Hacking the DSMx Drone RC protocol

Cyrille Morin[1], Leonardo S. Cardoso[1]
[1] Univ Lyon, Inria, INSA Lyon, CITI, France

Abstract

We present a decoder for a proprietary drone radio-control protocol, DSMx, implemented in GNURadio. This decoder is able to detect a transmission, decode transmitted data, find and follow the corresponding frequency jump sequence and identify the emitter by its manufacturing ID.

1 Introduction

Unmanned Aerial Vehicles (UAVs) or drones have been around fo decades for military and scientific purposes. But they have become increasingly popular with the general public in the last 10 years.

Their widespread usage causes airspace sharing issues. Existing regulations are tailored to big aircrafts, not drones, and their use is either not properly regulated or not covered at all and the public is often not aware of the legal issues with piloting what is sold to be a toy.

This creates privacy and safety issues: It's extremely simple to fly over private properties and film unwilling third parties, or even fly by restricted areas such as airports, military fields or nuclear plants. Manned aircrafts are also at risk of collision, and some drug cartels are starting to use some to deliver drugs to dealers.

Regulations enforcement is not simple due to the drones' size and stealthiness and, even if one is caught, it does not help with locating the pilot that can be more than a kilometre away.

Systems have been developed to physically capture drones with nets, to jam the radio transmission, or to simply destroy them. All these systems require knowledge of the drone's position and to be close to it.

A study and implementation of the radio control protocols in use could allow for detection of unwanted drones without line-of-sight and either takeover or targeted jamming without blocking the radio spectrum.

DSMx is a proprietary protocol from Spektrum used by other drone manufacturers such as Horizon Hobby. A couple of hobbyist groups such as PaparazziUav[1] and Deviation, whose objective is to create an universal transmitter implementing protocols from the various drone manufacturers, have it in their support list. The Deviation GitHub repository has a very good description of it[2]. But their approach uses the proprietary hardware with a source code modification to implement other protocols and add different transmission chips for increased compatibility.

Jonathan Andersson presented a system able to detect and decode a transmission and emit signals to hijack a drone in October 2016 at PacSec in Tokyo [3]. GNURadio was used to study the protocol but the implementation was done on hardware with the original radios. So, to the best of our knowledge, there is no previous DSMx decoder implemented with software defined radio

2 Protocol description

The DSMx protocol works on a CYRF6936 radio chip that implements a frame based data transmission with Frequency Hopping Spread Spectrum (FHSS) and Direct-Sequence Spread Spectrum (DSSS) based on a GFSK modulation scheme at 1Mbit/s in the 2.4GHz ISM band. After a preamble, a start-of-packet (SOP) code is sent in a specific sequence to tell the hardware the DSSS parameters of the communication. In normal operation, the DSSS step is done by sending variations over two 64bits long pseudo-noise (PN) codes: one bit corresponds to the PN code used, one tells if a NOT operation was applied and six more describe the number of bit-wise shifts of the code, so one 64bits chip codes for one byte.

Over this, the frame consists of the last two of the four identification bytes defined in each radio card, followed by 14 bytes coding for the values of seven RC channels. If needed, frames can be sent in pairs to handle up to 12 RC channels.

The PN codes used for data and SOP are selected from a 5 by 9 matrix depending on the current radio channel and the ID bytes.

A CRC is added at the end of each frame for error detection. It uses the first two ID bytes as seed. Unfortunately, the exact algorithm and/or polynomial was not found.

The FHSS element is done by changing the radio channel after each frame in a jump sequence of 23 different channels over 74 possibilities. We won't go into the details of the algorithm, but it is a constrained pseudo-random generator with a seed being the ID bytes.

There is also a pairing frame to configure the receiver to listen to one specific transmitter. It's sent

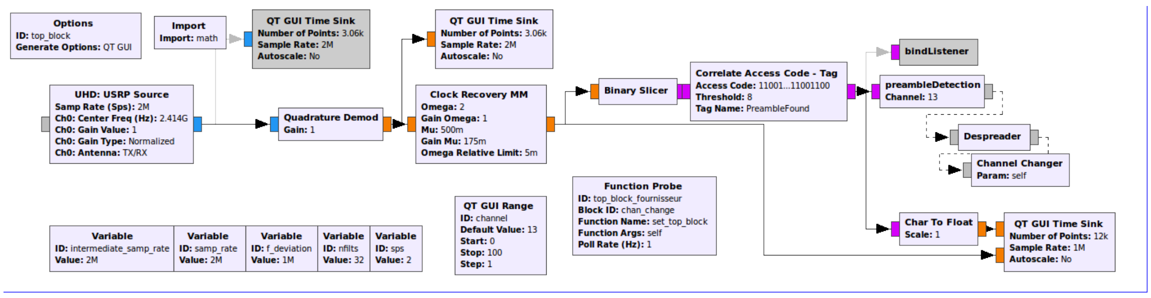


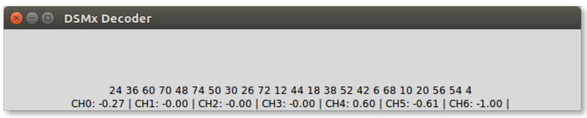
Figure 1: GNURadio flowgraph for DSMx detection, decoding, and FHSS channel sequence following

on a random radio channel and contains data describing the transmittter capabilities (RC channel number and protocol type) and, more importantly, the four ID bytes used in the jump sequence and the PN code selection.

3 Demo setup

This demonstration presents an implementation of a DSMx decoder in GNURadio.

The USRP listens to one channel at a time at 2Msps. In the three following blocks, the quadrature demod with the binary slicer handle the GFSK demodulation and the Clock recovery does the synchronisation. Then, we look for the preamble for time synchronisation and use next three custom blocks for: packet isolation and PN code discovery, data decoding, and frequency jump following and GUI output.



```
24 36 60 70 48 74 50 30 26 72 12 44 18 38 52 42 6 68 10 20 56 54 4
CH0: -0.27 | CH1: -0.00 | CH2: -0.00 | CH3: -0.00 | CH4: 0.60 | CH5: -0.61 | CH6: -1.00 |
```

Figure 2: Gui output of the system with current RC channel values and computed jump sequence

The codes used in each frame are based on the value of the ID bytes, that we don't know, but, with the selection algorithm and the radio channel number known, it becomes simple to search for the SOP code in only seven possibilities, and with that the PN codes can be extracted. So there is no need to know anything about the transmitter to be able to decode data.

If the algorithm was known, it would be possible to search over the space of 2 bytes to extract the two ID bytes that are not sent with each frame. Those two bytes would allow for exact transmitter identification and computation of the radio channel jump sequence.

To get these, the pairing frame can be listened to. This allows to demonstrate the capacity of the USRP to follow the jump sequence over a known transmitter.

4 Conclusion

In this demonstration, we present a GNURadio implementation of a system able to decode DSMx data from an unknown transmitter with a partial identification of the transmitting radio chip. If the transmitter is known, we can show that the SDR hardware used is able to follow the protocol's FHSS jump sequence quickly enough and not lose frames. Further progress on the system depends on the discovery of the exact CRC algorithm: It would allow full transmitter identification and jump sequence computation for full data decoding, selective jamming of the concerned channels or emission of custom frames.

The implementation code and flowgraph are available on GitHub at `https://github.com/lscardoso/gr-dsmx-rc`.

References

[1] Paparazzi. (2017) Paparazzi uav wiki. [Online]. Available: http://wiki.paparazziuav.org/wiki/Main_Page

[2] Deviation. (2017) Custom firmware for walkera devo radios. [Online]. Available: https://github.com/DeviationTX/deviation

[3] J. Andersson. (2016) Attacking dsmx spread spectrum frequency hopping drone remote control with sdr. [Online]. Available: https://pacsec.jp/psj16/PSJ2016_Andersson_Hacking_DSMx_with_SDR_PacSec_2016_English.pdf