Software Defined Radio for Analyzing Drone Communication Protocols

Dan Mototolea[1] , Cornelis Stolk[2]
[1] Doctoral School, Military Technical Academy, MTA, Bucharest, Romania
[2] Joint ISR Service Line, NATO Communications and Information Agency, NCIA, The Hague, Netherlands.
Contact author e-mail: mototoleadan@gmail.com

Abstract— **The aim of this research is to determine the possibilities of using commercial off the shelf FPGA based Software Defined Radio Systems to analyze and understand small drone communication protocols in order to hijack or smart jam the receiver. A study on decoding the DSM2 protocol of communication is presented. In order to understand the protocol, raw RF data were collected using Software Defined Radio and LabView software. A set of tools were written in MATLAB to analyze the data, extract the bursts, perform synchronization and demodulate the chips. Understanding the communication is vital when creating a counter drone system. The possibility of smart jamming the DSM2 protocol of communication is analyzed in the final part of the paper.**

Keywords— Drone communication protocols, Modulation schemes, Drone countermeasures, Hijack, Smart jamming, Software Defined Radio.

I. INTRODUCTION

Unmanned aerial vehicles (UAV) – alternately known as small drones, mini drones or micro UAVs – are remotely controlled from the ground, although higher-end models often additionally provide navigation technology so that they can independently fly predefined routes. UAVs are typically grouped into the following categories: drones for private use (toy and hobby), drones for commercial applications (aerial views, logistics, etc.) and drones for military applications (artificial targets, reconnaissance, combat). The next pages of this article consist on commercial use drones that are easy to buy and fly. Difficult to detect and capable of carrying payloads up to a few kilograms, drones represent an increasing threat to critical infrastructures and public figures and at public events. Security agencies, government authorities as well as private organizations and facilities that require protection must have the technological approach to counter this threat [1].

Incidents with commercially available drones appear almost daily in the media: drones in the vicinity of airports or even in the flight paths of aircraft (e.g. Heathrow, Munich, Warsaw, Taipei), drones above governmental buildings (the Japanese Prime Minister’s office, the White House South Lawn), drones at political events (German Chancellor Angela Merkel’s election rally in Dresden), drones above automotive test tracks, drones smuggling drugs over U.S.- Mexico border, etc.

Whether the operator is a negligent hobbyist or an agent intent on a malevolent act, an undetected drone can pose a significant safety or security threat [2].

A variety of approaches have been explored to interdict drones. These include shooting nets at the drones to tamper with their propeller to bring them down, using lasers to shoot down drones, spoofing GPS to confuse a drone’s localization system, hijacking the software of drones by hacking into them, using other drones to hunt down unauthorized drones, and even training eagles to attack and disable drones. However, these interdiction strategies typically presume that the presence of the drone has already been detected. Recent work has sought to develop drone detection systems that leverage either microphone, camera, or radar to sense the presence of drones. Each approach has its own limitations. Audio-based approaches can be confused by other sounds in noisy environments, has limited range, and cannot detect drones that employ noise cancelling techniques. Camera-based approaches require good lighting conditions, high quality lens, and camera with ultra-high resolution for detecting drones at long distance. Thermal and IR imaging cameras for long distance are prohibitively expensive and have limited coverage. Radio-frequency techniques based on active radar introduce RF interference. Geofencing is useful to prevent drones from flying into fixed areas known a priori as sensitive, but requires manufacturers to install such software and is less useful to prohibit drones from flying around temporary event venues [3].

In the next pages, a study on the communication between drone and controller will be presented. By understanding the communication between controller and drone, one could create a counter drone system that smart jams the receiver. Prior to understanding the communication between drone and its controller, a study on the frequencies and modulation schemes on which drones operate was performed.

Studies were performed on most popular drones, controllers and modulation schemes.

II. FREQUENCY BANDS AND MODULATION SCHEMES USED BY POPULAR DRONE BRANDS

The majority of controllers use the 2.4GHz spectrum (see Table I) with a proprietary Frequency Hopping Spread

Spectrum (FHSS) modulation. FHSS is used to maximize robustness and controlling distance. By exception Wi-Fi with Orthogonal Frequency Division Multiplexing (OFDM) modulation can be used in either the 2.4 GHz or the 5.8GHz band and is used to control a drone (Parrot AR2 on the 2.4GHz band). The advantage of using a flying hotspot (Parrot AR2) is that they can be controlled via a smartphone or tablet, making a separate controller unnecessary. A big disadvantage is the limited range and sensitivity for other Wi-Fi bands. Another exception is Yuneec which seems to use ZigBee 41 for control (Direct Sequence Spread Spectrum (DSSS) modulation). In some rare cases drones use the 5.8GHz band. An example is the DJI controller for the Phantom 2 Vision plus. Another rare case is the use of the 433MHz band. It is used by the Immersion EZUHF module [4].

Table I: Frequencies, modulations schemes and technologies used by popular drone brands

Brand	Frequency	Modulation	Technology
DJI Phantom	2.4/ 5.8 GHz	FHSS/ DSSS	FASST/ Lightbridge
Futaba	2.4 GHz	FHSS/ DSSS	FASST
Spektrum	2.4 GHz	FHSS/ DSSS	DSM2/ DSMX
JR	2.4 GHz	FHSS/ DSSS	DMSS
Hitec	2.4 GHz	FHSS/ DSSS	AFHSS
Graupner	2.4 GHz	FHSS/ DSSS	HOTT
Yuneec	2.4 GHz	DSSS	ZigBee
Parrot AR2	2.4 GHz	OFDM	Wi-Fi
Immersion	433 MHz	FHSS	EZUHF

The telemetry link is used to send information regarding the battery status, position and speed of the drone. This information is send overlaid the video link, or in different radio channel (868MHz, 433MHz or the 2.4GHz band). Another communication channel which can be used for telemetry is 4G. Note that the telemetry link is not present for every drone. Some manufactures use additional modules to transfer telemetry data. With that being said, we can conclude that if there is no camera and an additional telemetry module mounted on the drone, the drone only receives signals, being impossible to geo locate the drone by eavesdropping to the emitted signals from the drone. In that case, this technique for locating the drone only is not possible and only applies for the transmitter (the controller).

The video link is often done via the 5.8GHz band, but also the 2.4GHz is used. As can be seen in the overview of different video links in Table II, OFDM is a popular modulation technique for video. The reason is that OFDM enables high bitrates which are needed for video streams [4].

Table II: Frequencies, modulations schemes and technologies used for transmitting the video

Brand	Frequency	Modulation	Technology
DJI	2.4 GHz	OFDM	Lightbridge/Wi-Fi
Immersion	2.4 GHz	FM	-
Yuneec	5.8 GHz	OFDM	Wi-Fi
Connex	5.8 GHz	OFDM	-
Boscam	5.8 GHz	FM	-

4G is also used for the video link.

From Table I we can see that most commonly used digital modulations technologies are [4]:

- Frequency Hopping Spread Spectrum (FHSS);
- Direct Sequence Spread Spectrum (DSSS);
- FHSS+DSSS;
- Orthogonal Frequency Division Multiplexing (OFDM).

In the next chapter, the process of decoding the DSM2 protocol will be presented.

III. DECODING THE DSM2 PROTOCOL

DSM2 and DSMX are widely used radio protocols for 2.4GHz Spektrum RC transmitters and receivers. The protocol makes it possible to have multiple receiver/transmitters in the 2.4GHz Spektrum without much interference. There is only a small difference between DSM2 and DSMX; The hopping method between channels is different. In the DSM2 protocol the transmitter will choose two random channels, where the transmitter will look for the two best channels in the optimal case. In the DSMX protocol the transmitter and receiver both use the transmitter radio chip ID which is send during the binding process, for generating 23 channels. Each time the transmitter transmits a packet of data or the receiver receives a packet they will hop to the next channel [5].

In general, the protocol consists of three phases(procedures):

1. the binding phase;
2. the syncing phase;
3. and the normal transfer procedure.

The binding procedure. At the binding procedure both the transmitter and receiver will change to SDR (Single Data Rate) mode, a data code of 64 bits, a SOP (Start Of Packet) code of 64 bits and disable the CRC Checksum. The transmitter will then choose a random odd channel and starts sending the binding packet, at a very high rate. The receiver will start at the first odd channel and waits a bit longer than the rate the transmitters sends at. When it does receive a packet, it will check if it is a valid bind packet and goes to syncing mode. When it doesn't receive a valid packet, it will go to the next channel and will repeat this. During the binding procedure the transmitter also sends the amount of RC channels that it has. They will either transmit all the channels in one packet or in two packets after each other. [5]

The syncing procedure. Because the transmitter doesn't have a syncing procedure, the transmitter is already in the transfer procedure. The receiver on the other hand needs to synchronize itself with the transmitter. [5]

The transfer procedure will be presented in the next pages. The system is based on CYRF6936 chip.

In order to decode the protocol, we first collected raw RF data using Software Defined Radio (NI USRP-2952R [6]) and LabView software and decoded the signal into chips and then we corresponded the chips from the RF with the bits from the data collected via UART from the receiver (drone). A LabView application was developed to collect raw RF data at very high rate. The sample rate was set to 25M samples/second (1 sample is 0.04us). We collected files of 100ms each, that means 2.5milion samples per file. A set of tools were written in MATLAB to analyze the data, extract the bursts, perform synchronization and demodulate the chips. In addition, tools were written to record, decode and plot the serial data from the receiver. In Figure 1 we can see the hardware and software used for collecting the RF data and correspond it to serial data from the receiver.

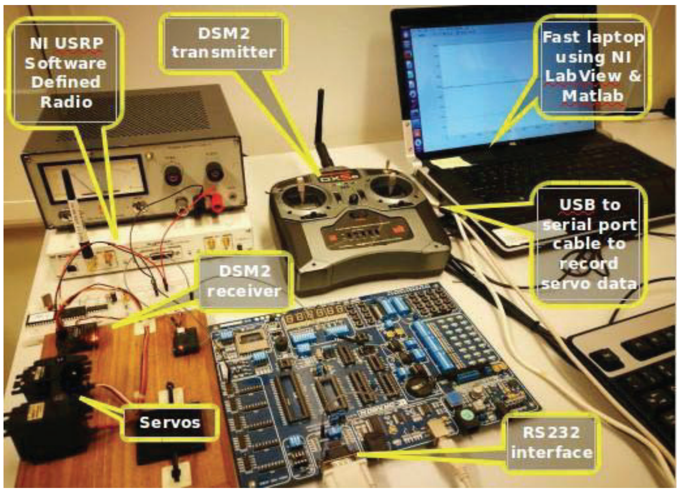
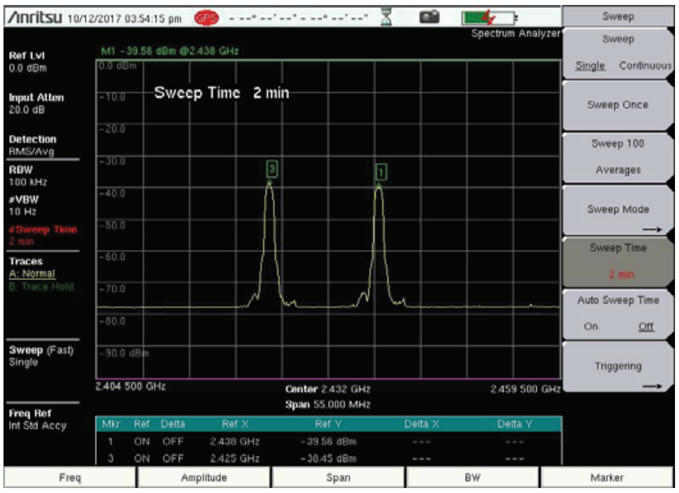

NI USRP Software Defined Radio
DSM2 transmitter
Fast laptop using NI LabView & Matlab
USB to serial port cable to record servo data
DSM2 receiver
Servos
RS232 interface

Figure 1: Setup used for decoding the DSM2 protocol

As it was mentioned above, in the DSM2 protocol the transmitter will choose two random channels.



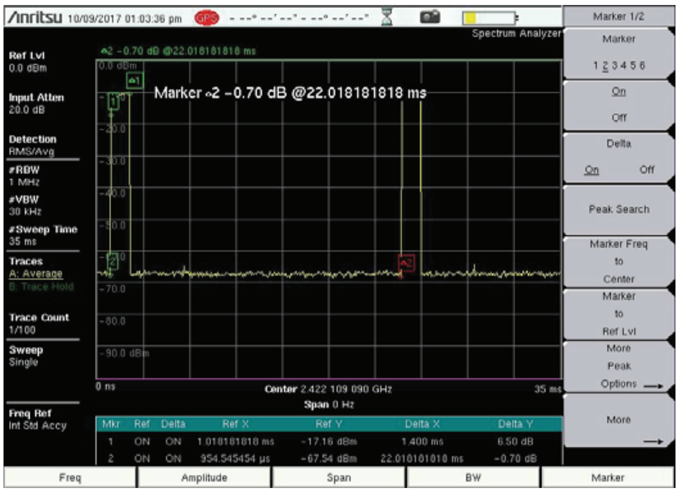
Ref Lvl 0.0 dBm
Input Atten 20.0 dB
Detection RMS/Avg
RBW 100 kHz
#VBW 10 Hz
#Sweep Time 2 min
Traces A: Normal B: Trace Hold
Sweep (Fast) Single
Freq Ref Int Std Accy

2.404 500 GHz Center 2.432 GHz 2.459 500 GHz
Span 55.000 MHz

Mkr	Ref	Delta	Ref X	Ref Y	Delta X	Delta Y
1	ON	OFF	2.438 GHz	-39.56 dBm	---	---
3	ON	OFF	2.425 GHz	-30.45 dBm	---	---

Figure 2: The two channels in use

Channels are centered on each whole MHz in the 2.4 - 2.48GHz band; this means a total of approximately 80 different channels are available. Figure 2 shows the two channels on the spectrum analyzer. Every time the transmitter is turned off and on, other two channels will be used. From Figure 3 one can conclude that one burst lasts approximately 1.4ms and it is transmitted every 22ms



Ref Lvl 0.0 dBm
Input Atten 20.0 dB
Detection RMS/Avg
#RBW 1 MHz
#VBW 30 kHz
#Sweep Time 35 ms
Traces A: Average B: Trace Hold
Trace Count 1/100
Sweep Single
Freq Ref Int Std Accy

0 ns Center 2.422 109 090 GHz 35 ms
Span 0 Hz

Mkr	Ref	Delta	Ref X	Ref Y	Delta X	Delta Y
1	ON	ON	1.018181818 ms	-17.16 dBm	1.400 ms	6.50 dB
2	ON	ON	954.545454 µs	-67.54 dBm	22.018181818 ms	-0.70 dB

Figure 3: Bursts on one channel

The information on the second channel is identical to the one transmitted in the first channel. Every burst in the second channel is shifted by 4ms after the burst from channel one (the spectrum analyzer is configured in zero span mode; meaning the horizontal axis is time, not frequency).

After seeing the two frequency channels on the spectrum analyzer, we tuned the LabView application on the carrier frequency of one channel and started to collect files. As it was mentioned before, each file has 2.5 million samples, equivalent of 100ms, so 4 or 5 bursts on each channel. After creating the MATLAB scripts that helped us understand the modulation scheme, we concluded that the manufacturer chose the Gaussian Frequency Shift Keying (GFSK). All elements of the transmission are transmitted using GFSK where the carrier frequency is shifted by +- 300kHz.

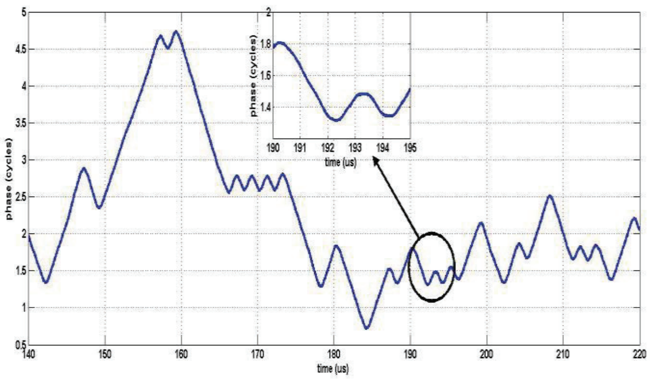


Figure 4: Phase- time diagram for DSM2 transmission

By doing so, it indicates a 1 or a 0, in this way 1 bit (chip) is encoded in 1 symbol.

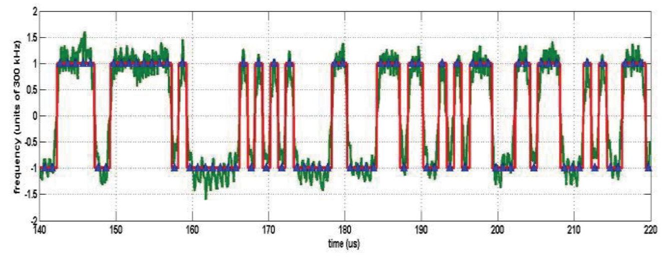


Figure 5: The carrier frequency in units of 300kHz (green), its sign (red) and the decoded data (blue triangles)

The rate at which the frequency is modulated is 1MHz, thus every µs a change in frequency may take place. Figure 4 shows the carrier phase over time. The frequency is simply the time derivative of the phase.

Figure 5 shows the frequency, normalized to units of 300kHz in green. We took the sign to convert these to 1's and -1's, as shown in red. The transitions indicate the symbol boundaries. "Sampling" in between the boundaries will provide the decoded symbols, as shown in blue triangles.

The packet structure for the DSM2 protocol is shown below:

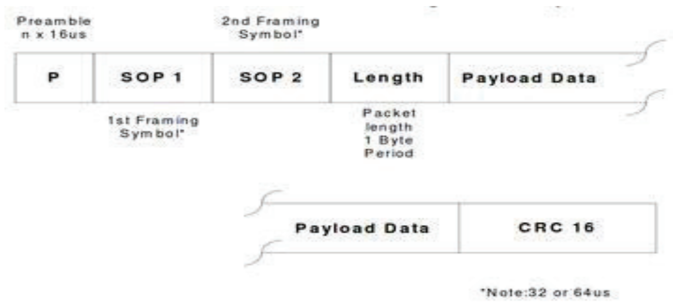


Figure 6: Packet structure

The packet structure consists of a preamble, start of packet data (SOP), length, the payload data and a cyclic redundancy check (CRC 16). We can say that all the above can be found in a burst (Figure 3).

The preamble is a 16-chip sequence transmitted at the start of a packet. Its primary purpose is to allow the receive correlator to establish the appropriate bit centring and synch up on the data stream before the other structures in the packet are transmitted. In general, a 2-3 symbol repetition with a pattern such as 0x3333 or 0x5555 should provide suitable performance [7].

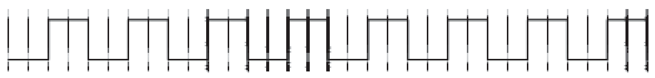


Figure 7: Recommended Preamble Sequence

Start of packet (SOP) symbols are used to bound the beginning of the packet data, and also provide the added feature of encoding the data rate for the remainder of the packet. The SOP code is (almost certainly) communicated during the pairing process where transmitter and receiver are linked to each other. SOP is composed primarily of two symbols. Each symbol is a 64-chip PN code. The second SOP is exactly just like the first, but inverted [7].

An optional *CRC16* is performed on data to check if there were errors during the transmission of the packet. The CRC16 is performed on only the length and data fields of the packet.

The CRC16 can be represented as a polynomial (1) as shown below:

$$G(X) = X^{16} + X^{15} + X^{2} + 1 \qquad (1)$$

From our analysis (see Table III) we found that the chips that correspond to payload data are located between chip 197 to chip 1220 (16 bytes). First, the serial data from the receiver were analyzed and then a correspondence between the RF chips and serial bytes was performed.

The receiver synchronizes and demodulates the RF signal, performs various checks (such as SOP matching, CRC error correction) and performs the mapping from 64-bit sequences to bytes.

Table III: The packet structure of DSM2

Field name	# of chips	From chip #	up to chip #
Preamble	x	-x	-1
SOP	64	0	63
4 chip pause	4	64	67
SOP inverted	64	68	131
1 chip pause	1	132	132
Length	64	133	196
Byte 0	64	197	260
Byte 1	64	261	324
Byte 2	64	325	388
Byte 3	64	389	452
Byte 4	64	453	516
Byte 5	64	517	580
Byte 6	64	581	644
Byte 7	64	645	708
Byte 8	64	709	772
Byte 9	64	773	836
Byte 10	64	837	900
Byte 11	64	901	964
Byte 12	64	965	1028
Byte 13	64	1029	1092
Byte 14	64	1093	1156
Byte 15	64	1157	1220
CRC byte 1	64	1221	1284
CRC byte 2	64	1285	1348
	----------- +		
Total number of chips	1349		
Burst length	1349 µs + preamble		

The 1's and -1's received over the air are referred to as "chips" rather than bits because they do not contain the information about servo positions directly. Instead they encode the data, where each byte is mapped to a sequence of 64 chips.

The bytes are output in serial format at low-voltage level. Using a simple RS232 transceiver chip, such as the common MAX232, this can be connected to a standard PC serial port. The applicable baud rate is 115200 baud. Table IV through Table VI explain how this serial data is formatted.

Table IV: The 16 bytes packet structure output by the receiver as serial data

0	Header
1	
2	Servo A
3	
4	Servo B
5	
6	Servo C
7	
8	Servo D
9	
10	Servo E
11	
12	Servo F
13	
14	Servo G
15	

Table V: The 2 bytes for each servo channel

servo	
Bits 15:10	Bits 9:0
Channel ID	Servo position

Table VI: The convention of function and channel ID

Channel ID	Function
0	Throttle
1	Aileron
2	Elevator
3	Rudder
4	Gear
5	AUX 1

Figure 8 shows the servo values for a period of approximately 75 seconds. Servo channel 2, the elevator was moved up and down during this period.

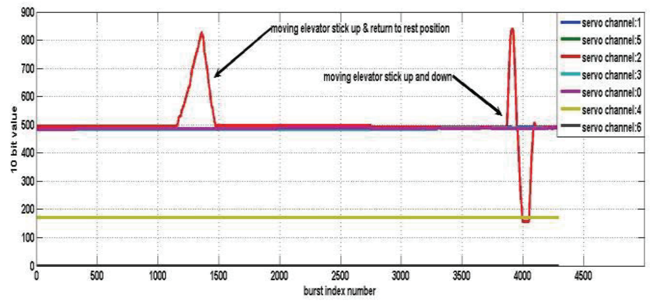


Figure 8: The servo positions for 4295 bursts (~75s)

In Table VII we can see the correspondence between RF chips and servo data bytes collected from the receiver.

Table VII: Correspondence between RF chips and servo data bytes

Field name	From chip #	up to chip #	Corresponds to
Byte 2	325	388	Aileron MSB
Byte 3	389	452	Aileron LSB
Byte 4	453	516	AUX1 MSB
Byte 5	517	580	AUX1 LSB
Byte 6	581	644	Elevator MSB
Byte 7	645	708	Elevator LSB
Byte 8	709	772	Rudder MSB
Byte 9	773	836	Rudder LSB
Byte 10	837	900	Throttle MSB
Byte 11	901	964	Throttle LSB
Byte 12	965	1028	Gear MSB
Byte 13	1029	1092	Gear LSB
Byte 14	1093	1156	AUX2 MSB
Byte 15	1157	1220	AUX2 LSB

The mapping from each byte to 64 chips is done in such a way to maximize the “distance” between two bytes. This means that each of the 256 sequences of 64 chips will differ in (almost) 32 places. In mathematical terms the 256 sequences form a near-orthogonal set. This is very similar to the use of so-called “Walsh codes” or “Hadamard Sequences”. There are many possible mappings and the one used is determined by the contents of a register in the CYRF chip (communicated during the pairing process).

The setup described in Figure 1 allows simultaneous recording of raw RF data from the transmitter and the corresponding servo data output by the receiver. This allows to “measure” the mapping to some extent. Moving one of the joysticks slowly up and down allows to build up the map. We found out that the map is not the same for both frequency channels and probably also different for each individual transmitter.

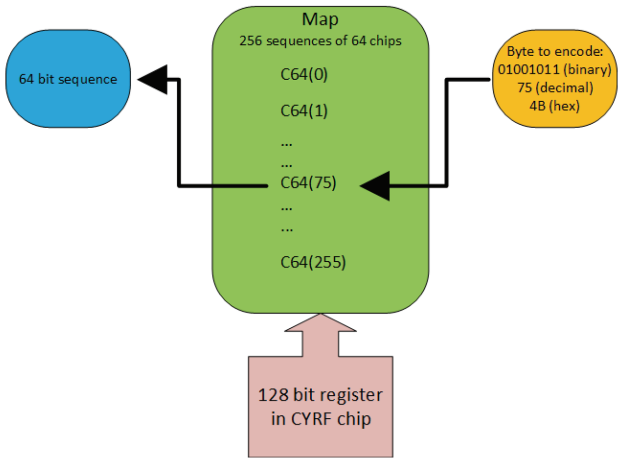


Figure 9: Mapping from 8 to 64 bits in the transmitter. At the receiver the reverse operation is performed

Effectively this mapping creates a chip rate which is a factor 8 times larger than the original bit rate. Given the same modulation method, this higher chip rate will spread the spectrum. At the receiver, due to the orthogonality properties of the 64 chip sequences, many chips will have to be flipped due to noise or interference, before a wrong byte is assumed. This makes a link which is fairly robust for interference.

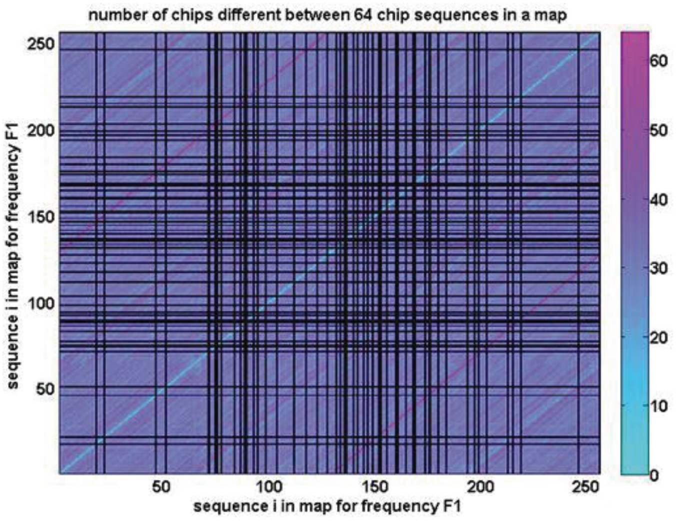


Figure 10: Auto-correlation property of mapping

Figure 11 shows the distance between the map for F1 and F2. It is obvious that a completely different mapping is used. The mapping is company proprietary.

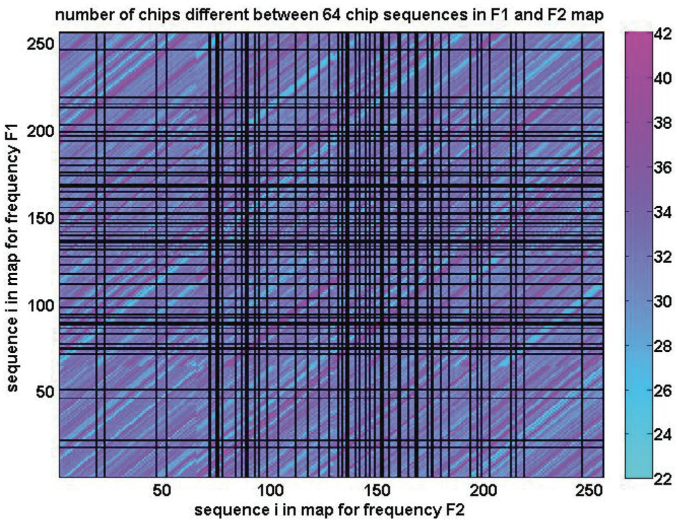


Figure 11: Cross-correlation property of mapping

IV. Conclusions

Taking over control of a UAS using this type of commercial radio link is difficult. The pairing process uniquely links transmitter and receiver. Proprietary codes are used to protect the drone from being taken over, which are shared during the pairing process. Fingerprinting the transmitter is relatively easy. Even though RF data may not be decoded, there are certainly enough clues to derive the model of radio control and some characteristics unique to this particular device.

Jamming this type of UAS using barrage jamming, without a reactive capability will be difficult. The entire 2.4 GHz band (80 MHz wide) would have to be jammed.

It is likely that the only reliable jamming method involves a reactive feature and attacks the link in a smart way. Surely this must include listening for activity, identifying the model of transmitter and the channel in use. This is especially important for later versions which hop all over the band rather than just two frequencies. Smart attacks could focus for example on destroying the SOP after a preamble is detected or perhaps jam the CRC part of the data.

When creating a counter drone equipment, one would have to keep in mind that there is no ideal response plan. In the present moment there is no such system which could offer a 100% detection and protection against drones. In order to increase the protection, different techniques need to be implemented simultaneously in the same system.

The work presented in this paper can definitely help future researchers to understand the process of decoding drone communication protocols or in the process of developing a counter drone system.

Acknowledgment

This article summarizes the work undertaken during an internship in the Joint ISR Service Line at the NATO Communications and Information Agency, The Hague, Netherlands.

I express my deepest thanks to Cornelis STOLK, the mentor of the project, for taking part in useful decision, for giving necessary advice and for guiding me throughout my internship. I choose this moment to acknowledge his contribution gratefully.

I would also like to extend a special thank you to Dr. Cristian COMAN from NATO Communications and Information Agency, The Hague and to my Ph.D. advisor, Prof. Ioan NICOLAESCU.

References

[1] Phuan, Y. (2016). Radiomonitoring / radiolocation Drone radiolocation. Drone alert! Available at: http://docplayer.net/53454605-Radiomonitoring-radiolocation-drone-radiolocation-drone-alert.html
[2] "Drone Detection System Radar | Drone Geolocation System | TCI", TCI International, 2018. [Online]. Available: https://www.tcibr.com/blackbird-integrated-geolocation-drone-detection-system/.
[3] Phuc Nguyen, Hoang Truong , Mahesh Ravindranathan, Anh Nguyen, Richard Han and Tam Vu, "Drone Presence Detection by Identifying Physical Signatures in the Drone's RF Communication", MobiSys '17 Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services Pages 211-224, 2017.
[4] Gerton de Goeij, Eildert H. van Dijken, Frank Brouwer, "Research into the Radio Interference Risks of Drone", Vianen, NL, May 2016.
[5] DSM informations available at https://wiki.paparazziuav.org/wiki/DSM.
[6] LabView informations available at http://www.ni.com/documentation/en/labviewcomms/1.0/2952r/overview/.
[7] Cypress Semiconductor, "WirelessUSBTM LP/LPstar and PRoCTM LP/Lpstar", datasheet, September 10, 2015.