Linnæus University
Sweden

Master Thesis

Electrical Engineering, Specialisation in signal processing &Wave propagation

FMCW radar implemented with GNU Radio Companion

Author:Zhu Qizhao and Wang Yaqi
Supervisor: Sven-Erik Sandström
Examiner: Sven-Erik Sandström
Date: 2016/5/19
Subject:Electrical Engineering
Level: Advanced
Course code: 5ED36E

Contents

Abstract

Continuous-wave frequency-modulated radar, or FMCW radar, is simple in design, small in size and weight and uses low transmitting power. The range resolution depends on the bandwidth. FMCW radar is used in applications ranging from guided weapons systems to vehicle collision avoidance systems. Measuring the distance to the target is the essential feature of FMCW radar. Firstly, this thesis introduces the basic structure of the FMCW radar and the principle for measuring distance. Secondly, by using software-defined radio (SDR), FMCW radar can be implemented and configured with a reduced cost and complexity. In this report, the radar is implemented by means of the software GNU Radio Companion with a test signal. HackRF may be used in future work with an osmocom source instead of the test signal.

Keywords : FMCW radar, software defined radio, GNU Radio Companion

1 Introduction

FMCW radar is a range measuring radar. In order to determine the distance, a continuous carrier signal is modulated and transmitted. Signals, or echoes [1], that are reflected from the targets are received, and the frequency difference between the transmitted signal and received signal is obtained. This difference corresponds to a time delay and to the distance. In other words, the frequency of the beat signal, that is obtained by means of multiplication, is proportional to the distance.

GNU Radio is a free software that was first published in 2001 by Eric Blossom. This software provides a framework for signal processing based on blocks and flow graphs and makes simulation easy. GNU Radio could be used with external hardware like HackRF. If there is no external hardware, GNU Radio can be used for simulation, and has facilities for presentation of the results. In GNU Radio Companion (GRC), the signal processing blocks are connected in a flow graph. The underlying coding is made in Python and C++. Nowadays, GNU Radio is widely used in wireless communication and radio systems.

This report describes how FMCW radar is implemented with GRC by using internal simulation rather than the osmocom source (Open Source Mobile Communication) and the hardware HackRF.

2 Theory

There are two kinds of FMCW radar, LFM (linear frequency modulation) and NLFM (nonlinear frequency modulation). NLFM is easy to implement, using for example sinusoidal modulation, but the beat frequency of each object is not unique. Objects at different distances can not be distinguished, so NLFM is only fit for single object situations. With LFM the problem does not occur since the beat frequency is unique. Saw tooth or triangle wave can be used for modulation and in this report saw tooth is considered.

2.1 The principle of FMCW radar

A typical block diagram for FMCW is shown in Figure 1 [2]. FMCW radar transmits a continuous carrier, which is modulated by a periodic function. The chirp generator is used as the signal source in the transmitter. It can also be used for mixing if the radar is monostatic. A voltage controlled oscillator (VCO) produces the chirped carrier that results from the analog input. The input is generated by a D/A converter. Both a transmitter and a receiver antenna are shown here and they are connected to HF front ends. The chirped carrier passes the transmitter front-end and on to the antenna.

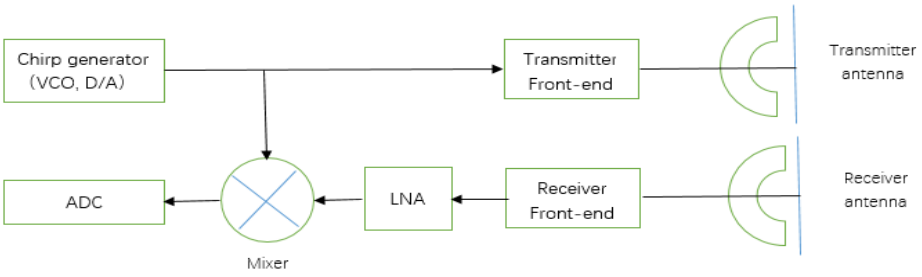

Figure 1: A simple FMCW radar.

The received signal is amplified (LNA) and mixed with the reference signal in order to extract the beat frequency. The range of the target is proportional to the beat frequency.

2.2 Distance measurement using saw tooth modulation

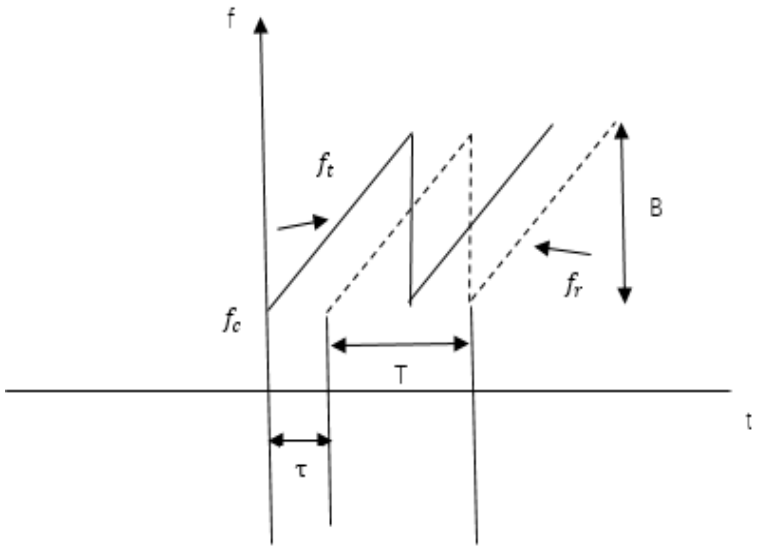

Figure 2: A transmitted (solid) and a delayed (dashed) saw tooth function.

Assume that the modulating signal is an ideal saw tooth wave. If Doppler frequency shifts are ignored, the modulating and the demodulated signals would correspond to Figure 2. The solid line shows the transmitted saw tooth and the dashed line shows a saw tooth that corresponds to the received signal. The frequency of the chirped signal is given by,

$$f(t) = f_c + B\frac{t}{T}, \tag{1}$$

where f_c is the carrier frequency, B is the bandwidth of the signal and T is the chirp period of the saw tooth [2]. The delay τ of the received signal,

$$\tau = \frac{2R}{c}, \tag{2}$$

relates to the distance R from the radar, and the speed of light $c = 2.9979 \times 10^8$. The range resolution is given by,

$$\Delta R = \frac{c}{2B}, \tag{3}$$

and the range can be obtained from the beat frequency f_B as,

$$R = \frac{Tc}{2B} f_B. \tag{4}$$

The beat frequency is the frequency difference between the transmitted and received wave [3].

3 Simulations

In this chapter, the software GNU Radio Companion and the process of installation will be described. The FMCW radar is then implemented and simulated with GRC version 3.7.8.

3.1 The graphical interface GNU Radio Companion

Gnu Radio Companion (GRC) is a graphical tool for creating signal flow graphs and the corresponding Python code. The features of GRC are:

1. GRC is bundled with the GNU Radio source. If all dependencies are met, GRC will be installed with GNU Radio.

2. GRC builds the flow graph and generates the python source for the flow graph in the updated QT format.

3. The variable blocks define variables as numerical values or as functions of other variables, cf. Fig. 3.

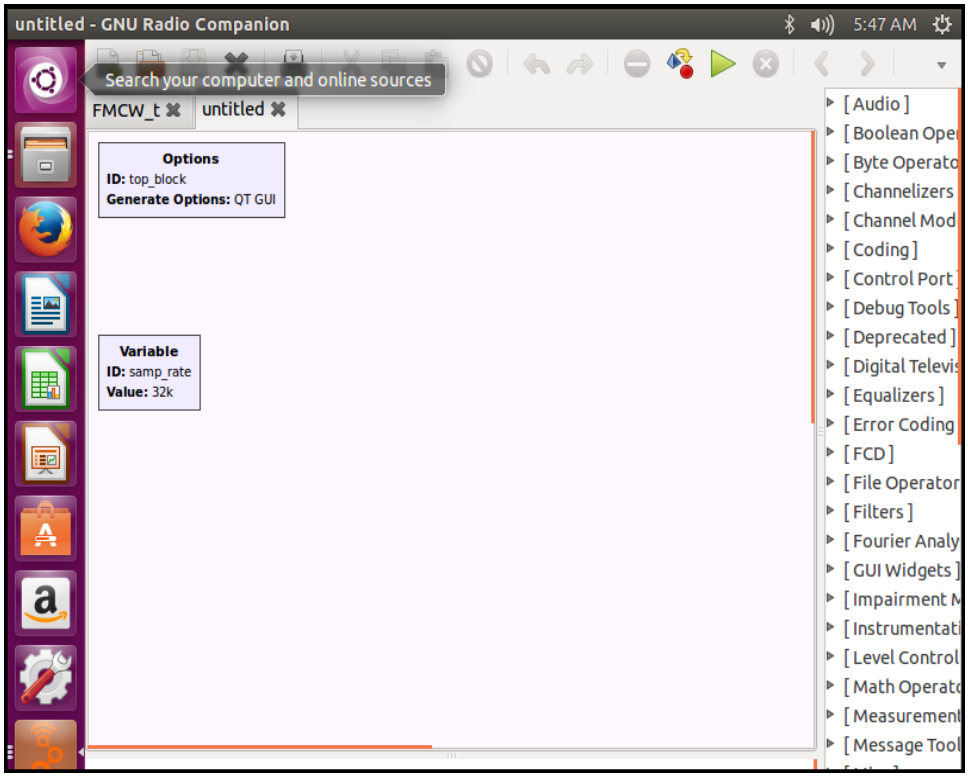

Figure 3: The graphical interface of the GRC in Linux.

3.2 Installation of the GRC

The installation is made with commands and a terminal window is needed (see Appendix A).

3.3 Implementing a radar system in GRC

As shown in Figure 4, the flow graph consists of several blocks. A ramp function is generated with a saw tooth signal. This signal modulates a VCO that produces a chirped complex (quadrature) signal. The throttle block makes it possible to use the processor efficiently. For the FFT, 2^n samples are taken on the ramp. There should be some margin so that both the transmitted and the delayed signal are on the same ramp. The sampling points along the ramp are reused so that time integration can be used to reduce noise. The delay block corresponds to a target at a specific range.

Finally, the received signal is mixed with the transmitted signal, using a multiply/conjugate block, and the result is saved into a binary file.

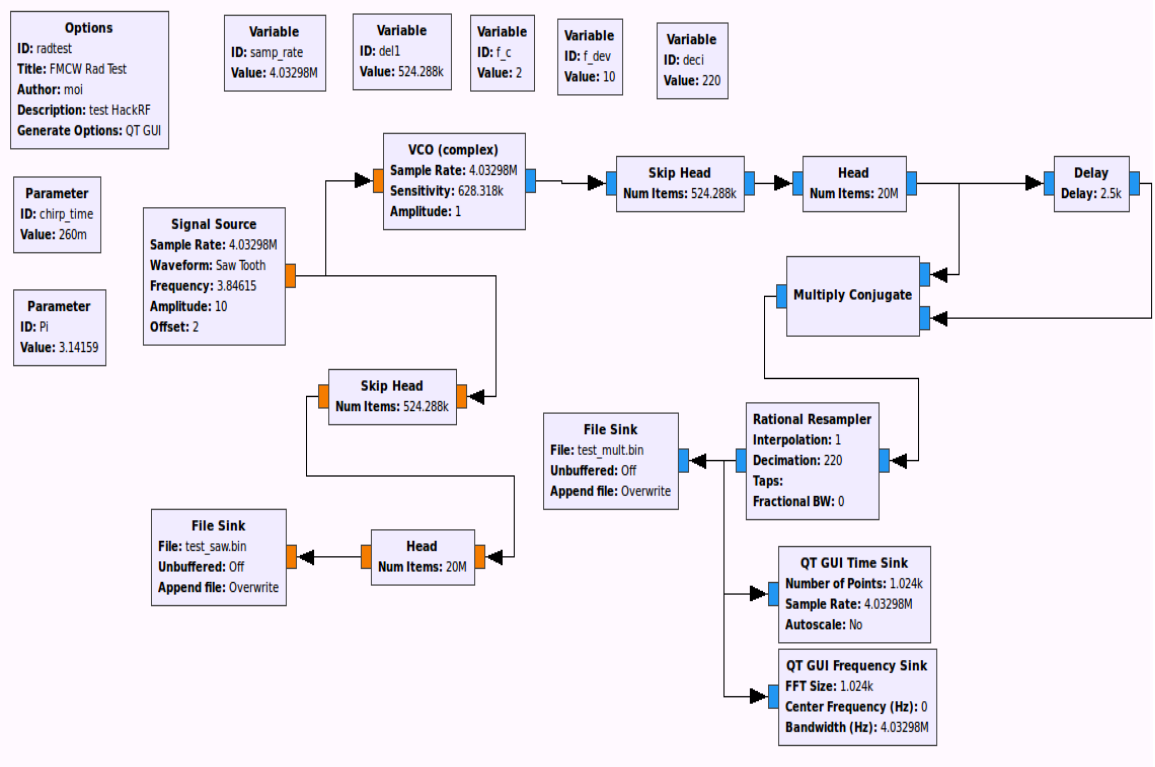

Figure 4: GRC flow graph for an FMCW radar with one simulated target.

3.4 Analysis of simulations

From the simulation it is clear that the FMCW radar can detect targets in the form of beat frequencies. Figure 5 shows the beat signal in the time domain for a single target. Figure 6 shows the same signal in the frequency domain, as given by the FFT in GRC. The single peak is the beat frequency that results from the mixing. It is determined by the range of the target.

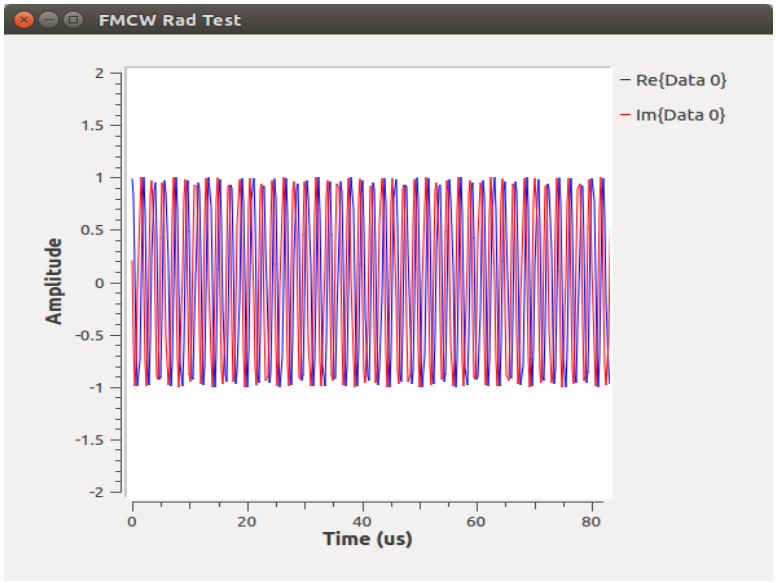

Figure 5: Beat signal for a single target.

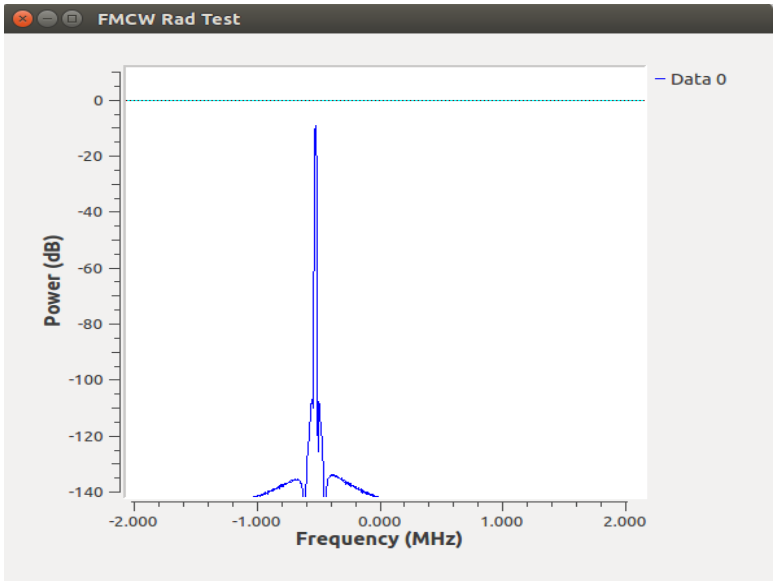

Figure 6: Beat frequency spectrum for a single target.

For three targets (Figure 7) there is a more complicated beat signal. Figure 8 shows the beat signal for three targets that are delayed by 300, 1500 or 2500 samples. Figure 9 presents the spectrum that is needed to obtain the range information on these three targets. For simplicity, the three targets have the same amplitude. Damping and radar cross section are not included in the model.

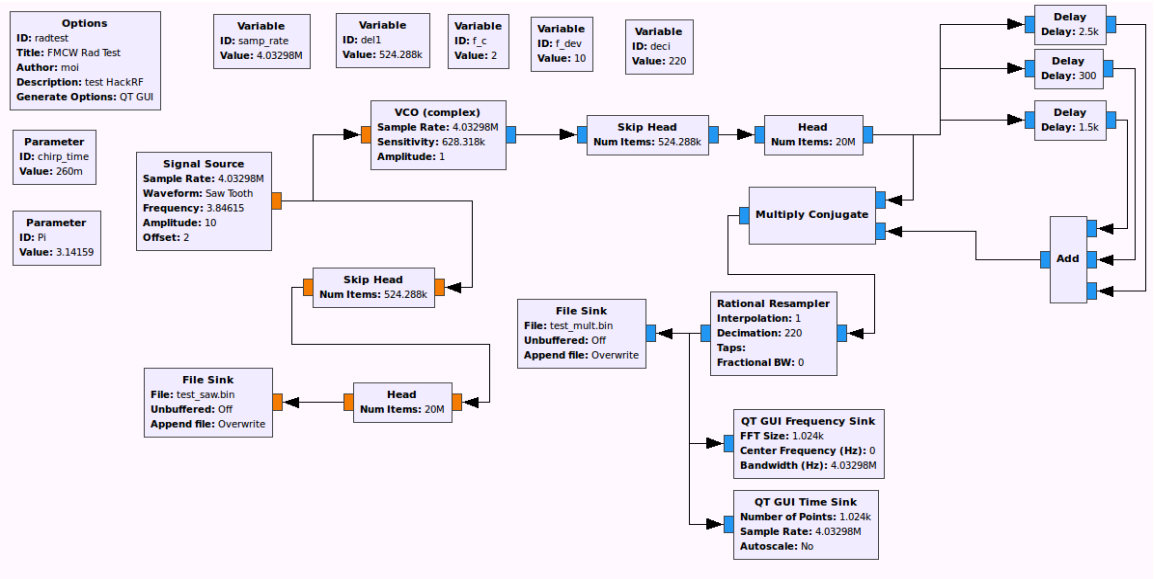

Figure 7: Flow graph for three targets.

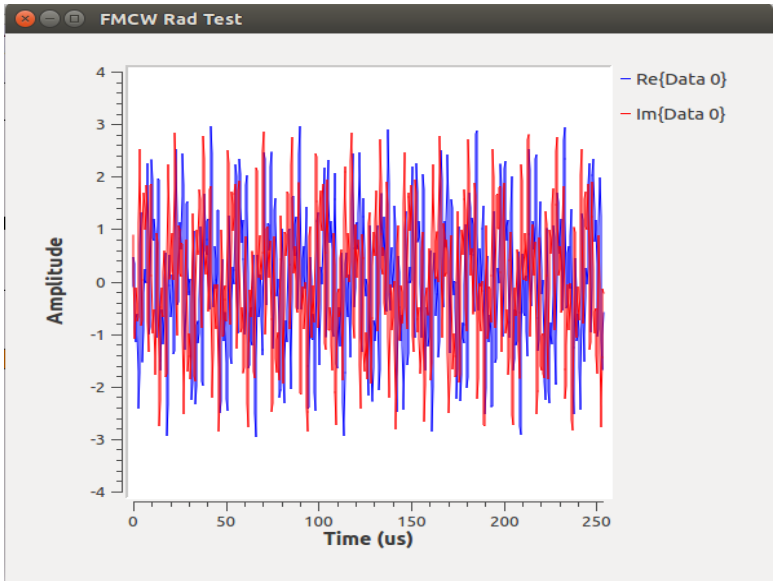

Figure 8: Beat signal for three targets.

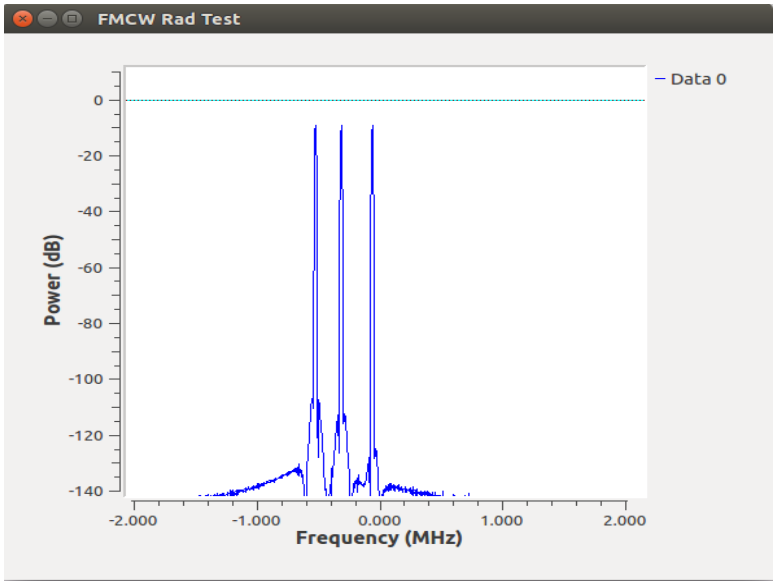

Figure 9: Spectrum for three targets.

The parameters are shown in Table 1.

Parameters	Values
Chirp time	260 ms
Chirp waveform	Saw tooth
Sampling rate	4.03298 MHz
Decimation	220
Sensitivity	628.318 KHz/V
Carrier frequency	0.2 MHz
Bandwidth	1 MHz
Delay1	300 samples
Delay2	1500 samples
Delay3	2000 samples

Table 1: FMCW radar design specification.

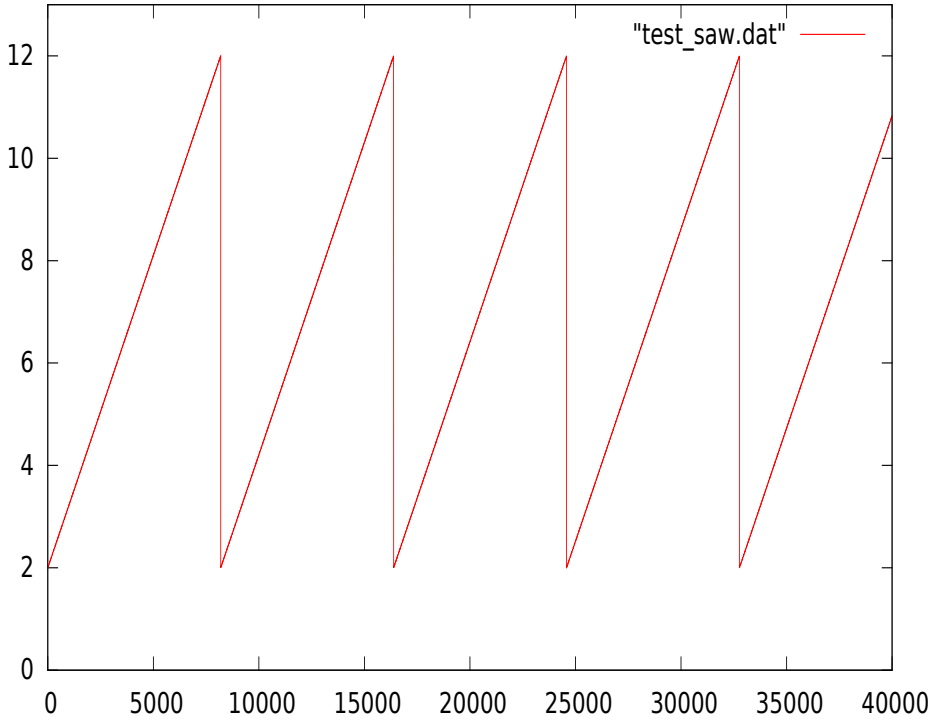

Figure 10: The exported version of the saw tooth function.

The bandwidth of the bandwidth of the FM signal is determined by the amplitude of the saw tooth and the sensitivity of the VCO. Decimation is needed to reduce the amount of output data and to simplify presentation.

As shown in Figure 7, the data is sent to the binary files test_saw.bin and test_mult.bin. The binary file is converted to a file with real numbers by means of Linux commands. The script for this is listed in Appendix B and the result of the conversion is shown as Figure 10.

4 Conclusion

GRC is used in this study since the simulation can be done with no cost and little complexity.

The simulations show that reasonable results can be obtained when the unmatched internal signal processing is used. Although the sample rate is satisfied with Nyquist theorem, the sample rate and sensitivity have to be adjusted to a higher value, the decimation is 220 which is not satisfied with ideal decimation value in 2^n. The FMCW prototype can separate targets in a systematic fashion.

In future work one could improve the signal processing by taking samples systematically rather than relying on the GRC features. One could use the exported saw tooth for synchronisation and the FFT [4] could be combined with automatic identification of the peaks.

A realistic measurement, rather than a simulation, could be achieved by connecting HackRF units. This requires more synchronisation [5] since HackRF does not have full duplex. In the case of multistatic radar, the synchronisation problem has to be solved even if hardware with full duplex is available. The internal clocks of the computers need to be synchronised (NTP) and commands must be transferred between the computers (PSSH). One could use WLAN and GPS receivers for this. Calibration with the known distance between the transmitter and the receiver is also needed.

References

[1] W. Melvin and J. Scheer, *Principles of modern radar*. The Institution of Engineering and Technology, SciTech publishing, 2012.

[2] S. Sundaresan, C. Anjana, T. Zacharia, and R. Gandhiraj, "Real time implementation of FMCW radar for target detection using GNU radio and USRP," in: Communications and Signal Processing ICCSP, Chengdu, pp. 1530–1534, 2015.

[3] Q. Wenbao, "The research and implementation of an FMCW SAR," Master's thesis, North University of China, Shan Xi, China, 2013.

[4] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes*. Cambridge University press, 1992.

[5] G. Aloi, V. Loscri, A. Borgia, E. Natalizio, S. Costanz, P. Pace, G. D. Massa, and F. Spadafora, "Software defined radar: synchronization issues and practical implementation," in: 4th International Conference on Cognitive Radio and Advanced Spectrum Management, Article No. 48 ACM, New York, 2011.

Appendices

A Installation of GRC

```
sudo apt-get install git
apt-get install build essential

sudo apt-get install cmake to install any missing components and re-run configure.

sudo apt-get install libusb-1.0-0-dev
sudo apt-get install libboost-dev
sudo apt-get install liblog4cpp5-dev
sudo apt-get install libboost-system-dev
sudo apt-get install libboost-thread-dev
sudo apt-get install libboost-program-options-dev

sudo apt-get install swig to install extra packages swig 2.0, swig-doc and swig 2.

next, clone the link by entering:

git clone https// github.com/mossmann/harckrf.git

cd hackrf
cd host
mkdir build
cd build

cmake ../ -DINSTALL UDEV RULES=ON

make
sudo make install
sudo ldconfig

hackrf info
```

```
cd~

sudo apt-get install gnuradio


sudo apt-get install gnuradio-dev
sudo apt-get install gr-iqbal

git clone git: //git.osmocom.org/gr-osmosdr

cd gr-osmosdr

git checkout 58d95651
cd build
cmake
make
sudo make install
sudo ldconfig
```

B Linux command for test the saw tooth signal

```
truncate -s 160000 test_saw.bin
hexdump -v -e '1/4 "%f "' -e '"\n"' < test_saw.bin > test_saw.dat
sed -i '$ d' test_saw.dat
sed -i 's/,/./g' test_saw.dat
gnuplot -e "file_saw='test_saw.dat'" test_saw.plg

plot file_saw with lines
pause -1

gnuplot
set yrange [0:13]
plot "test_saw.dat" with lines 1 linetype 1
set term pdf
set output "saw.pdf"
replot
exit
```