Final Project: RF remote (CC1350)

Course: Advanced Computer Systems (Fall 2018)

March 2018

Lecturer: Prof. Sivan Toledo

Students:

Michael Ozeri I.D: 302444229

Ran Toledo I.D: 200024438

<u>Summary:</u>

In this work we were requested to mimic a specific door car remote control using of the shelf and hardware tools.

<u>The main steps taken are:</u>

A. Record the RF signal generated by the car remote using the SDR SW and a USD sniffer.
B. Analyze the record using MATLAB tool and try to extract the code and its accompanying parameters such as BAUD rate.
C. Program the CC1350 evaluation board using the SmartRF studio7 SW tool such a way that it shall be recognized as a car remote by a remote control board supplied with the car door remote.

We succeeded in copying the remote control signal after handling these problems: (see appendix)

1. The car remote use amplitude shift keying (OOK), but the CC1350 has only frequency shift keying, FSK and GFSK. We reviewed the data sheet deeply but couldn’t find any option for the generation of amplitude shift keying.
   a. After consulting our lecturer we found out there is a capability of OOK transmitting through legacy options in the cc1350
2. The CC1350 and the SDR append a preamble to each packet in the massage, not only at the start of the massage
   a. We set a few packets in the transmitted packet thus disabling the preamble and sync word which we set on 0.
3. There is no option to turn off the sync words in each packet. So we tried to put some of the code as a sync words.
   a. See 2.a

The first step we took was to learn the SDR tool and its user interface, by trying to listen to various FM commercial radio stations, for instance in the range 0f 90MHz to 130MHz. We chose radio type AM. A screenshot of the SDR window while listening to music at 103.98 MHz station is shown in figure 1.

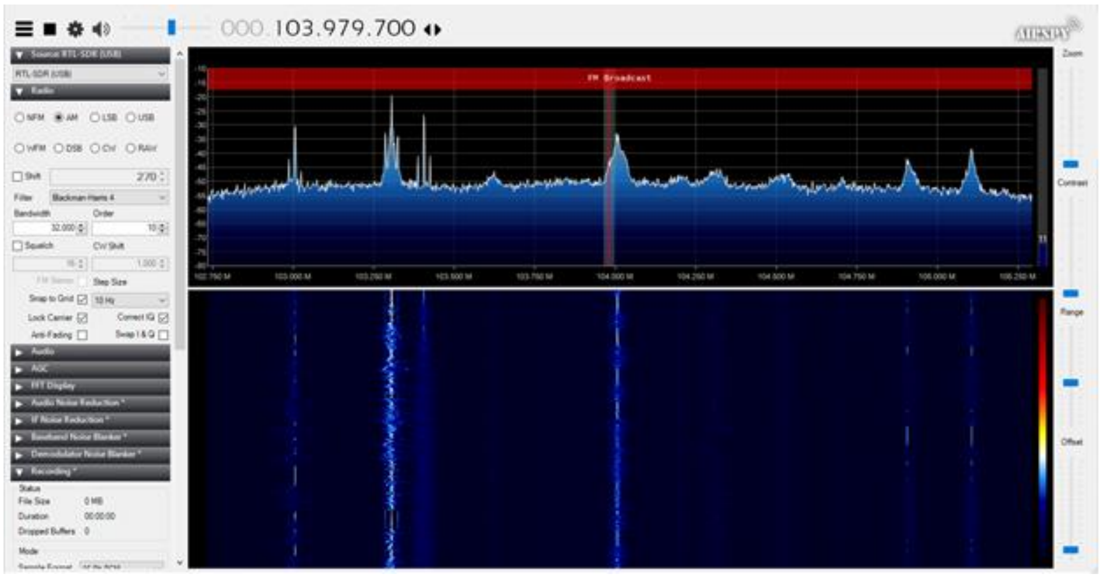


Figure 1

After gaining some experience with the SDR SW tool, we experimentally tried to find the frequency at which the car key radio transmits. After some trials we found that it transmits in 433.87MHz. If we press the button long enough we can see a long train that repeatedly transmit the same sequence as shown in figure 2.

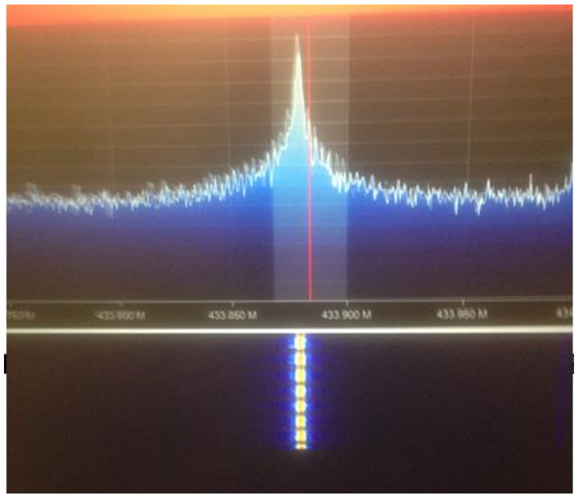
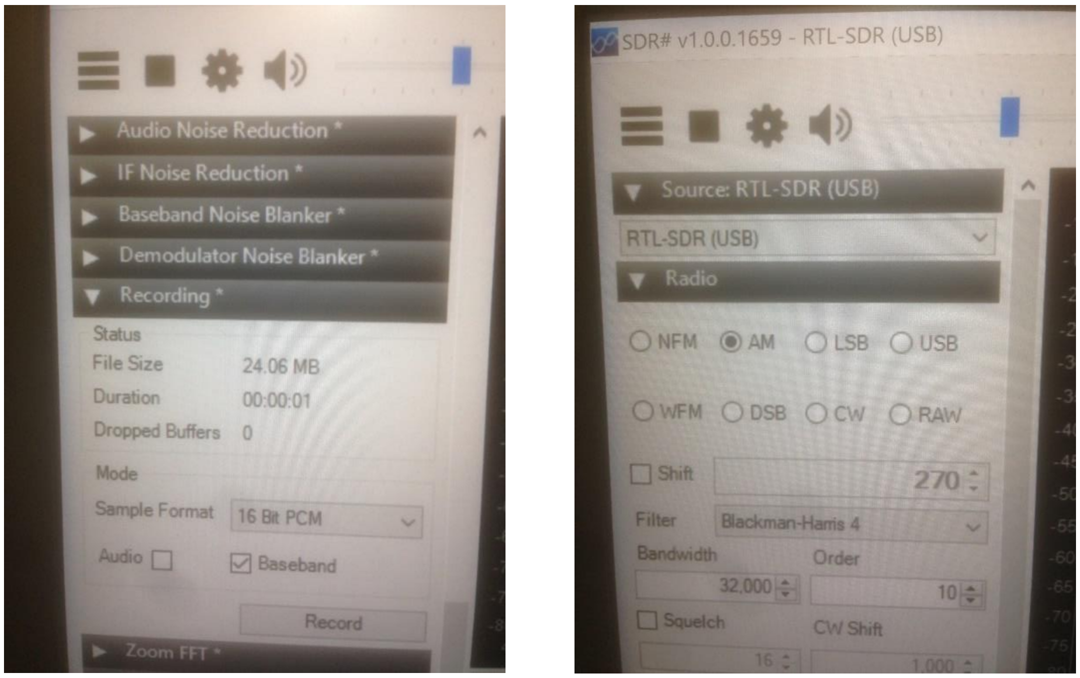


Figure 2

A screenshot of the SDR menu selection is shown in figure 3


Audio Noise Reduction *
IF Noise Reduction *
Baseband Noise Blanker *
Demodulator Noise Blanker *
Recording *
Status
File Size 24.06 MB
Duration 00:00:01
Dropped Buffers 0
Mode
Sample Format 16 Bit PCM
Audio ☐ ☑ Baseband
Record
Zoom FFT *

SDR# v1.0.0.1659 - RTL-SDR (USB)
Source: RTL-SDR (USB)
RTL-SDR (USB)
Radio
NFM [✓ AM] LSB USB
WFM DSB CW RAW
☐ Shift 270
Filter Blackman-Harris 4
Bandwidth 32,000 Order 10
☐ Squelch 16 CW Shift 1,000

Figure 3

We then proceed to record the RF transmission generated, due to a long press close-button key. In order to assure that a whole pattern is recorded, we started to record before pressing the close button and stopped recording after releasing the button Parameters: radio AM, sample format 16 bit PCM. A MATLAB m file used to process and generate the plot is:

```
clc;
[y_close3,Fs]=audioread('C:\Users\Downloads\sdrsharp-
x86\SDRSharp_20180407_170802Z_433900000Hz_IQ.wav');
absyclose3=abs(y_close3);
%absyclose3=absyclose3(5.92e6:6.08e6);
plot(absyclose3);
grid on;
figure;
for k=1: length(absyclose3)
   % if absyclose3(1,k)> 0.05
     if absyclose3(1,k)> 0.05
        absyclose3(1,k)=1;
    else
        absyclose3(1,k)=0;
    end
end

plot(absyclose3);
ylim([-0.2 1.2]);
grid on;
```

The plot generated is shown in figure 4:

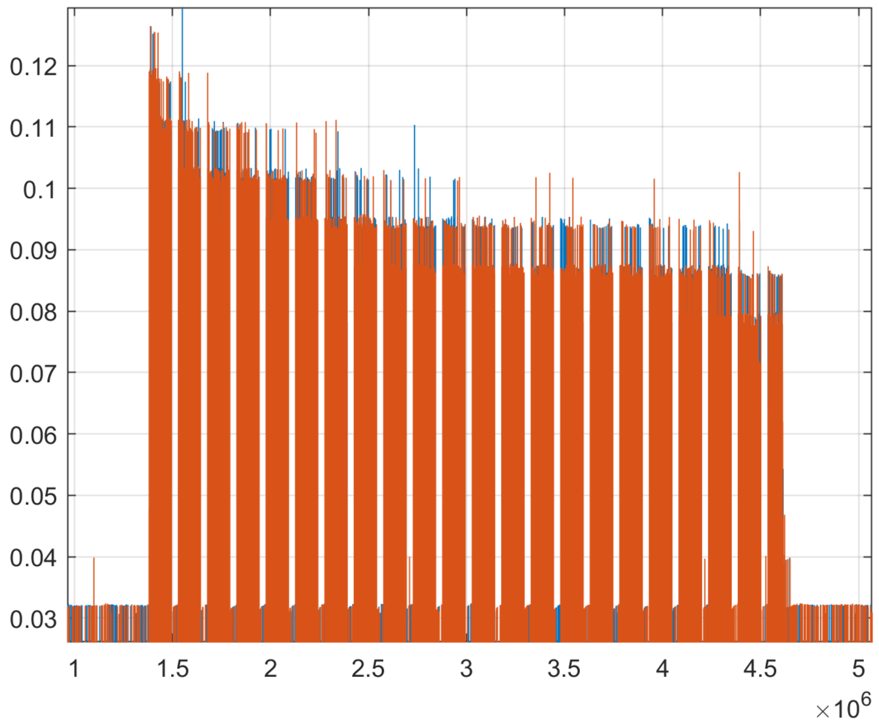


Figure 4

From figure 4 we can see than the car key transmitted the code repeatedly. We assume that the amplitude drop was caused by the battery voltage in the car key drop during the long button press. So the next step was to try to extract one of the pattern from the recorded sequence. Figure 5 shows the plot result of one of the patterns in the bursts.

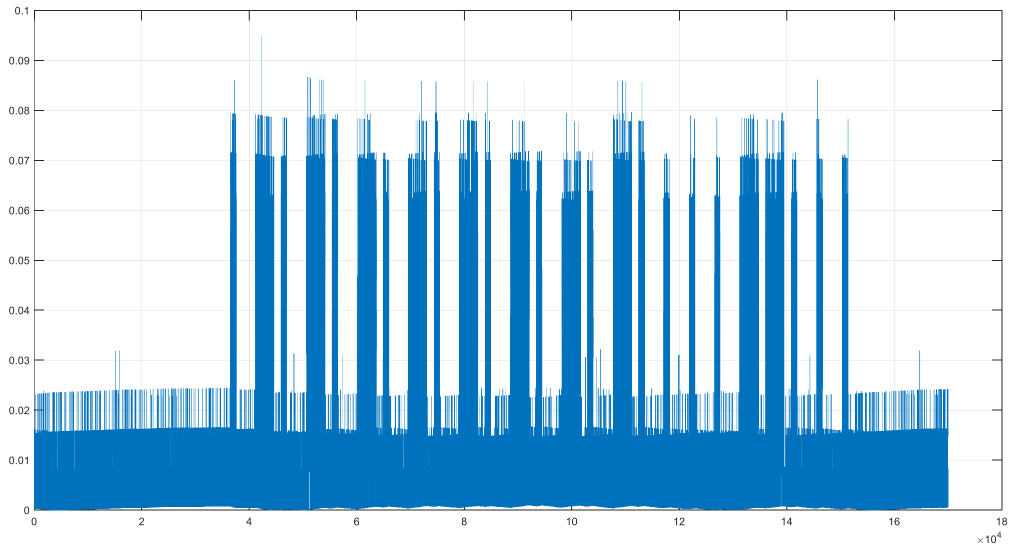


Figure 5

The last step was a decision stage. A simple ‘if’ command decided whether the signal is a binary ‘1’ (> decision threshold of 0.05), or binary ‘0’ (else). The result is shown in figure 6.

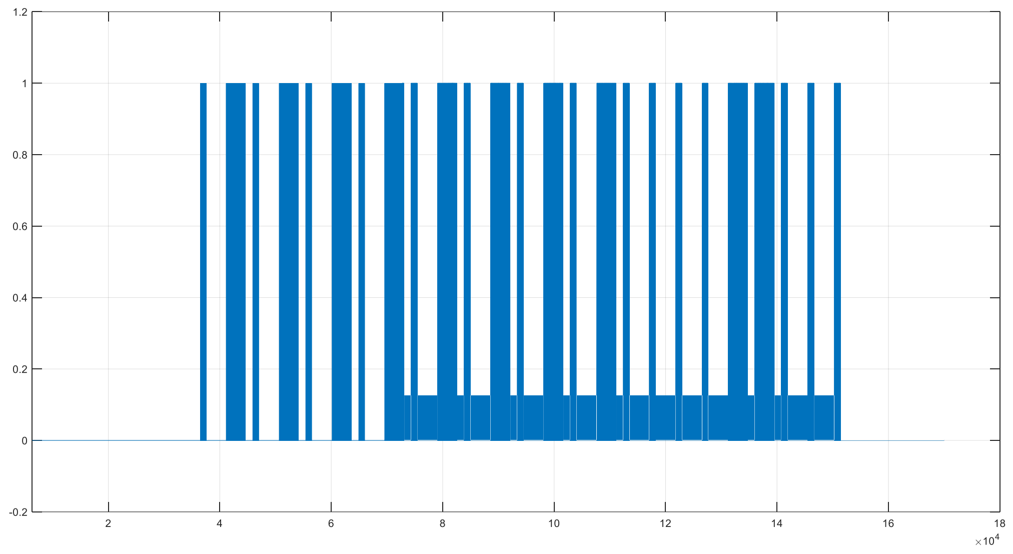


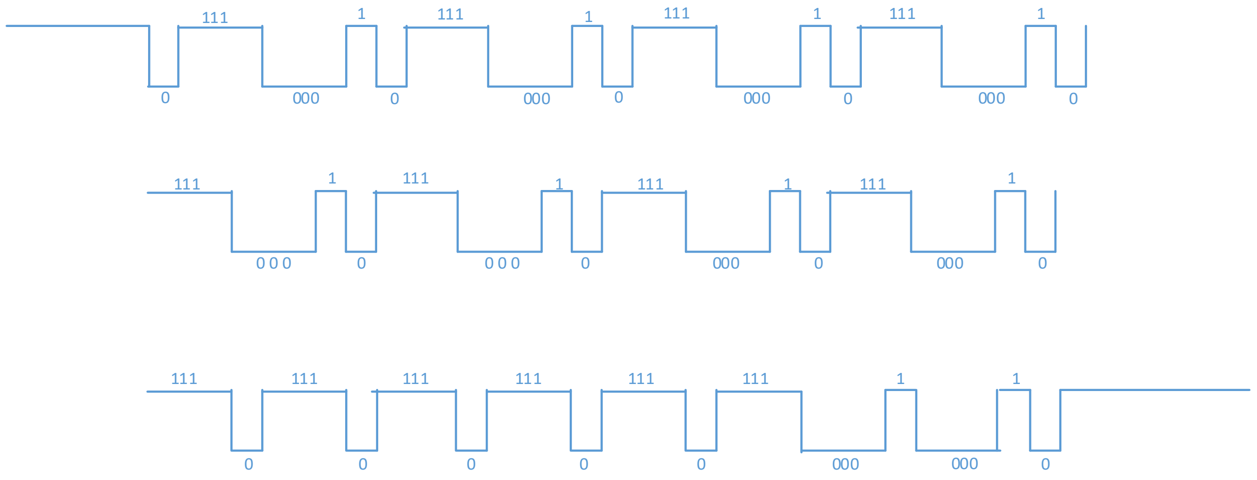
Figure 6

If we refer just to the envelope of the signal, then the smallest bin width is a single '1' bit (or '0'), and by comparing its width to the other bins we can extract the close button code:

close button code:
1 0 0 0 1 1 1 0 1 0 0 0 1 1 1 0 1 0 0 0 1 1 1 0 1 0 0 0 1 1 1 0 1 0 0 0 1 1 1 0 1 0 0 0 1 1 1 0 1 0 0 0 1 1 1 0 1 0 0 0 1 1 1 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 0 1 1 1 0 1 0 0 0 1 0 0 0 1

And using the same process we got the code for the open button:
Open code:
100011101000111010001110100011101000111010001110100011101000 11101
00010001000100010001000111011101

A sketch of the code is presented in figure 7:



We tried also to invert the code, replacing zeros by ones and ones with zeros.

Extracted data:

The MATLAB audioread M function gives us also the sampling rate of 3,200,000 samples/sec. This value enables us to calculate the symbol time by multiplying the length of the symbol in the MATLAB figure with the sampling interval of t=1/3.2e6.

So the symbol rate used by the car remote is: 360Us

The symbol rate is: 1/360uS = 2778 symbols/sec

The time interval between adjacent packets in the massaging frame is calculated the same way:

Tdelay = 11mSec

Number of packets typically transmitted by the car remote depends on the key pressing length and seems to be between 3 to 8 packets.

Since no presetting scenario fitted our needs, we chose the first one within the frequency range of 431MHz-527MHz and modified it using the graphic user interface, and the command view that allowed us to directly modify the registers values. The values used to operate the SDR were:

- Fc = 433.86MHz carrier frequency, almost similar to what the car remote generates
- symbol rate: 2.78931 kBaud
- Deviation 25 Khz
- Rx filter: 39 kHz
- Packet count: 5 (we tried 1 -8)
- CRC- no
- Codding- no
- Modulation= FSK (tried also GFSK, no option for ASK)
- Packets interval : 11mSec

Some graphical results:

Figure 8 shows a message transmitted by the CC1350. It contains 5 similar packets with a 11mSec time interval between adjacent packets composing the message:

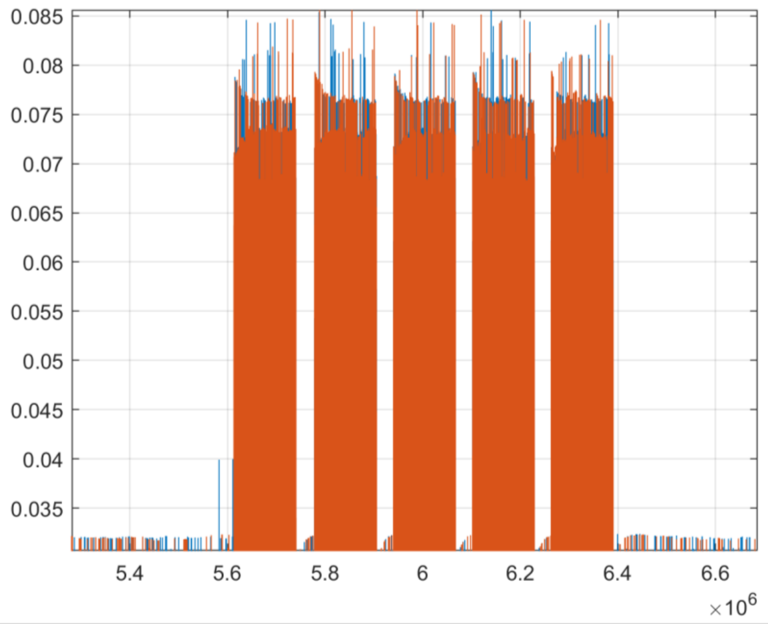


Figure 8

Figure 9 shows one packet by zooming into figure 8:
In figure 8, two data points that indicate the start and end packet intervals are shown. The time interval can simply be calculated:

Tdelay: (6.067e6 – 6.102e6)* 1/3,200,000 = 10.94 mSec. The difference from the expected 11 mSec interval is probably caused by the figure zoom out.

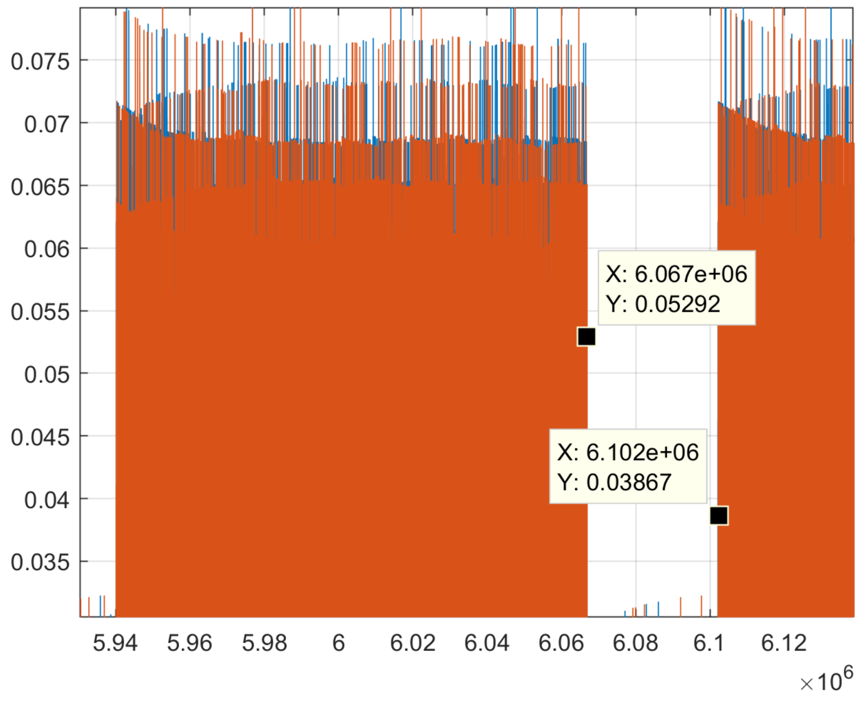


Figure 9

Figure 10 shows the raw data before a decision stage. It is clearly show that the frequency was modulated and not the amplitude:

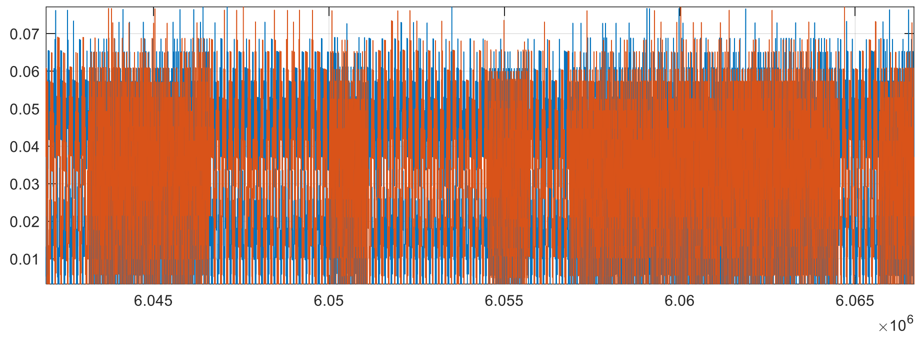


Figure 10

Figure 11 shows the transition between symbols from the 'mark' frequency to 'space' frequency.

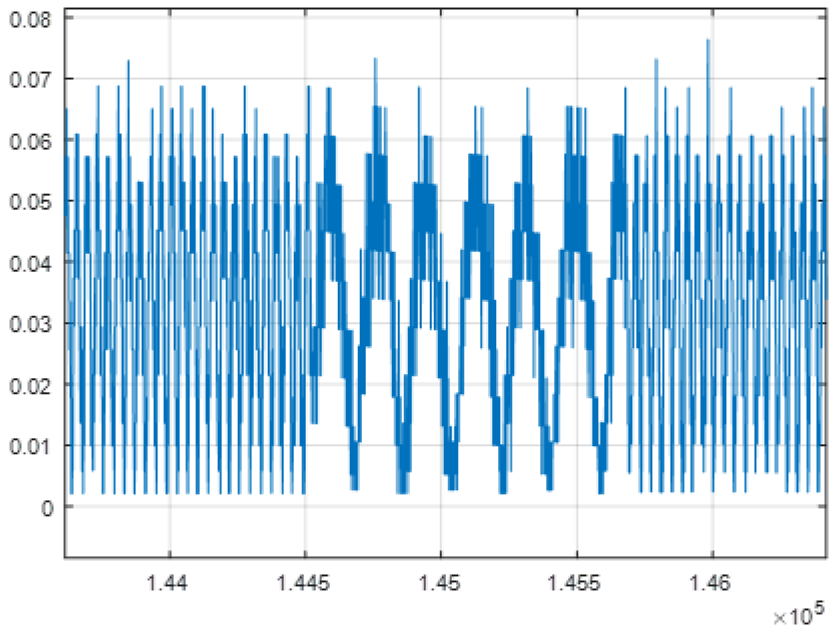


Figure 11

A single packet was extracted, and after a decision stage (an ‘if‘ MATLAB statement) that converted the levels into ‘0’ and ‘1’s is shown in figure 11

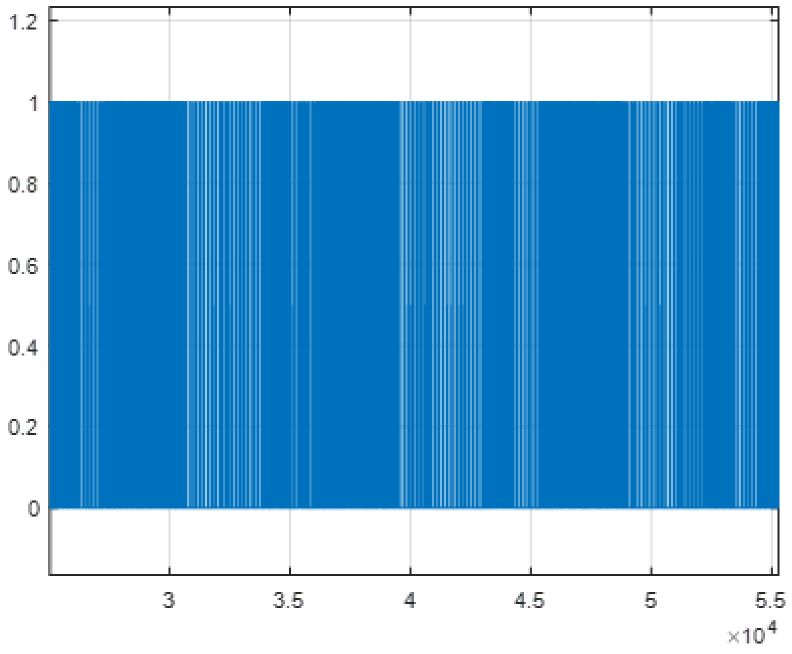
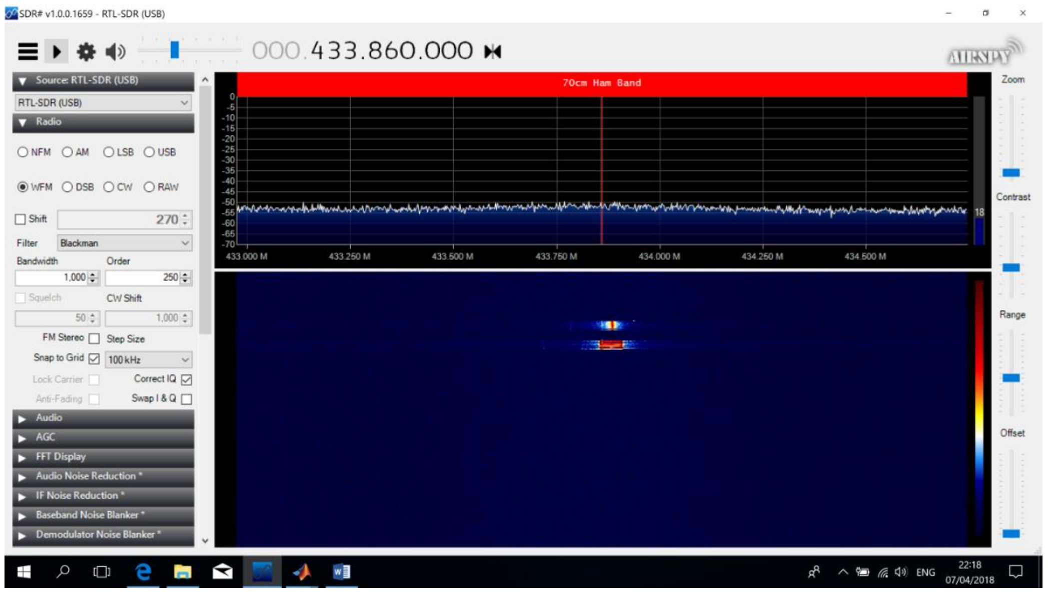


Figure 12

Finally, a screenshot pf the SDR showing the CC1350 generated packet followed by the car remote packet (Figure 13):


SDR# v1.0.0.1659 - RTL-SDR (USB)

000.433.860.000

Source: RTL-SDR (USB)

RTL-SDR (USB)

Radio

NFM AM LSB USB

[✓ WFM] DSB CW RAW

Shift 270

Filter Blackman

Bandwidth 1,000 Order 250

Squelch 50 CW Shift 1,000

FM Stereo Step Size

Snap to Grid ✓ 100 kHz

Lock Carrier Correct IQ ✓

Anti-Fading Swap I & Q

Figure 13

Appendix

After consulting our lecturer – we succeeded:
Even though the OOK modulation does not appear in the launchpad datasheet as shown in figure 14, we realized that the OOK option is still available in the SmartRF Studio.

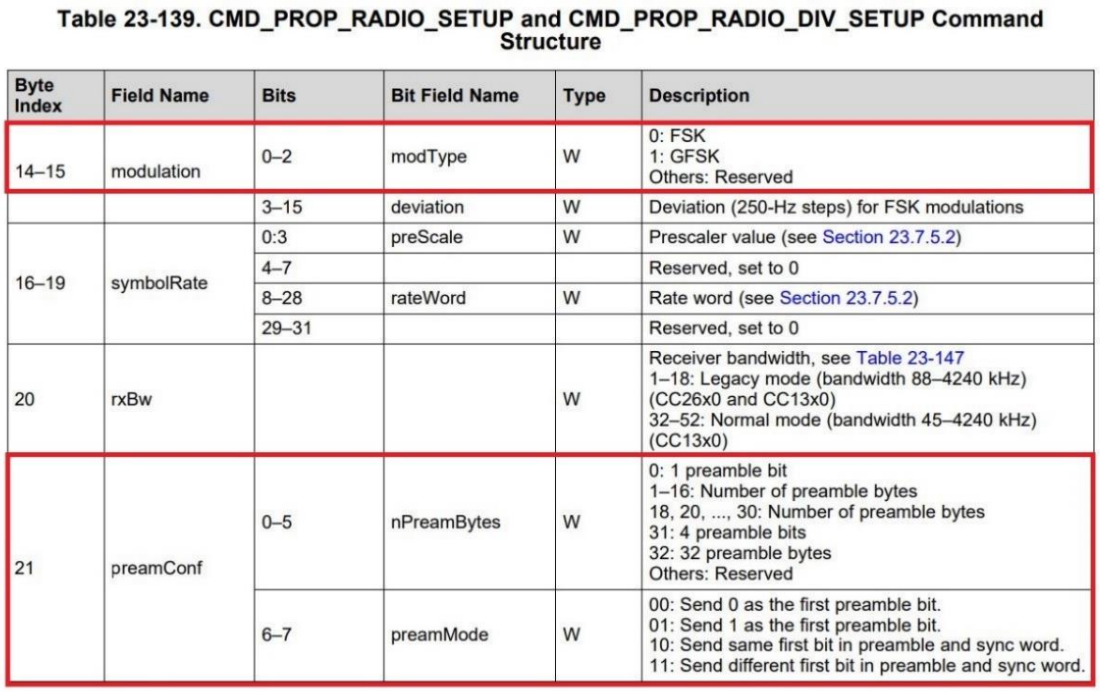
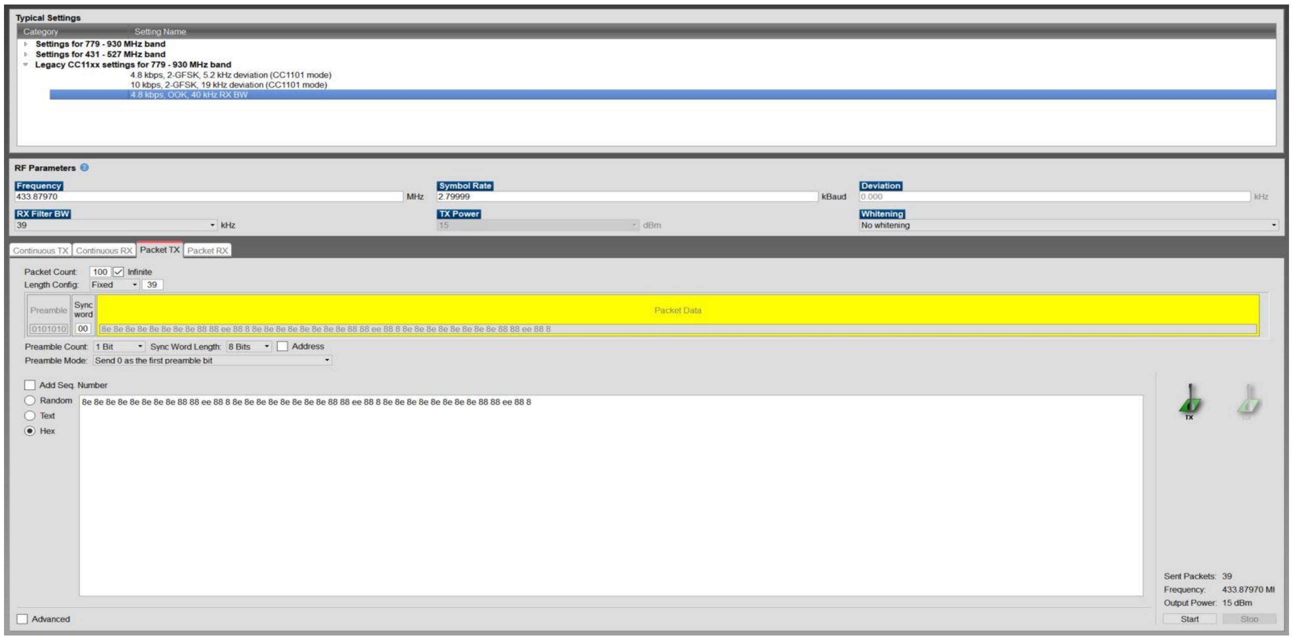


Table 23-139. CMD_PROP_RADIO_SETUP and CMD_PROP_RADIO_DIV_SETUP Command Structure

Byte Index	Field Name	Bits	Bit Field Name	Type	Description
14–15	modulation	0–2	modType	W	0: FSK 1: GFSK Others: Reserved
		3–15	deviation	W	Deviation (250-Hz steps) for FSK modulations
16–19	symbolRate	0:3	preScale	W	Prescaler value (see Section 23.7.5.2)
		4–7			Reserved, set to 0
		8–28	rateWord	W	Rate word (see Section 23.7.5.2)
		29–31			Reserved, set to 0
20	rxBw			W	Receiver bandwidth, see Table 23-147 1–18: Legacy mode (bandwidth 88–4240 kHz) (CC26x0 and CC13x0) 32–52: Normal mode (bandwidth 45–4240 kHz) (CC13x0)
21	preamConf	0–5	nPreamBytes	W	0: 1 preamble bit 1–16: Number of preamble bytes 18, 20, ..., 30: Number of preamble bytes 31: 4 preamble bits 32: 32 preamble bytes Others: Reserved
		6–7	preamMode	W	00: Send 0 as the first preamble bit. 01: Send 1 as the first preamble bit. 10: Send same first bit in preamble and sync word. 11: Send different first bit in preamble and sync word.

Figure 14

We set the SmartRF Studio to transmit in OOK modulation (under legacy settings), set the preamble bits to 0 and the syncword length to be 8 bits of zeros. We inserted the symbol rate and frequency according to what we calculated. We set the data packet to be the close button packet (Figure 15).



Typical Settings

Category	Setting Name
Settings for 779 - 930 MHz band	
Settings for 431 - 527 MHz band	
Legacy CC11xx settings for 779 - 930 MHz band	
	4.8 kbps, 2-GFSK, 5.2 kHz deviation (CC1101 mode)
	10 kbps, 2-GFSK, 19 kHz deviation (CC1101 mode)
	4.8 kbps, OOK, 40 kHz RX BW

RF Parameters

Frequency: 433.87970 MHz
Symbol Rate: 2.79999 kBaud
Deviation: 0.000 kHz
RX Filter BW: 39 kHz
TX Power: 15 dBm
Whitening: No whitening

Continuous TX | Continuous RX | Packet TX | Packet RX

Packet Count: 100 [✓ Infinite]
Length Config: Fixed 39

Preamble: 0101010
Sync word: 00
Packet Data: 8e 8e 8e 8e 8e 8e 8e 8e 88 88 ee 88 8 8e 8e 8e 8e 8e 8e 8e 8e 88 88 ee 88 8 8e 8e 8e 8e 8e 8e 8e 8e 88 88 ee 88 8

Preamble Count: 1 Bit
Sync Word Length: 8 Bits
Address: ☐
Preamble Mode: Send 0 as the first preamble bit

Add Seq. Number: ☐
Random / Text / [✓ Hex]

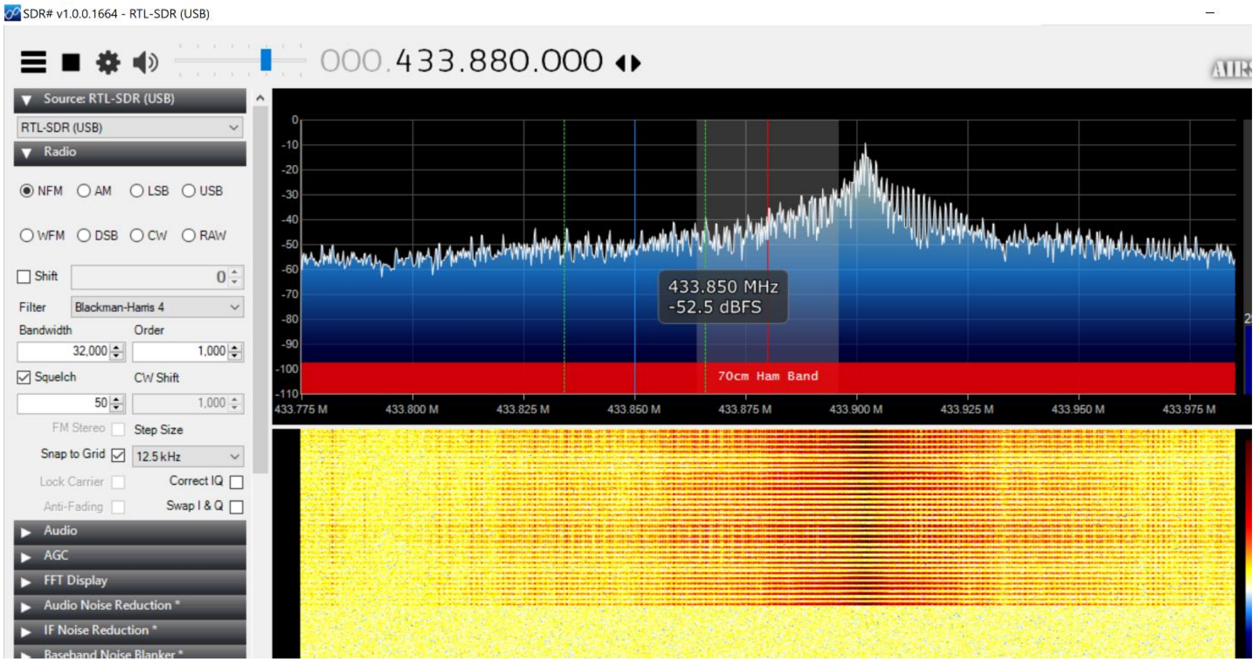
```
8e 8e 8e 8e 8e 8e 8e 8e 88 88 ee 88 8 8e 8e 8e 8e 8e 8e 8e 8e 88 88 ee 88 8 8e 8e 8e 8e 8e 8e 8e 8e 88 88 ee 88 8
```

Sent Packets: 39
Frequency: 433.87970 MI
Output Power: 15 dBm

Advanced: ☐

Figure 15

Then we started transmitting the data packet (close button) and the parking barrier control unit got the signal and turned on.


SDR# v1.0.0.1664 - RTL-SDR (USB)

000.433.880.000

Source: RTL-SDR (USB)

RTL-SDR (USB)

Radio

[✓ NFM] AM LSB USB

WFM DSB CW RAW

Shift 0

Filter Blackman-Harris 4

Bandwidth 32,000 Order 1,000

[✓ Squelch] 50 CW Shift 1,000

FM Stereo Step Size

[✓ Snap to Grid] 12.5 kHz

Lock Carrier Correct IQ

Anti-Fading Swap I & Q

433.850 MHz
-52.5 dBFS

Figure 16 – Transmitting through Launchpad the packet

We can see the packet through the sdrsharp tool (Figure 16).

Link to view us transmitting the packet successfully – click here.