carry information on the event's duration & persistence, the event's direction and number of segments affected, the event code (11bits) and the location code (16bits). Event codes are then looked up in the event table (universal, defined in ISO 14819-2), while location codes are looked up in the corresponding location table (country-specific, defined in ISO 14819-3).

The above message format is called the "ALERT-C" protocol and it's the most common, but not the only format. In fact, ISO 14819-4 defines the more advanced "ALERT-PLUS" protocol for additional information such as travel times, and transmitters have the option to send (for example) ALERT-C messages unencrypted (free-of-charge) and ALERT-PLUS messages encrypted (for a charge).

In addition to the 8A groups, which carry the RDS-TMC messages, the protocol uses also 3A groups to announce the availability of the service and to send "System Information" messages, such as LTN (Location Table Number) and SID (Service Identifier).

IV. GNU Radio & USRP

We used a laptop computer running Ubuntu Linux 9.10, onto which GNU Radio version 3.2.2 was compiled and installed. Our code re-used many existing GNU Radio blocks, while some others were written by us in C++. The blocks were connected into a flow-graph using Python. A BasicTX daughterboard was installed on a USRP, with an antenna connected to the Tx/Rx port without any filter or amplifier.

Before describing the shape and details of our transmitter, we will briefly introduce the GNU Radio software toolkit and the USRP RF frontend.

A. The GNU Radio software toolkit

The GNU Radio toolkit is a FOSS (Free Open Source Software) project for developing Software Defined Radios. Started as a fork of the MIT SpectrumWave Psectra code in 2001 and funded by philanthropist John Gilmore, it is an official GNU project.

GNU Radio includes a wide selection of DSP blocks written in C++, such as:

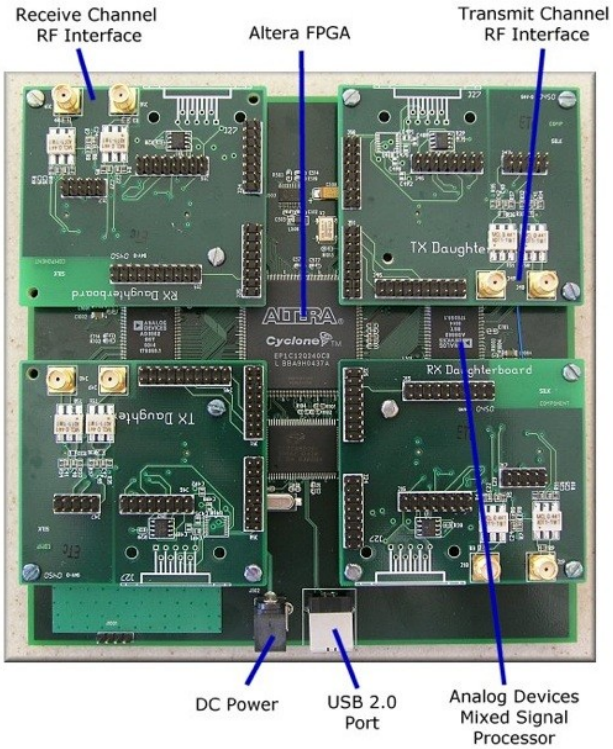
a) filters, mixers, modulators, synchronisers, etc
b) several graphical sinks, such as scope, FFT, constellation, waterfall, etc
c) sources and sinks for various types of hardware, such as the USRP, USRP2, the soundcard, wav files, etc
d) all the above have Python equivalents, automatically generated through SWIG glue

A GNU Radio application consists of one or more flowgraphs, with each flowgraph having one or more sources, one or more sinks, and processing blocks in between. Flowgraphs are written in Python, either by hand or using the graphical GNU Radio Companion (GRC).

Most GNU Radio DSP blocks don't have an actual throughput rate. Instead, the flowgraph's rate is derived from any USRP or Audio sources and sinks included (the only blocks that have a „real" sampling rate) and enforced by the GNU Radio scheduler.

Finally, users are not constrained to the selection of standard DSP blocks included in GNU Radio, but can write their own signal processing blocks and use them in a flowgraph alongside the standard blocks. In fact, a selection of such user-contributed blocks is available at the „Comprehensive GNU Radio Archive Network" (CGRAN) website.

B. The USRP RF frontend



The Universal Software Radio Peripheral

The USRP (Universal Software Radio Peripheral) is a computer peripheral designed within the auspices of the GNU Radio project and produced by Ettus Research LLC. It is "Open Source Hardware", in the sense that all schematics, BOM, etc. are freely available, so anyone can use them to produce (and even sell) an exact copy of the USRP.

The USRP connects to the host computer over a USB 2.0 wire; a Cypress FX2 chip translates

between the USB bus and the FPGA, an Altera Cyclone EP1C12; the GNU Radio project includes some standard Verilog bitstreams for this FPGA, but it's also possible to modify them using the free Altera tool "Quartus II Web Edition". The FPGA acts as a DDC (Digital DownConverter) and/or DUC (Digital UpConverter).

The FPGA is connected to two AD9862, each containing two 12-bit ADCs and two 14-bit DACs at 64 Msamples/sec. These in turn are connected to daughterboards, which translate various RF ranges to complex baseband signals (and vice-versa), from DC up to 6 GHz. The available bandwidth at the host is limited by the USB2.0 bus; while theoretically it can reach 480Mbps, practically it can sustain approx. 32MB/s, which corresponds to 8 MHz of complex (I/Q) 16-bit samples. If real-sampling is used, the USRP provides access to 16MHz of spectrum.

In this experiment we used a BasicRX board for reception and a BasicTX board for transmission.

V. Experimental Setup

A. The RDS-TMC receiver

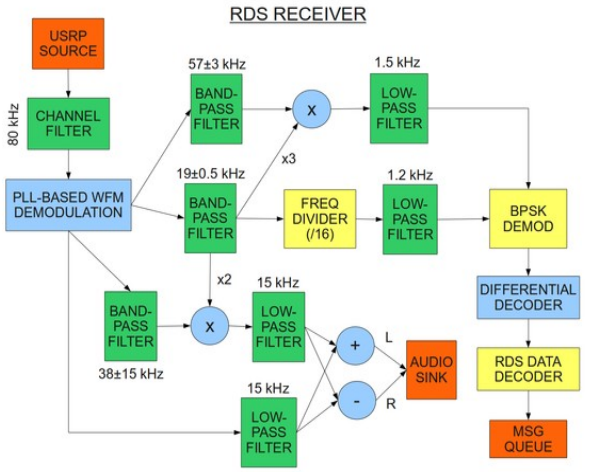


Figure 3: Structure of the RDS receiver

Before starting work on the transmitter, we dedicated some time in studying and understanding the RDS-TMC standard(s): extensive commenting of the (existing) GNU Radio-based RDS receiver code was done, as well as cross-referencing to the relevant sections in the standards, much like in a reference implementation (RI). Furthermore, the missing RDS-TMC receiver capabilities were added and correct operation was confirmed by receiving a real RDS-TMC signal. The structure of the RDS receiver is illustrated in Figure 3.

The RDS-TMC receiver came in handy also during the conducted testing phase, described further down.

B. The RDS-TMC Transmitter

The shape of transmitter flowgraph can be seen in Figure 4 (the blocks in yellow colour were written specifically for this proof-of-concept):

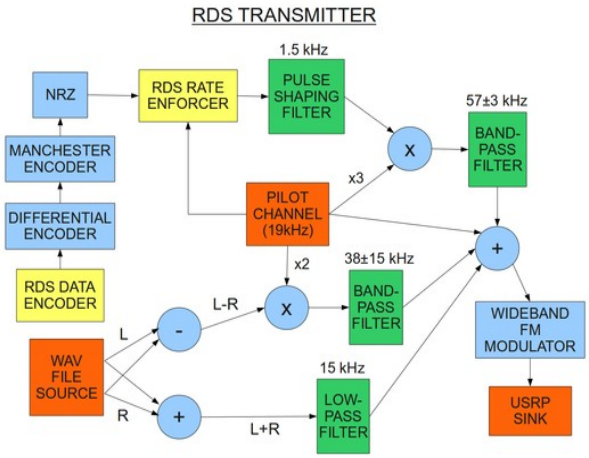


Figure 4: Structure of the RDS transmitter

The relevant RDS data is read from an XML file and encoded into infowords; the checkwords are then calculated and joined with the infowords into blocks and groups, which are then streamed out of the RDS Encoder block. This bitstream is then differentially-encoded (to overcome the 0-180 phase ambiguity of the BPSK modulation), Manchester-encoded (for self-clocking) and NRZ-encoded (non-return-to-zero, i.e. in the range of [-1, 1]).

Please note that, while no block exists in GNU Radio to do Manchester encoding, it's quite trivial to create one: map([1,2]) → unpack_k_bits(2).

The RDS data rate of 1187.5bps is enforced by a separate block just before filter-shaping and BPSK-modulation. The data rate of 1187.5bps is created by dividing the 19 kHz frequency of the pilot tone by 16 (by means of counting zero-crossings). The same pilot tone is mixed with itself twice to create the 3rd harmonic as the carrier of the RDS data stream.

The audio source for the transmission is a stereo .wav file. From the left and right channels we create the L+R (mono) and L-R (stereo) signals. The stereo signal is then upconverted to 38 kHz; the 38 kHz carrier is created by mixing the pilot tone with itself.

The L+R, L-R and RDS signals are filtered and, along with the pilot tone, mixed to create the full baseband signal. It is then FM modulated with a

maximum frequency deviation of ±75 kHz, amplified and sent to the USRP for transmission.

C. Transmission power

As mentioned earlier, it is a criminal offense to transmit in the 87.5-108MHz spectrum without a license. To overcome this obstacle, most of the testing during the development of this system was done using conducted testing: on a single USRP containing 1 BasicTx and 1 BasicRX daughter-board, the output of RDS-TMC transmitter was connected directly to the input RDS-TMC receiver. A snapshot of the conducted testing can be seen in Figure 5:

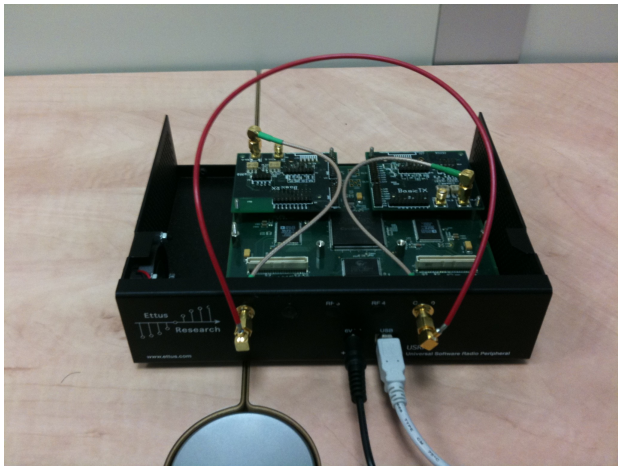


Figure 5: Conducted Testing

Once we achieved satisfactory results with conducted testing, limited radiation testing was done with a COTS satellite navigator. We made sure throughout these tests that the transmission power remained well within the 50nW e.r.p. (effective radiated power) limit defined in Annex 13 of ERC Recommendation 70-03: "*Relating to the use of Short Range Devices (SRD)*".

Figure 6: Radiation testing

D. The frequency choice

While for the purpose of simplicity we specifically tuned our TMC-enabled satellite navigator to the frequency of our transmission, there are other ways to "convince" such a device to decode such a transmission:

a) transmitting at one of the Alternative Frequencies of the transmission, to which the device is currently tuned (see RDS group 0A)
b) many modern TMC-enabled satellite navigators regularly scan the entire FM spectrum looking for TMC transmissions
c) if none of the above methods work, one could always jam (locally) the valid TMC transmission, forcing the device (or the driver) to manually retune to our transmission

VI. Results

VII. Conclusion & Future Work

We successfully demonstrated that the lack of compulsory encryption makes is easy for someone to intentionally spoof an RDS-TMC signal using COTS hardware and open-source software.

We strongly advise against the use of un-encrypted RDS-TMC signals, both for the general public and for structured Intelligent Transportation Systems (ITS).

We intend to investigate similar vulnerabilities in the DAB DLS (Digital Audio Broadcasting - Dynamic Label Segment) protocol (ETSI EN 300 401), as well as in any other protocols that may come out of the TPEG (Transport Protocol Experts Group).

VIII. Acknowledgements

A similar demonstration was made in April 2007 at the CanSecWest security conference in Vancouver, Canada by Andrea Barisani and Daniele Bianco of Inverse Path Ltd. Contrary to the work presented in this paper, they used custom-

built hardware in their experimental setup. Their presentation was titled “Unusual Car Navigation Tricks: Injecting RDS-TMC Traffic Information Signals” and released under a Creative Commons by-nc-nd license. The work of Barisani and Bianco has been the inspiration for this one.

The receiver part of the source code is based on previous work by Ronnie Gaensli, which was later adopted by Ryan Shoff, and finally by Matteo Campanella. The work presented here expands upon those.

IX. References

[1] *The new RDS IEC 62106:1999 standard*, The RDS Forum, December 1999

[2] *Traffic and Traveller Information (TTI) – TTI messages via traffic message coding*, ISO Standard 14819, June 2003

[3] *Unusual Car Navigation Tricks: Injecting RDS-TMC Traffic Information Signals*, Andrea Barisani and Daniele Bianco of Inverse Path Ltd, CanSecWest 2007

[4] The GNU Radio website, http://gnuradio.org/

[5] The Comprehensive GNU Radio Archive Network website, https://www.cgran.org/

[6] The Ettus Research LLC website, http://www.ettus.com/

[7] *The USRP under 1.5X Magnifying Lens*, Firas Abbas Hamza, June 2008

[8] *Relating to the use of Short Range Devices (SRD)*, ERC Recommendation 70-03, October 2009