UMASS AMHERST

UMass Amherst ECE 697KK Final Project

Intercepting Quad-Copter Spread Spectrum Communications

Ryan Lagoy

April 24, 2016

Introduction

Today, commercial drones are the must-have toy for children and adults alike. In the past year alone, the number of registered radio controlled aircraft outnumbered the amount of registered manned aircraft. And this should come at no surprise— the astoundingly entertaining technology has reached a price-point that many people can afford. This all may seem very great; however, have all security concerns been addressed, or has that been left to the wayside?

The purpose of the project is to analyze the communications link between a popular radio controlled quad-copter and the transmitter. Upon sufficient analysis of the commercial hardware, it is of interest to see if a relatively wideband and inexpensive software defined radio can intercept, demodulate, and decode the signal that this RF system uses to communicate. Below, is a concept of operations visualization that describes the overall scope of the project.

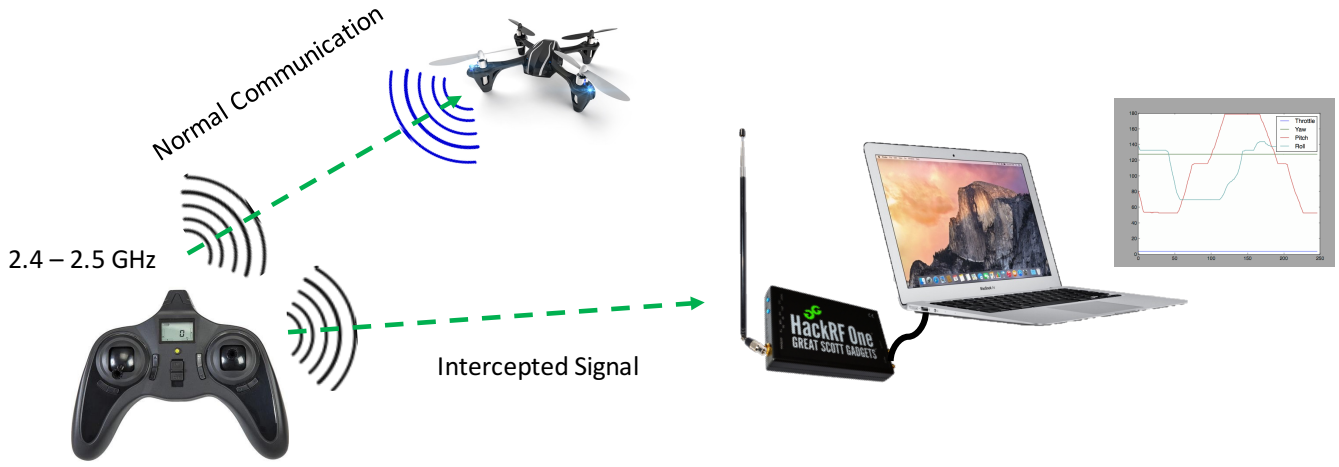

Figure 1. System Concept of Operations.

From the figure, you may have identified the two core devices used in this project: the Hubsan-X4-H107L quad-copter and the HackRF One software defined radio. The Hubsan quad-copter was chosen for this project because it is inexpensive, the communication protocol has been reverse-engineered by the hacker community, and it is one of the more popular copters available. As for the choice of software-defined radio, it was selected because it is relatively inexpensive and one was available to use. The radio is a half-duplex transceiver, operating from 1MHz-6GHz, and it uses an 8-bit ADC that outputs in-phase and quadrature signal components.

Now, a short overview of communications background information will be presented to the interested reader.

Spread Spectrum Communications

In the past, narrowband, low-frequency, and low data rate (between 27-72 MHz) RC systems were commonly used [11]. Until now, multiple users in a common area were forced to use different instantaneous frequencies, for if they shared a common frequency band, their devices would jam each other unintentionally and result in the improper operation of their devices.

With the current technology, users no longer need to worry about these issues. A concept called spread spectrum communication has been implemented, which allows users to operate within the same frequency band, across several channels, without affecting each other's devices. There are two main types of spread spectrum: Direct Sequence Spread Spectrum (DSSS) and Frequency Hopping Spread Spectrum (FHSS). Direct sequence spread spectrum is the idea that different signals move around in a single channel, while in FHSS, the signals move to completely different channels at a specified rate and frequency list [11]. Spread spectrum technology allows multiple users to operate in the same spectrum and increases the data rate limits that previously existed.

To provide a solid intuitive grasp on this concept, an analogy will be used as shown in the figure below [11]. The outdated narrowband system is represented by the narrow, one lane road. One car may be able to use this road (albeit at a slow pace), but if two cars are traveling towards each other, then surely they'd crash. In the spread spectrum analogy, many cars can use a freeway without any problems as they are able to travel in different lanes.

Figure 2. Communications Analogy. (a) Old Narrowband System (b) New SS System.

Frequency Shift Keying Modulation

A part from phase shift keying (PSK), which was taught in class, frequency shift shift (FSK) keying is commonly used in practice, as non-coherent receivers can be cheaply implemented in

hardware or software. This section will outline several core concepts about binary FSK which can be extrapolated from what we learned about in class about BPSK.

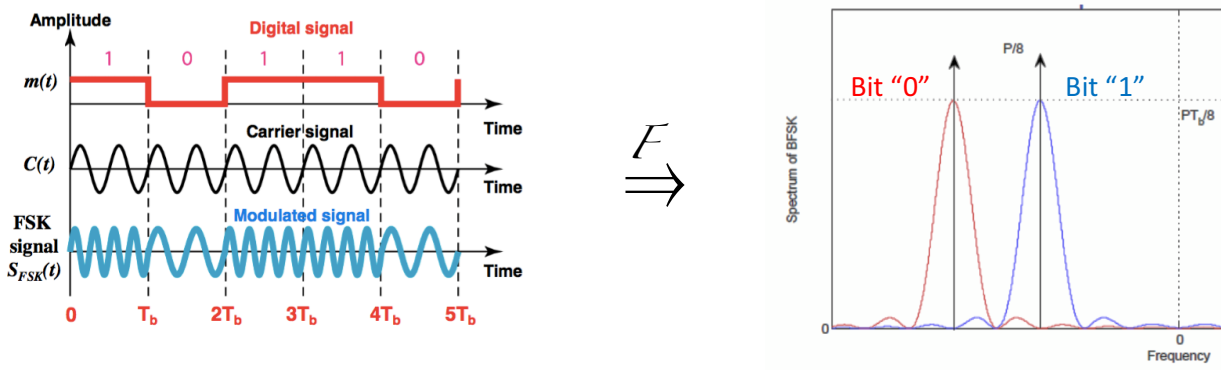

Figure 3. Frequency Shift Keying Overview. (Left) Time Domain. (Right) Frequency Domain [2].

In the figure above, the BFSK modulation scheme is shown pictorially. On the left, a time domain representation is shown for a digital signal. The digital signal, m(t), is modulated with a carrier signal, in which a higher frequency is mapped to a "1" bit, and a lower frequency is mapped to a "0" bit. The carrier frequencies of choice have to be mutually orthogonal as was shown with the PSK modulation scheme, but these special requirements are outside of the scope of the project. However, it is important to relate the bit error rate (BER) to the signal to noise ratio $\left(\frac{E_b}{N_0}\right)$. This expression is shown below [2]:

$$BER = P_{be,BFSK} = \frac{1}{2}\text{erfc}\left(\sqrt{\frac{1}{2}\frac{E_b}{N_0}}\right)$$

For example, to achieve a BER of 0.1%, the signal to noise ratio has to be at least 9.802dB. This is an important take-away as this will be used in future sections to characterize the performance of the receiver.

In the following sections, three core subsystems will be analyzed: (i) Hubsan-X4 quad-copter communication link (ii) Hubsan-HackRF communication link, and the (iii) digital receiver required to demodulate and decode the intercepted signal.

Hubsan Quad-Copter Analysis

In this section, the Hubsan-X4-H107L quad-copter system will be characterized and analyzed. The overall calculated range of the system will be compared with what is quoted by the company, and several plots are provided which outline the performance of the system.

Hardware Overview and Block Diagram

First, to fully understand how it operates, the system was reverse-engineered by observing the physical hardware and layout of the circuit board. The product was disassembled and several pictures were taken which shed some light on the system architecture so that a block diagram could be created. A viewgraph is show below which highlights some key findings.

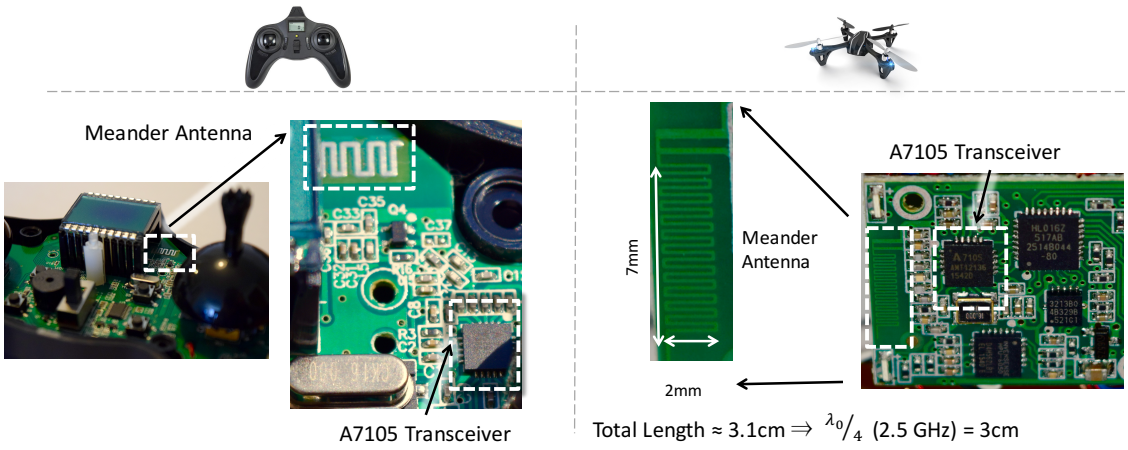

Figure 4. Physical Architecture of Hubsan System.

The Hubsan system utilizes an A7105 frequency shift keyed transceiver and a meander antenna for radiation of the signal. The meander antenna is larger in the controller than in the copter itself, which may suggest larger gain; however, a portion of the antenna is shielded by the LCD screen which may reduce the overall gain of the antenna. The total length of the meandered trace is roughly 3.1 cm in both cases, which is inline with the quarter-wavelength distance at 2.5 GHz in free-space.

Knowing that the Hubsan devices utilized a simple A7105 transceiver chip, it was possible to find a datasheet which provided useful information about the chip that could be used to characterize the entire system. Through disassembly and reading the datasheet, the following detailed block diagram of the antenna/transceiver system was created.

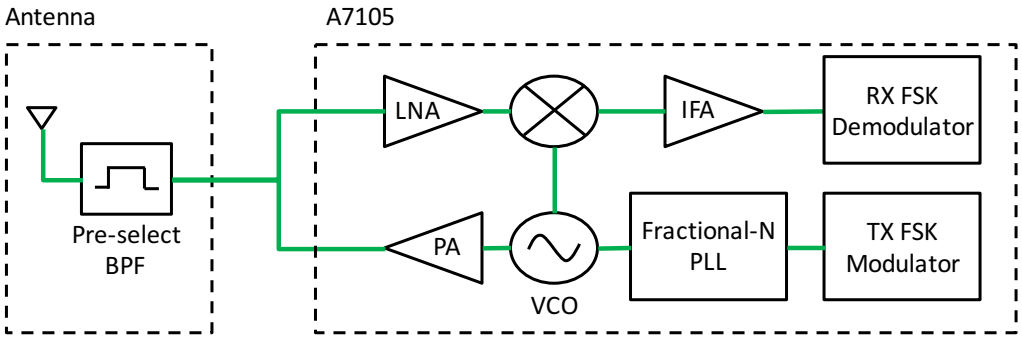

Figure 5. Block Diagram of Hubsan Transceiver System [1].

The meander antenna is by nature narrowband, as it is an electrically small element, so it acts as a pre-select band-pass filter in the 2.4 – 2.5 GHz frequency range. The A7105 chip is a half-duplex transceiver which means that it can both transmit and receive but at different times. On the receive side, there is an LNA which sets the noise figure of the system, and it is mixed down to an intermediate frequency where the signal is demodulated and decoded. On the transmit side of the device, a modulator, fractional-N phase locked loop, voltage controlled oscillator, and power amplifier is used for proper signal conditioning before entering the antenna. The fractional-N phase locked loop uses negative feedback to produce a highly stable desired output frequency at the output of the VCO.

Performance Analysis

Next, the main specifications for each component in the block diagram is outlined in Table 1. Note, the datasheet only provided the sensitivity of the receiver given a BER=0.1% at given data rates, but it is possible to derive the noise figure of the receiver from these values. This is an important calculation because the noise figure value can be used for other calculations and trade studies, while the sensitivity for a specific bit error rate cannot be used directly. The expression below was derived to relate the sensitivity of the receiver and SNR_d to the noise figure of the system.

$$(F_{recv})_{dB} = Sensitivity_{dB} - (k * T_0 * B)_{dB} - (SNR_d)_{dB}$$

For this calculation above, an IF bandwidth of 500 kHz was used and the temperature of the system was assumed to be 290°K.

Table 1. Hubsan Receiver Specifications [1],[4].

Receiver

Specifications	Units	Min	Max	Used
Sensitivity [BER =0.1%]	dBm	-95	-107	-100
SNR_d	dB	-	-	9.802
IF Bandwidth	kHz	250	500	500
Mixer Gain	dB	6	24	24
LNA Gain	dB	0	24	24
Max Input	dBm	-	5	
F_{rec}	dB	12.19	0.19	7.19
Spurious Emission	dBm	-57	-47	-
Antenna Gain	dBi	-10	3	-2

Several values are outlined in green above because they were derived values, or values that weren't provided in any datasheet. The noise figure calculation was described above; however, it is important to discuss the gain choice for the meander antenna. Several referenced sources discussed experimental values for optimal gain of this type of antenna. Because this antenna is electrically small, lower efficiencies and gain is expected, and so a safety margin of 5dB from the maximum was used.

For the transmitter, several key specifications were recorded and shown in Table 2. The most important characteristics for link budget calculations are the range of the output power levels and the antenna gain. The other values are important as they are needed for demodulation in the digital receiver section of this report.

Table 2. Hubsan Transmitter Specifications [1],[4].

Transmitter

Specifications	Units	Min	Max	Used
Output Power	dBm	-20	1	-10
Frequency Deviation	kHz	124	186	186
Data Rate	Kbs	2	500	100
Out-Of-Band Emissions	dBm	-47	-30	-
Antenna Gain	dBi	-10	3	-2

Using the above information and the Friis transmission equation below, it is possible to determine the maximum range assuming R^2 uniform path loss.

$$R^2 = P_T \frac{G_T G_R \lambda^2}{(4\pi)^2 \, SNR_d \, k \, T_0 \, B \, F_{recv}}$$

For a BER=0.1%, the maximum expected range is 198.47m. On the Hubsan website, the advertised range is 100m, so this result might have arrived at empirically, the path loss model was not a basic one, the drone needs a lower BER to operate, or the company provided a margin on the maximum range. However, the theoretical result is within the same order of magnitude of the advertised result, so this model seems to hold well.

Several trends were observed with the given performance data. The specific values used for creating the graphs are shown in the tables above in the right-most columns labeled “Used,” unless otherwise specified. It is shown that as the transmit power from the A7105 chip is increased, the maximum range also increases. Also, several constant BER curves are plotted on the chart and this shows that as BER is decreased past 10^{-4}, the maximum range seems to converge.

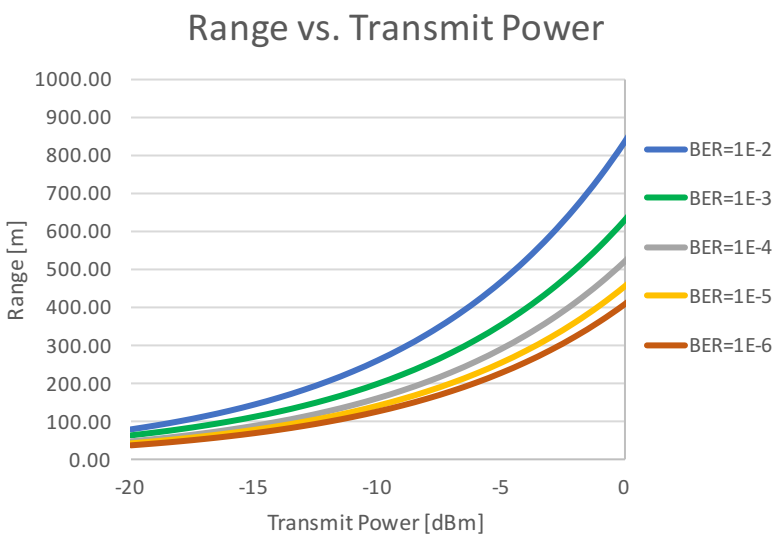

Figure 6. Range vs. Transmit Power.

Another interesting relationship is how the sensitivity is affected by the data rate of the transceiver. As shown, for low data rates, the sensitivity is the greatest, and it decreases as the data rate increases. As the BER decreases in several orders of magnitude, the sensitivity converges. This occurs because for slight increases in SNR, the BER decreases drastically.

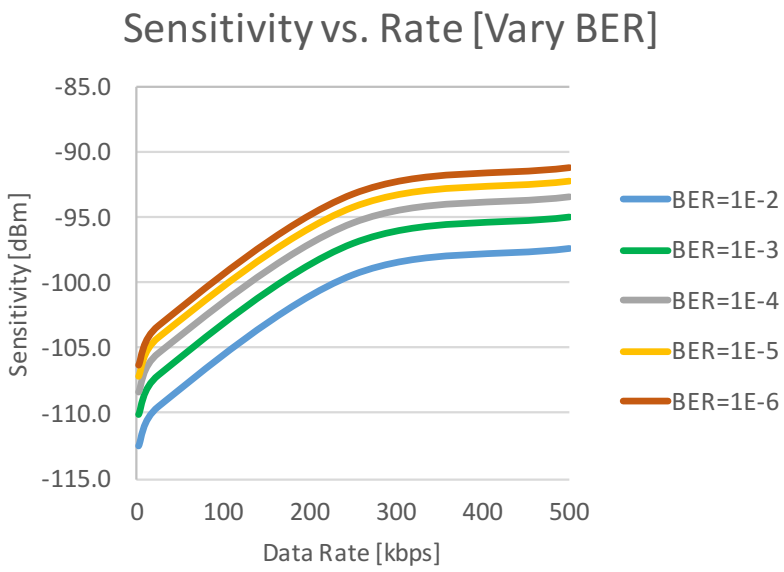

Figure 7. Sensitivity vs. Data Rate.

HackRF One Receiver Analysis

In this section, the HackRF One software-defined radio receiver subsystem will be briefly described and analyzed. A less intensive review will be completed on this device as the main purpose is to use this device to only intercept communications from the Hubsan. Also, the transmitter portion of the HackRF was not considered as this is outside the scope of this project. Lastly, a brief performance comparison between the A7105 transceiver and the HackRF One will be conducted in terms of maximum range for bit detection.

Hardware Overview and Block Diagram

The HackRF one is a relatively inexpensive software defined radio, and one of the best features is that it is open-sourced, which means that everyone has access to the entire design, including hardware schematics, firmware code, and software [9]. Thus, no reverse engineering was required as was needed for the Hubsan quad-copter. By browsing the GIT repository where all of this information was stored, the following block diagram was created.

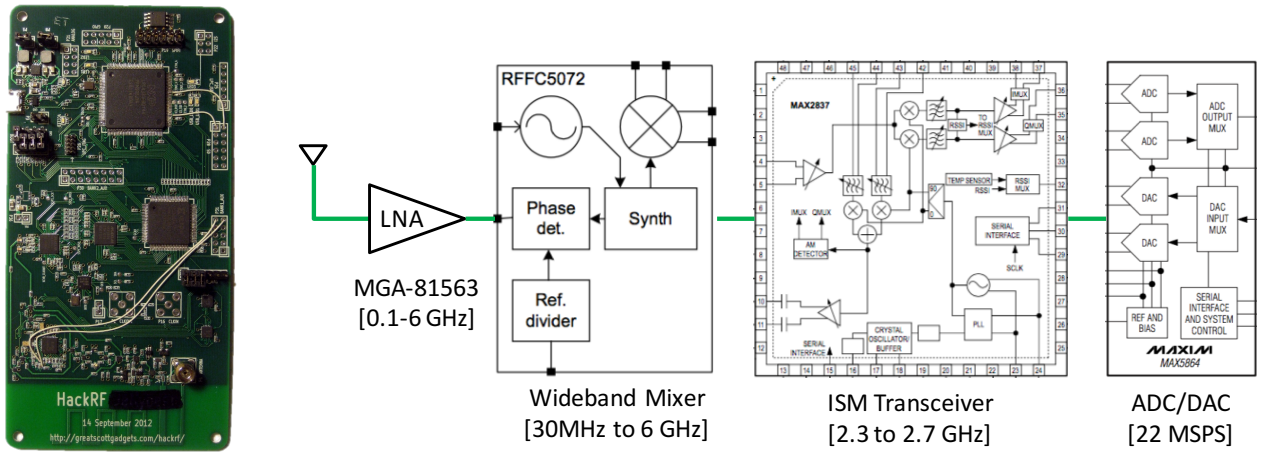

Figure 8. HackRF One Hardware and Block Diagram [9],[3],[7],[8],[12].

Performance Analysis

As mentioned above, only the receiver chain of the HackRF One was analyzed for the purpose of the project. However, the analysis was slightly more complicated for this system, as several discrete components were used in the chain instead of a fully characterized system on a chip, like

the A7105. The cascade analysis (using the general cascaded noise figure equations) is shown below in Table 3.

Table 3. HackRF Receiver Specifications and Cascade Analysis [3],[7],[8],[12].

Specifications	Units	Min	Max	Used
SNR_d [BER=0.1%]	dB	-	-	9.802
IF Bandwidth	MHz	2	22	4
Max Input	dBm	-	15	15
F_{LNA}	dB	2.7	3.5	2.7
G_{LNA}	dB	10.2	12.5	12.3
F_{mixer}	dB	10	15	10
G_{mixer}	dB	-	-2	-2
$F_{transciever}$	dB	2.3	27	2.3
$G_{transciever}$	dB	8	40	10
G_{BB}	dB	0	62	20
F_{rec}	dB	2.94	16.70	2.85
Antenna Gain	dBi	-10	2.15	0

As highlighted in the table above, the total cascaded noise figure is 2.85 dB which is much less than the 7.19 dB of the A7105 chip. The antenna used with the HackRF was a retractable monopole antenna with a maximum operating frequency of 1 GHz. The estimated gain of this antenna out of band was 0dBi. Lastly, a brief comparison between the HackRF and Hubsan systems is shown in the plot below. The BER performance is plotted against range, and one can see that the HackRF is significantly more sensitive than the A7105 chip and is able to theoretically intercept the signal from much further away than the controller can reach. For a BER of 0.1%, the maximum range for the HackRF is ~411.6m.

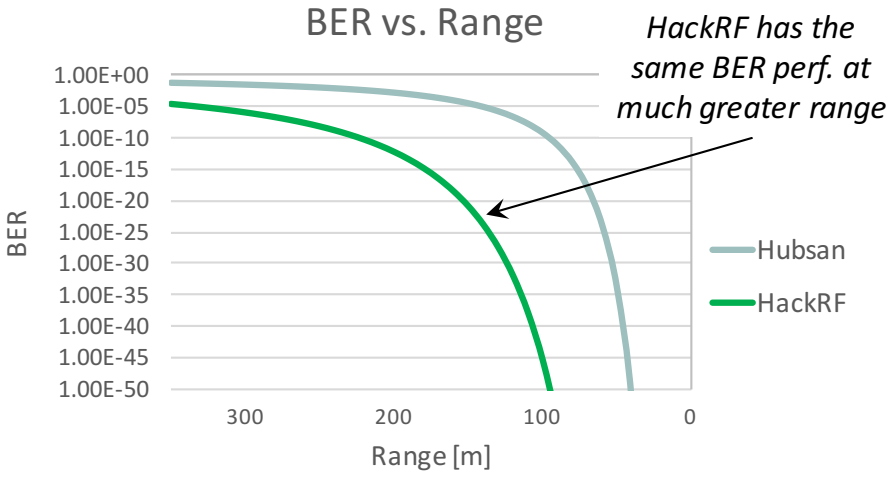

Figure 9. BER vs. Range (Hubsan vs. HackRF).

Digital Receiver Analysis

Lastly, a non-coherent frequency shift keyed digital receiver was implemented in a combination of GNURadio, C++, and Python code. This part of the project was highly intensive, requiring a significant insight into non-coherent digital modulation theory as well as understanding the complex (and proprietary) packet and data format of the Hubsan controller. Significant understanding of the course topics was necessary for successful demodulation of the signal. This section will *briefly* describe the design and results of the receiver implementation, as many lessons were learned from completing this part of the project.

Functional Block Diagram

Below is a functional block diagram of the digital receiver that was implemented for this project. This block diagram was created in GNURadio, which is a RF software platform which allows engineers to easily create radio subsystems with a given input source (HackRF) and output sink (file, plots, etc.). The light blue rectangles are modular functions called blocks that can be interchanged and connected freely, and GNURadio provides many of these which perform digital signal processing, such as filtering, frequency mixing, channel coding, etc. For this receiver, several blocks needed to be designed and implemented in C++, as they were highly specialized and not provided by default.

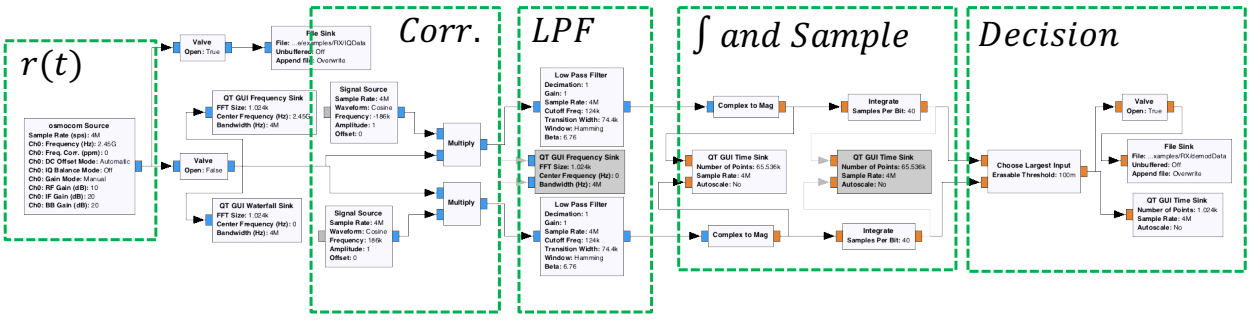

Figure 10. GNURadio Digital Receiver.

There are five main sections to this FSK receiver. First, there is the source, r(t), which provides the I and Q samples to the receiver. These samples are then fed through a correlator, which correlates the signal against a low frequency cosine wave (“0”) and a high frequency cosine wave (“1”). Next, the two complex signals were passed through a low-pass filter with cutoff frequency slightly above the data rate of the signal. Then, these signals were integrated and sampled. The optimum maximum likelihood decision rule was implemented next, and the output bit sequence was then saved to a file. At several points in this chain, the spectrum or time domain

representation was plotted to see how the system operates in a specially designed graphical user interface (GUI).

Interception Results

As mentioned in the earlier sections, the Hubsan communication system was spread spectrum. This system in particular spends a moment in the initialization to find a channel within the 2.4 – 2.5 GHz spectrum that has the lowest total power and sets the system to that that channel permanently for duration of the flight (this is referred to as DSSS as described in a section above). Once the channel is set, the controller and the quad-copter exchange a series of handshakes to establish the link, and after about 20 packets are exchanged (each sent out every 10ms), the controller is able to send flight signals to the quad. To intercept the signal, the user has to search manually across the band for the FSK signal using the special GUI. In Figure 11, the signal has been located, and the FSK power spectral density is plotted on the left. On the right hand side, a waterfall plot of the signal over time is shown. One can see the variation of the frequencies (bit sequence) in this waterfall plot.

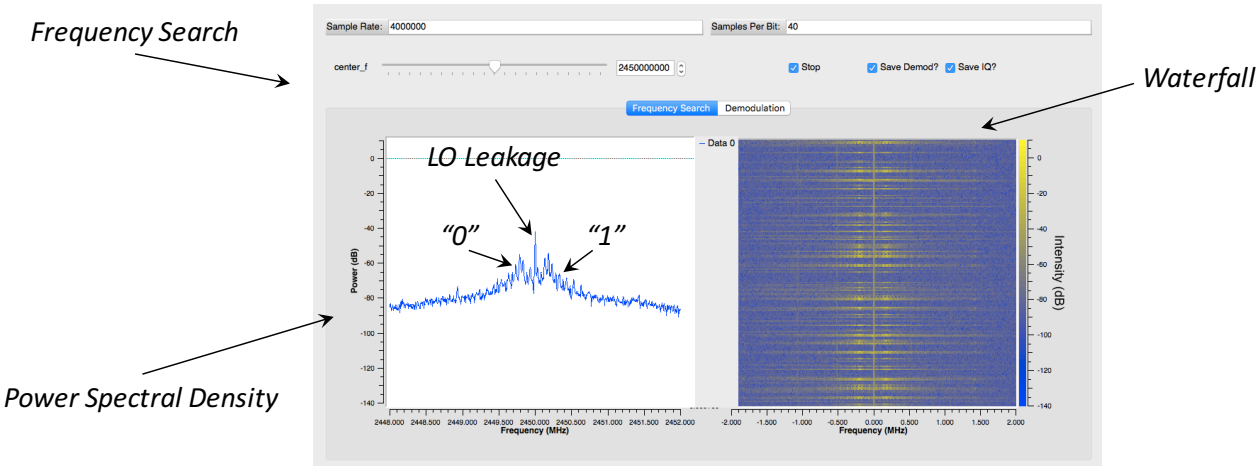

Figure 11. Graphical User Interface.

At the output of the receiver is a continuous flow of bits. A separate Python script (attached in the Appendix) was developed to identify the packets from the controller. The packets contain an 8-byte preamble, which is a series of alternating 0's and 1's used for packet identification, and the flight control data. Another engineer had reversed engineered the protocol with a logic analyzer and provided a protocol document that is freely available— this was extremely valuable, as decoding the bit stream would have been near impossible to do within the time constraints [6]. The flight packet data structure is shown below.

```
                                1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|A|B|C|D|E|F|G|H|I|J|K|   L   |M|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   Flight Packet Data Structure
              Figure 5.
```

Tag	Len	Description
A	1	Always set to \x20 for flight control packets.
B	1	NULL
C	1	Throttle
D	1	NULL
E	1	Yaw
F	1	NULL
G	1	Pitch
H	1	NULL
I	1	Roll
J	1	Additional Control Flags
K	1	Always set to \x19 in testing.
L	4	NULL
M	1	CHECKSUM (see section X)

Figure 12. Flight Packet Structure [6].

The figure below is a visualization of the transmitted packet which was identified in the Python script that was written. One can clearly identify the preamble in the beginning of the packet. Using the guideline of the flight packet structure above, the bit stream was successfully decoded.

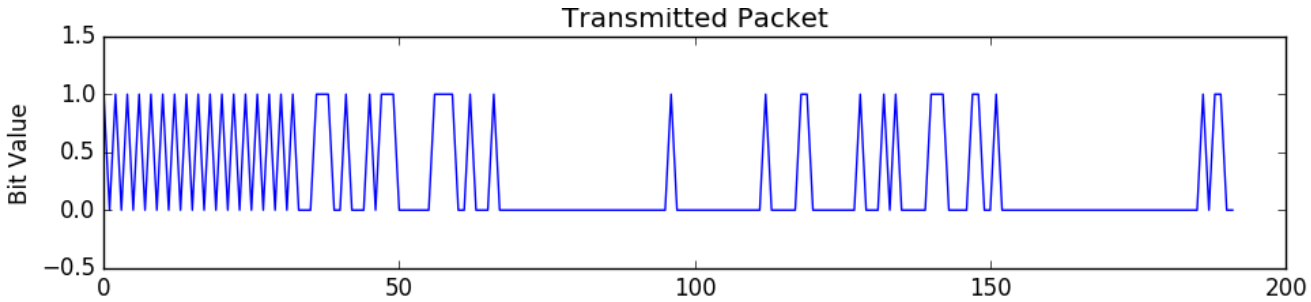

Figure 13. Transmitted Packet.

To test the accuracy of the digital receiver, the quad-copter was flown in a specific and identifiable pattern while the signal was being intercepted by the HackRF One. The flight pattern was as follows:

1. Lift off of the ground.
2. Fly in a clockwise circle.
3. Land on the ground.

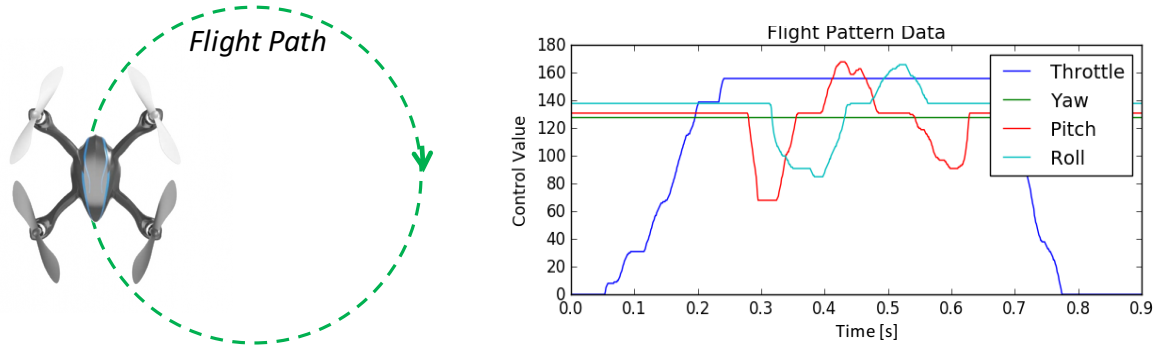

Figure 14. Decoded Data for Given Flight Path.

The chart in Figure 14 plots the decoded data, and the results show the actual throttle, yaw, pitch and roll of the quad with the simple flight path described above. It is especially clear to see the throttle increase to equilibrium for the take-off and then decrease for the landing.

Conclusion

While commercial quad-copters may be enjoyable to fly, they are by no means secure. RF signals from one of the best-selling quad-copters on Amazon.com can be intercepted by a more sensitive and powerful external transceiver (such as the HackRF), as was shown by completing this project.

To show that the Hubsan communication link is not particularly secure, this report discussed spread spectrum technology and the frequency shift keying modulation scheme used in the Hubsan communication system design. To understand how the system worked, the devices were disassembled so as to view the hardware components for further analysis. The link between the Hubsan controller and the quad was analyzed to show that the maximum range is approximately 198.47m with the performance specifications of the A7105 chip and meander antenna. Next, the HackRF One was analyzed for its receiver performance. It was shown that the HackRF has a roughly 4dB less noise figure, allowing it to receive the FSK signal at a much further range than the A7105 chip. Lastly, a digital receiver was implemented to demodulate and decode the incoming RF I and Q data stream from the HackRF.

When time permits, I'd like to analyze the HackRF transmit capabilities as well as the laws that govern the 2.4GHz ISM band. After sufficient analysis, I'd like to attempt to create a link between the HackRF One and the quad after a handshake with the controller is completed (to simplify the task).

Lastly, while not legal and highly unethical, it is fathomable to imitate the the transmitted signal to gain control of the quad-copter, even while someone else was controlling it with their transmitter. From a defense industry perspective, this is particularly worry-some and should definitely be addressed. Also, as an ethical engineering aside, it is important for us to be responsible in our designs and understand any adverse implications that may result from shortcuts taken such as leaving seemingly insensitive wireless signals unencrypted. While improving technology rapidly is fun and making profits is important, nothing is more important than being safe and addressing every security concern properly!

References

[1] AMICCOM, "2.4G FSK/GFS Transceiver," A7105 datasheet, Feb. 2010.

[2] Atlanta RF, "Link Budget Analysis: Digital Modulation, Part 2," Lecture Series, Jun. 2013.

[3] Avago Technologies, "0.1–6 GHz 3 V, 14 dBm Amplifier," MGA-81563 datasheet, May 2010.

[4] E. Lim, Z. Wang, S. Zhang, Y. Jiang, "Compact USB WLAN Printed Meander Line Antennas," *Procedia Engineering*, vol. 15, pp. 2516-2520, 2011.

[5] Freescale Semiconductor, Inc., Appl. Note AN2731, Sep. 2015.

[6] J. Hung, "Hubsan X4 H107L Quadcopter Control Protocol," Remote Control Protocol Library. Nov. 2014. Available: www.jimhung.com. [Accessed April 13, 2016].

[7] Maxim Integrated, "2.3GHz to 2.7GHz Wireless Broadband RF Transceiver," MAX2837 datasheet, Jul. 2015.

[8] Maxim Integrated, "Ultra-Low-Power, High-Dynamic- Performance, 22Msps Analog Front End," MAX5884 datasheet, Oct. 2003.

[9] M. Ossmann, "hackrf," Github Repository, Apr. 2016. Available: https://github.com/mossmann/hackrf. [Accessed April 24, 2016].

[10] M. Walters, "gr-hubsan," Github Repository, Apr. 2016. Available: https://github.com/miek/gr-hubsan. [Accessed April 24, 2016].

[11] RC Model Reviews, "2.4GHz Radio Control Explained," *rcmodelreviews.com*, 2008. [Online]. Available: http://www.rcmodelreviews.com/spreadspectrum01.shtml. [Accessed April 24, 2016].

[12] RF Micro Devices, "Wideband Synthesizer/VCO With Integrated 6GHz Mixer," RFFC5071/5072 datasheet, 2012.

Appendix

Python Decoding Code

```
import scipy as sp
import numpy as np
import matplotlib.pyplot as plt

print 'Loading Data! \n'
sampleArray=np.fromfile('demodData', dtype='float32')

# channel ID 25 : 2.425

# Get the location of the transmitter in 2.4GHz channel
# osmocom_fft -a hackrf -v -f 2.4e9 -g 15

# Record data in GNURadio with sampling frequency 8M

# Convert to Binary From List
def list2bin(binList):
        if (binList==[]):
                return -1
        return int("".join(map(str, binList)),2)


bits=list(sampleArray)
bitInt=[]

# Pick out the packet
packet=[]
#preambleControl=[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]
preambleControl=[1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]
#preamble2=[0,1,0,1,0,1,0,1,0,1,0,1,0,1]
for i in range(0,len(bits)-len(preambleControl)):

        checkPreamble=bits[i:i+len(preambleControl)]
        if (checkPreamble == preambleControl):
                packet.append(bits[i:i+192])

print 'Number of Packets: ', len(packet), '\n'
# Decode the packets
```

```
# Take the last 16-bytes of the packet

throttleTS=[]
yawTS=[]
pitchTS=[]
rollTS=[]
packetInt=[]


for i in range(0,len(packet)):

        packetInt=[]
        for j in range(0,len(packet[i])):
                packetInt.append(int(packet[i][j]))


        preamble=hex(list2bin(packetInt[0:8*4]))
        a7105_id=list2bin(packetInt[8*4:8*8])
        flightControl1=hex(list2bin(packetInt[8*8:8*9]))



        if list2bin(packetInt[8*8:8*9])==32:
                throttle=list2bin(packetInt[8*10:8*11])
                yaw=list2bin(packetInt[8*12:8*13])
                pitch=list2bin(packetInt[8*14:8*15])
                roll=list2bin(packetInt[8*16:8*17])
                flightControl2=hex(list2bin(packetInt[8*17:8*18]))
                flightControl3=hex(list2bin(packetInt[8*18:8*19]))
                tx_id=list2bin(packetInt[8*19:8*23])
                checksum=list2bin(packetInt[8*23:8*24])


                print 'Preamble: ', preamble
                print 'Flight Register: ', flightControl1
                print 'RF Chip ID: ', a7105_id
                print 'TX ID: ', tx_id
                print 'Throttle: ', throttle, ', Yaw: ', yaw, ', Pitch: ', pitch, ', Roll: ', roll, '\n'

                '''
                print flightControl2
                print flightControl3
```

```
                print tx_id
                print checksum, '\n'
                '''

                throttleTS.append(throttle)
                yawTS.append(yaw)
                pitchTS.append(pitch)
                rollTS.append(roll)

        else:
                packet[i]=np.zeros((1,192))


plt.figure()
plt.title('Transmitted Packet')
for i in range(0,1):
        plt.plot(packet[i])
plt.ylim([-.5,1.5])
plt.ylabel('Bit Value')
plt.xlabel('Bit Number')

time=np.array(range(0,len(yawTS)))*0.001

plt.figure()
plt.title('Flight Pattern Data')
plt.plot(time,throttleTS)
plt.plot(time,yawTS)
plt.plot(time,pitchTS)
plt.plot(time,rollTS)
plt.ylabel('Control Value')
plt.xlabel('Time [s]')
plt.legend(['Throttle', 'Yaw', 'Pitch', 'Roll'])
plt.tight_layout()
plt.show()
```