GPS Signal Spoofing

EE691: R&D Project

Prasann Viswanathan Iyer | 190070047
under the supervision of
Prof. Sibi Raj B Pillai

Apr 30, 2023

Overview

The Global Positioning System (GPS) has evolved into a vital tool for many facets of our everyday life, such as time synchronization, tracking, and navigation. However, spoofing attacks, where malicious actors send false GPS signals to trick receivers, are possible against GPS technology. From minor disruptions to critical infrastructure failures, such attacks may have serious repercussions. However, it is not extremely straight forward to execute this GPS spoofing. The aim of my Research and Development Project under Prof. Sibi Raj B Pillai is to try different methods to conduct GPS signal spoofing for which I have learnt about GPS signal structure, GNSS-SDR: an open-source software-defined Global Navigation Satellite Systems (GNSS) receiver that allows researchers and engineers to process GNSS signals using standard personal computers and inexpensive hardware and tested out possible methods to spoof the GPS signals using the software and a USRP N210 Software Defined Radio.

1 GNSS-SDR

GNSS-SDR is designed to process signals from a wide range of Global Navigation Satellite Systems constellations, including GPS, Galileo, BeiDou, and GLONASS. Here SDR stands for Software Defined Receiver, instead of the standard radio. It is built using the GNU Radio software development kit and is distributed under an open-source license. The software is designed to run on Linux and Windows operating systems and supports a wide range of hardware platforms, including software-defined radios (SDRs), general-purpose processors, and graphics processing units (GPUs). We still need a radio frequency front-end that down-converts signals to a lower frequency, making some filtering and amplification in the process, and sampling them at a certain rate, delivering a stream of quantized, digital raw samples to the computing platform. For this purpose, we use a USRP N210 Software Defined Radio device. The core functionality of GNSS-SDR includes signal acquisition, tracking, and navigation data decoding. On installing GNSS-SDR and setting up all its dependencies, we moved on to:

1.1 First Position Fix

To gain hands on experience with working with GNSS-SDR we worked with a pre-existing signals file available on the internet. This approach does not call for the availability of a radio frequency front-end or a powerful computer running the software receiver because the signal source is a file comprising raw signal samples. The signal source file can be found here.
It contains 100 seconds of raw GNSS signal samples collected by an RF front-end centered at 1575.42 MHz, that was delivering baseband samples at 4 MS/s, in an interleaved IQ 16-bit integer format. This data is acted upon by GNSS-SDR to get our first position fix; for which we set up a configuration file where we define the raw data file location and the configurations for the signal source, signal conditioner, channels, acquisition, tracking, telemetry and other elements. This is then run with the command `gnss-sdr --config_file=file_name.conf`
On successful execution, this creates .geojson, .kml and .gpx files. A .geojson file is a format for storing geospatial data in JSON format, commonly used for exchanging geographic data between different applications and systems. A .kml file is a format for storing geospatial data used by Google Earth to display various forms of information on a map, and a .gpx file is a format for storing GPS data that contains information such as waypoints, tracks, and routes that can be used by mapping and navigation software. These together give us our resulting position fix.

1.2 Tracking

The role of a Tracking block is to follow the evolution of the signal synchronization parameters: code phase $\tau(t)$, Doppler shift $f_D(t)$ and carrier phase $\phi(t)$. For our projects purpose we wish to track signals on the L1 channel, where $f_{GPSL1} = 1575.42MHz$. We do this with the Implementation:

`GPS_L1_CA_DLL_PLL_Tracking` where we can set other parameters as well. The point to note, however, is that this tracking can be stored in dump files of .dat or .mat signal types. Here we get tracking dumps equal to the number of distinct channels we have defined in the config file. Each Channel encapsulates blocks for signal acquisition, tracking, and demodulation of the navigation message for a single satellite. These abstract interfaces can be populated with different algorithms addressing any suitable GNSS signal.

2 GNSS Signal Structure

GNSS Signals have basically three types of signals

- Carrier Signal
- PRN code
- Navigation Data

These GNSS signals are based on CDMA, where CDMA (Code Division Multiple Access) is a multiple access technique that allows multiple signals to occupy the same frequency band by assigning unique codes to each satellite signal. Below depicts a schematic explaining the construction of an L1 band GPS signal using CDMA: PRN Code is a sequence of randomly distributed bits that is one

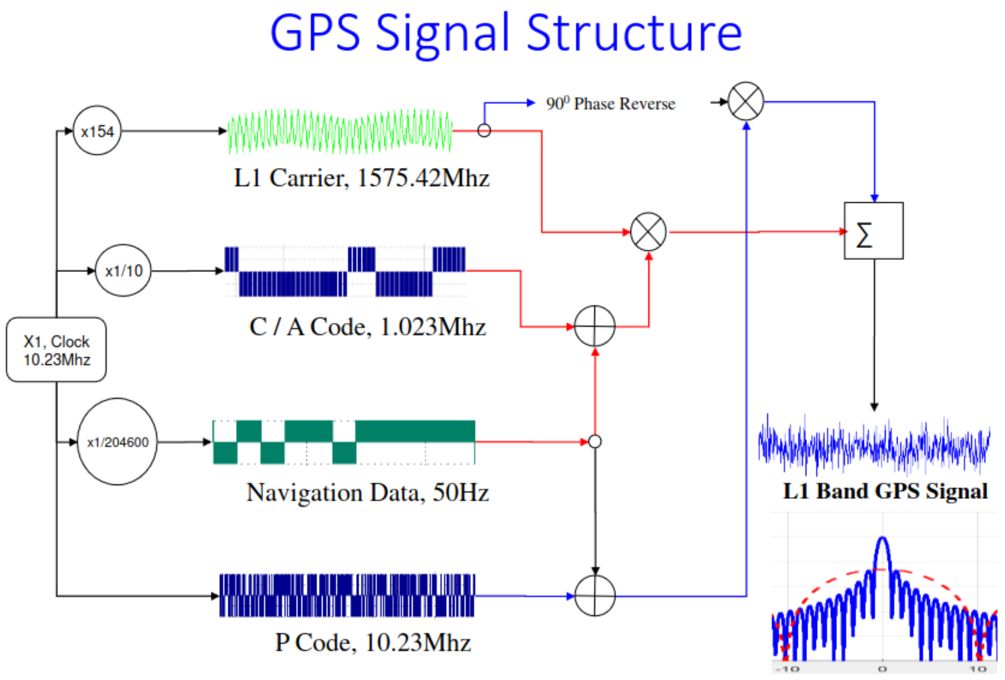


Figure 1: L1 Band GPS Signal Construction using CDMA

millisecond long. It regenerates via the Gold Code technique. The number of bits in a millisecond is 1023. A distinct PRN Code is transmitted by each GPS satellite. Satellites are recognised by

their specific PRN code or ID by a GPS receiver. It is used to measure signal transit time and is continuously repeated every millisecond. Where the PRN code ended or repeated can be determined by the receiver. Here the difficulties in spoofing are created by the fact that Maximum Cross-correlation Value is -23dB and if any signal above this power enters a GPS receiver, it will totally block all GPS signals. This effectively means increasing gain of spoofing signals cannot be a solution as it forbids signals to pass. Also, if a longer PRN code is used, receiver becomes more resistant to Jamming signal.
Navigation Data or Message is a continuous stream of digital data transmitted at 50 bit per second. Each satellite broadcasts its own highly accurate orbit and clock correction and approximate orbital correction for all other satellites as well as system health, etc.

3 Methods Tested

In this section we discuss the procedure of some different techniques tried to spoof GPS signals. The results of simulations of these methods shall be discussed in the next section.

3.1 Random PRN Generation With Fixed Navigation Data

For this method, a fixed navigation data stream of 1500 bits (continuously repeated) is modulo 2 added (equivalently XoR-ed) to a C/A code generated from another PRN number satellite using the Gold Codes algorithm in GNU radio. This signal is transmitted and received by the two USRP N210 devices to see if they result in spoofing of the GPS system.
To generate the PRN C/A code here, we look at the Gold Codes algorithm which can be read succinctly here. We use a python code to generate the C/A stream for a given satellite PRN input.

```python
def PRN(sv):
    """Build the CA code (PRN) for a given satellite ID
    :param int sv: satellite code (1-32)
    :returns list: ca code for chosen satellite
    """
    # init registers
    G1 = [1 for i in range(10)]
    G2 = [1 for i in range(10)]
    ca = []
    # create sequence
    for i in xrange(1023):
        g1 = shift(G1, [3,10], [10])
        g2 = shift(G2, [2,3,6,8,9,10], SV[sv]) # <- sat chosen here from table
        ca.append((g1 + g2) % 2)
    return ca
```

Listing 1: C/A Code Generation

This is continuously added modulo 2 to the 1500 bit navigation data stream and sent as the signal. We use GNU Radio for this. Given below is the flowgraph we have used. We used the above

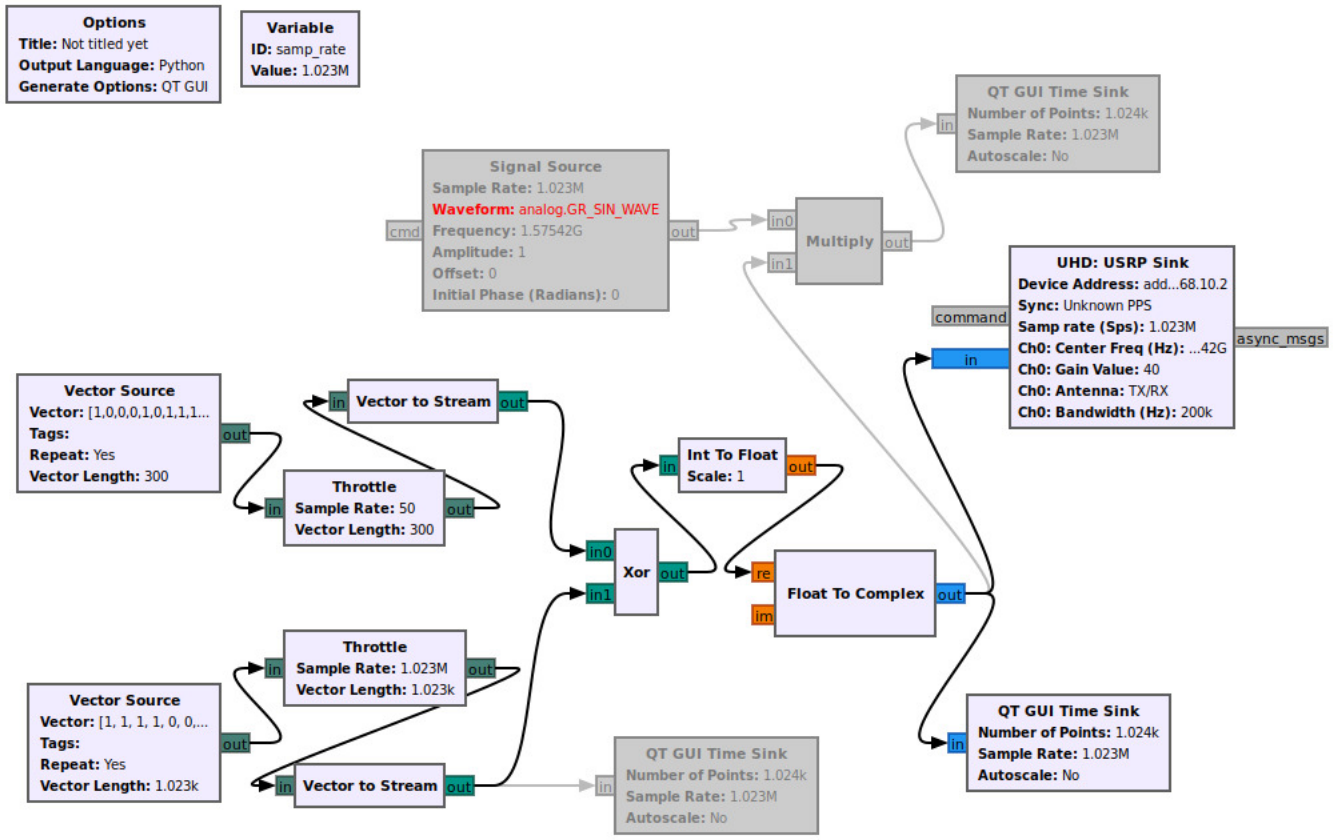


Figure 2: GNU Radio Flow Graph to Generate CDMA

schematic for construction of an L1 signal as our reference. The PRN vector source and navigation data vector source have been directly copied and pasted. The rates correspond to the frequencies of the signals. Therefore we have a rate of 50 bps for the navigation data stream and 1023 bps for the CA code.

3.2 .mat Data Manipulation

In this method I tried to resend manipulated signal data that we acquired from the first position fix signal file. When you are working with python there are multiple ways to read and write to .mat files but the way that works best (without errors/dependency requirements) is described below:

```python
import mat73
import scipy.io as sio
import h5py
import hdf5storage

mat1 = mat73.loadmat('tracking_ch_1.mat')
mat1['PRN'][:] = 17 #new_val
```

```
hdf5storage.write(mat1, '.', 'test.mat', matlab_compatible=True)
```

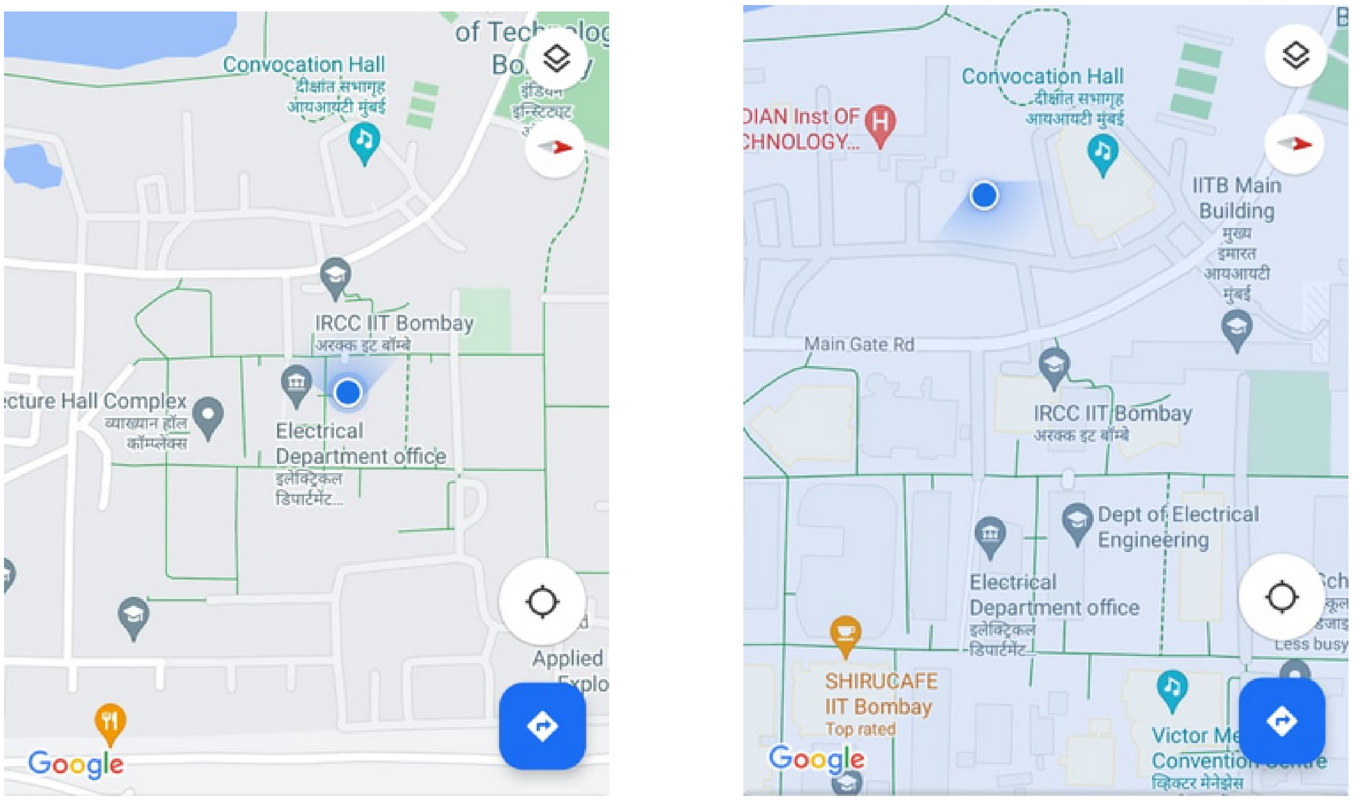
Listing 2: Signal Data Manipulation

However the issue with this approach is that it is not considering all the channels data. Each individual dump file (for a channel) contains very little tracking data so changing the PRN value for this small file doesn't result in a long enough signal source for GPS spoofing. Hence the results of spoofing obtained here are poor.

4 Results

4.1 Experiment 1

On running the Flowgraphs for experiment 1 (Fakely Generated CDMA), we observed that position fixes were not being received of latitude and longitude by GNSS-SDR. This indicated some level of successful jamming. The signals cause certain deflection from actual position on google maps, which aren't major but aren't insignificant either. The results of which are shown below:



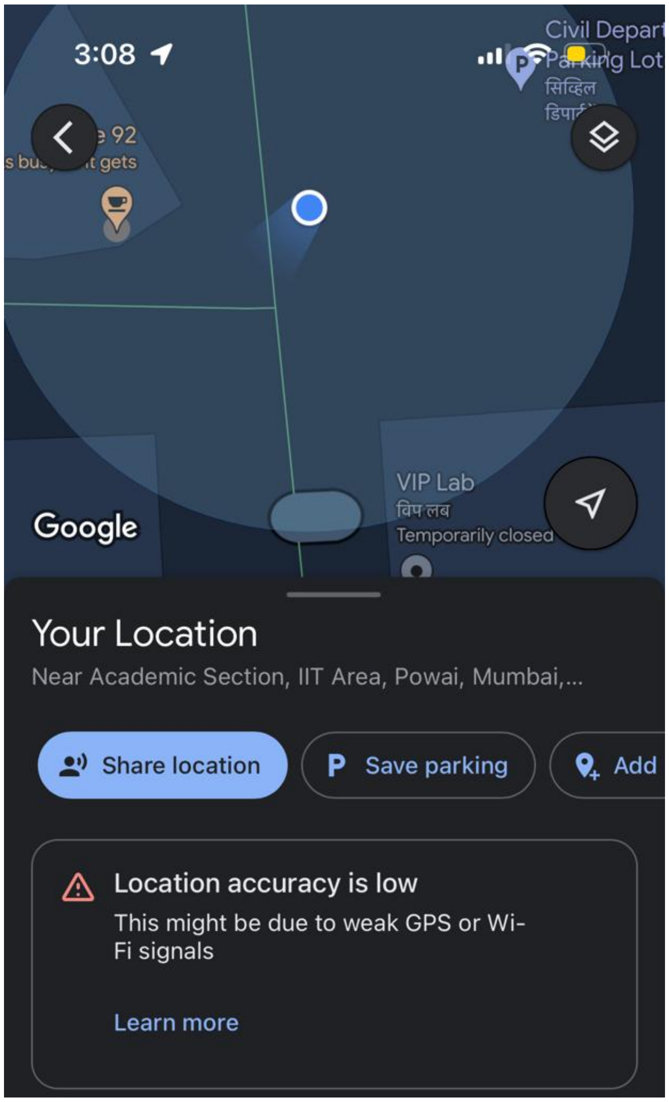
Actual Position

Displaced Position

Figure 3: Effect of Experiment 1 on Google Maps

4.2 Experiment 2

On running the modified .mat file (with changed PRN number) the results were not as satisfying. The signal acts as a basic random stream jammer and seems to be equivalent in functioning to any bitstream on the same frequency. Nevertheless, the experiment showed some displacement on google maps and the app showed low location accuracy. This experiment can at best be considered a failed jamming attempt. This was my first idea on how to possibly send wrong signals to cause spoofing and hence needs additional work.



Your Location

Near Academic Section, IIT Area, Powai, Mumbai,...

Location accuracy is low

This might be due to weak GPS or Wi-Fi signals

Learn more

Figure 4: Google Maps on Signals from Changed PRN Signal Data

5 Conclusion and Future Work

In this project I researched methods to conduct GPS Signal Spoofing using GNSS-SDR and the USRP N210 Software Defined Radio. I have tried two methods but there are many more possibilities to explore, and different bits to try to manipulate to break the GPS System. I learnt a lot about GPS Signal structure and the security measures GNSS Signals have to prevent jamming and spoofing.
In the future work it may be worth while to explore the package at this link. It provides vast functionality in creation of GPS Signals but is cumbersome with respect to number of parameters and needs sufficient time to be researched.