Real Time Packet Reception Using Software Defined Radio's
Project Progress Report

Abbas Jhabuawala
Anant Semwal

March 8, 2017

1. Introduction

In this project, we are using a 433MHz RF transmitter and a RTL software defined radio to detect incoming wireless signals and decode the message contained in them in real time. We are using the Arduino micro controller to generate and encode the message and send it via the RF transmitter. At the receiver end we are using GNU radio SDR tool set to demodulate, decode and reconstruct the data transferred.

2. Progress

1. Understanding the code example for RFID decoding given by you
2. Study about the modulation type of the transmitter we have
3. Study about the messaging format and the encoding used
4. Started with getting acquainted with the SDR and SDR# using POSCAG decoding example given on the RTL SDR website
5. Using SDR# to record the incoming signal and to an audio file and analyzing the recorded data using Audacity
6. Experimenting with the use of GNU radio software and SDR

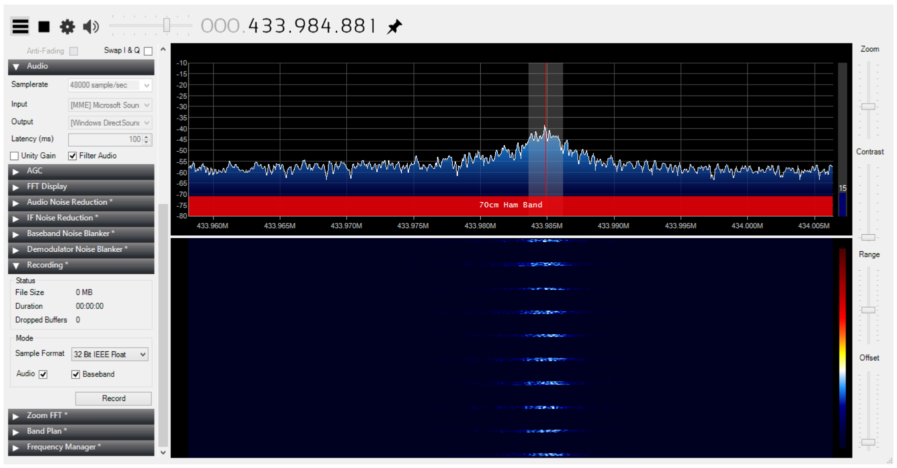
3. Challenges/Problem

1. Too much noise on the receiver and therefore reconstructing the signal a little more challenging
2. Analog to Digital Conversion in the GNU Radio to sample the incoming data and to verify the message format and details with the previously identified data
3. Weak Power of the transmitter.

4. Results

The message format used for transmission is defined in the Arduino library $RH_ASK.h$. It is as follows:

1. 36 bit training preamble
2. 4 bit header (To, From, ID, Flag)
3. 12 bit start symbol ($0xb38$)
4. 8 bit message length count including frame count sequence


000.433.984.881

Samplerate 48000 sample/sec

Input [MME] Microsoft Soun

Output [Windows DirectSounc

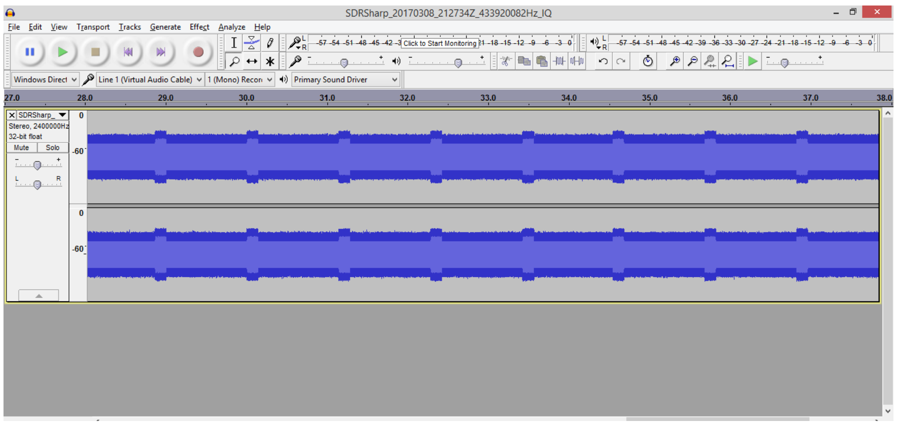
Latency (ms) 100

Unity Gain ☐ Filter Audio ✓

Status
File Size 0 MB
Duration 00:00:00
Dropped Buffers 0

Mode
Sample Format 32 Bit IEEE Float
Audio ✓ Baseband ✓

Figure 1: Frequency Domain Representation of Signal Captured by RTL-SDR in SDR#


SDRSharp_20170308_212734Z_433920082Hz_IQ

SDRSharp_ Stereo, 2400000Hz 32-bit float

Figure 2: Time Domain Representation of Signal Captured by RTL-SDR in Audacity

5. N message bytes – The maximum n is defined in the header file also

The speed of data transfer is selected at 200 bits per second

5. Next Step

1. Figure out the message encoding scheme and implementing it on the receiver side in order to decode the message
2. Reconstructing the signal using A2D converter to the original text in real time and verifying that noise / interference has not caused error in transmission.

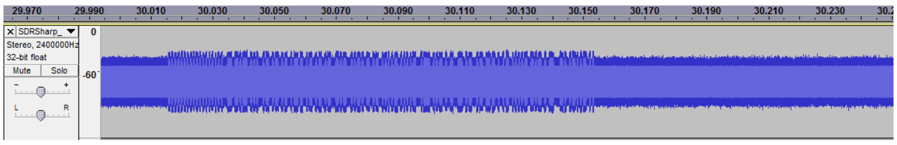


Figure 3: A single message frame representation viewed in Audacity

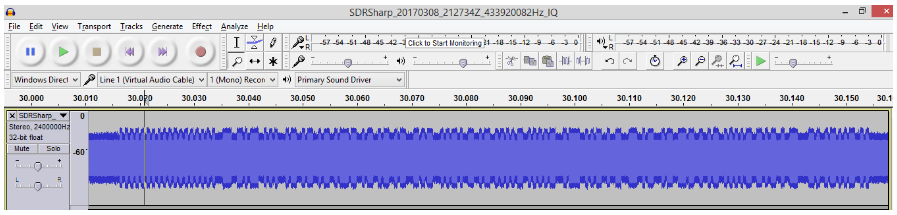


Figure 4: Each packet of the message shown clearly

6. Further Reference

All the source code,referenced documents,images and recorded audio files are available on GitHub.