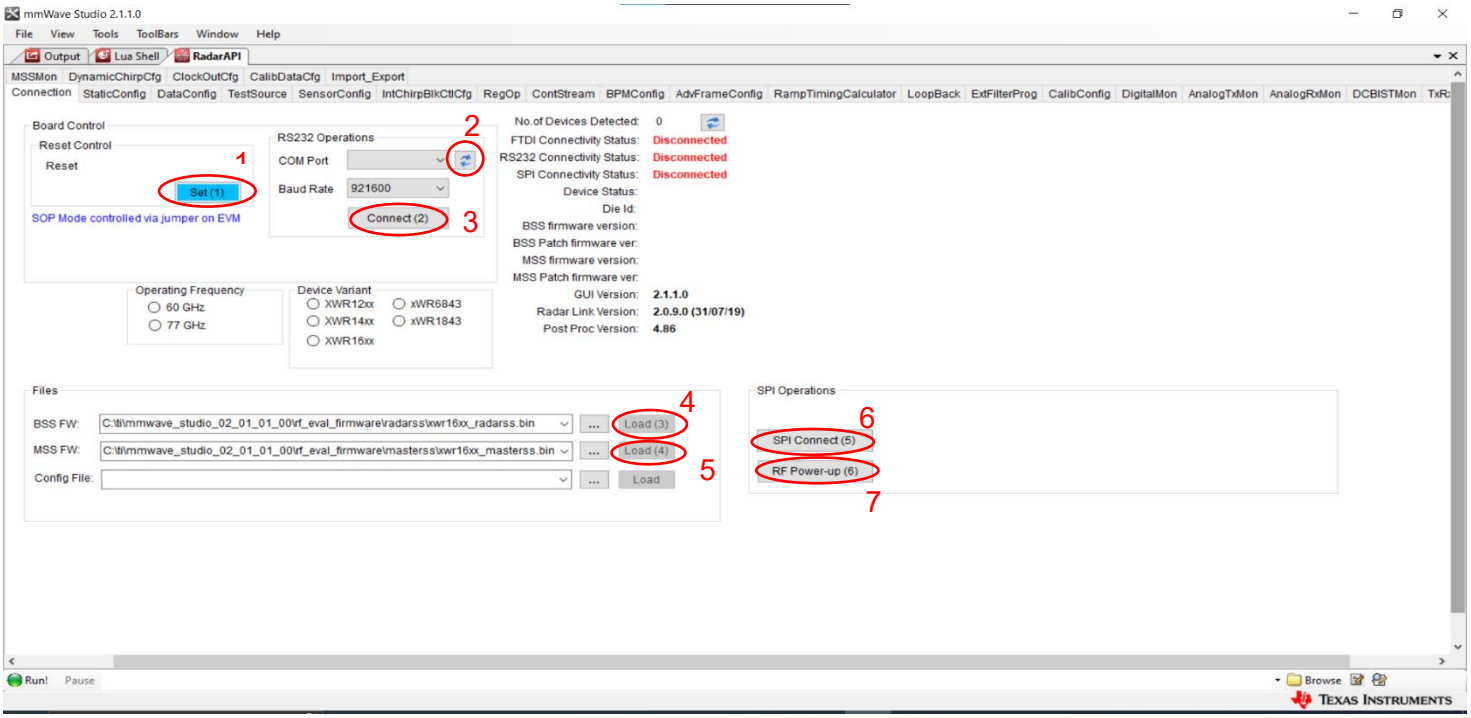
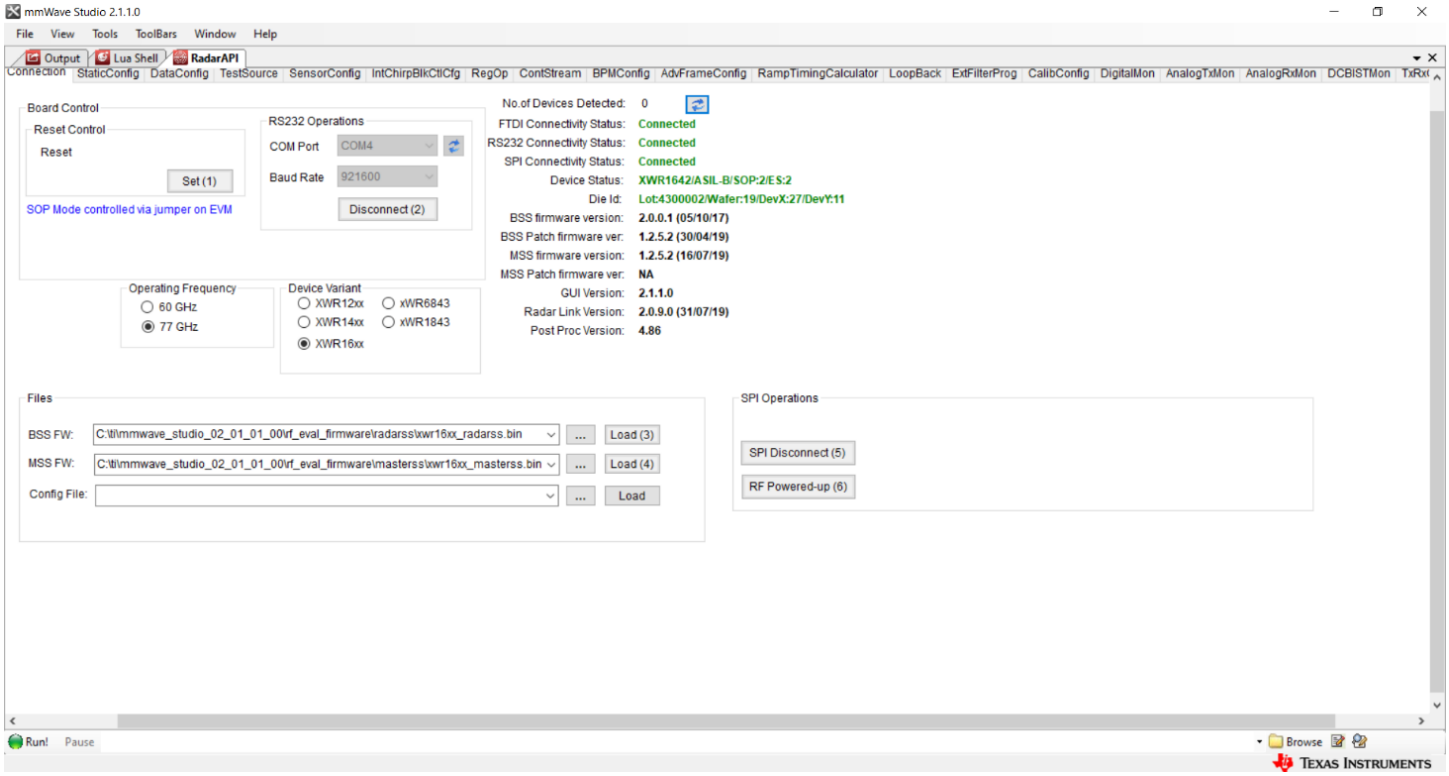
How to set up Radar

1. Download mmWave Studio, MATLAB Runtime Engine, and Microsoft Visual C++, and set up the radar as shown.
2. Power on the radar
3. Open mmWave Studio
4. Go to ‘Connection’ under RadarAPI. To setup the connection follow the image below. Ensure the BSS FW and MSS FW file is selected.

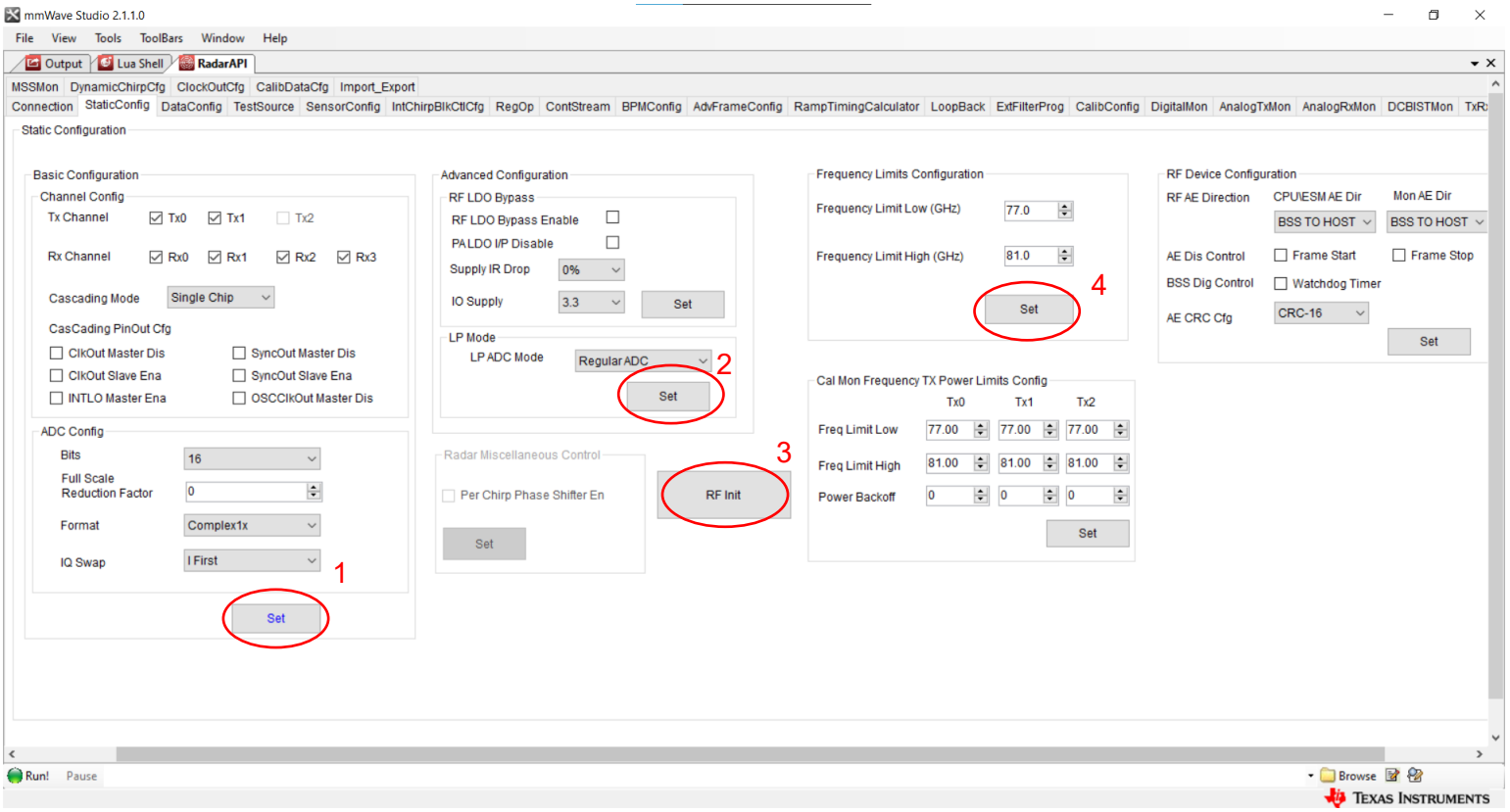


5. After all connections are set up, it should look like this:

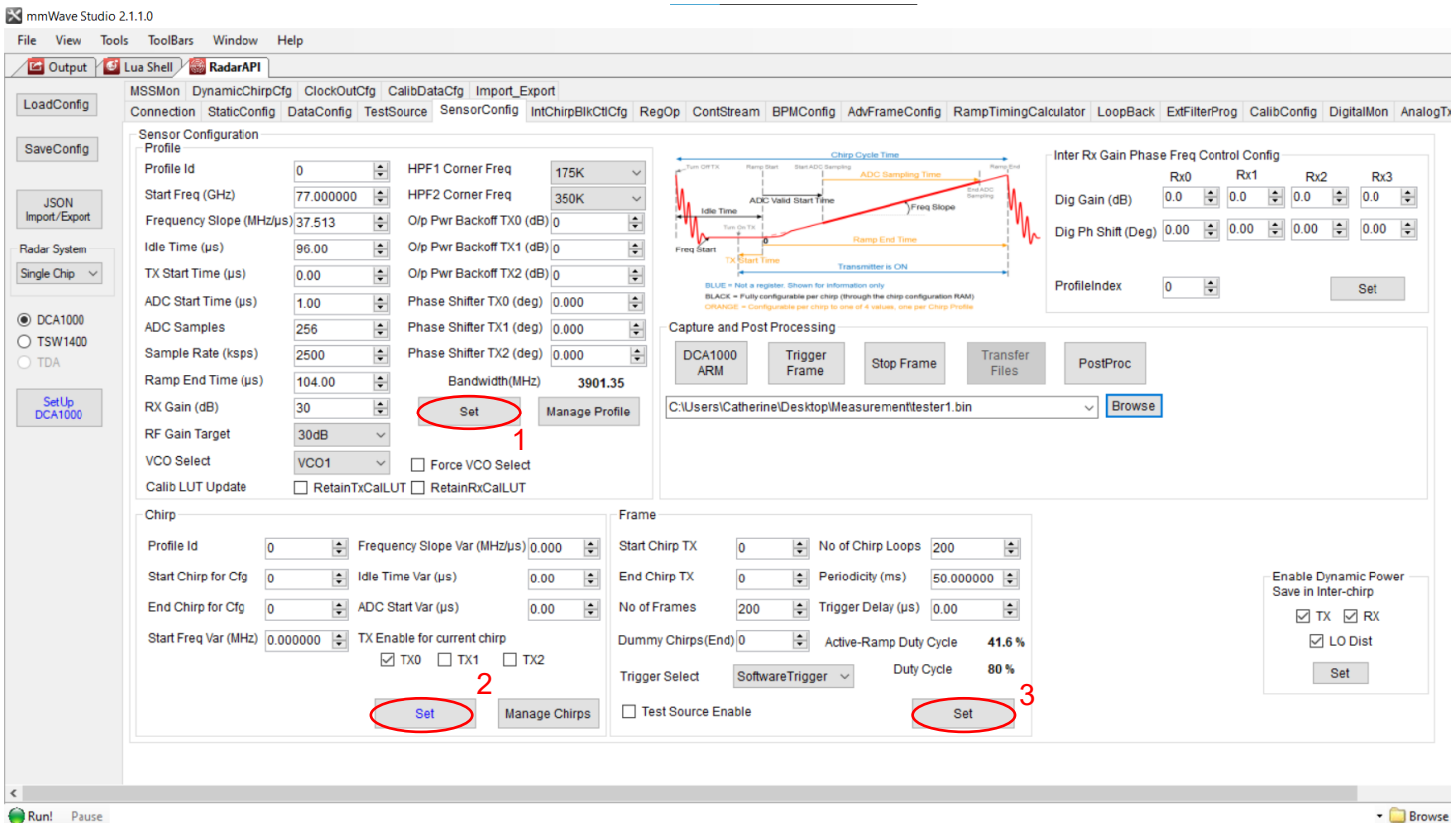


6. Next, select the **staticConfig** tab, and follow the image below:

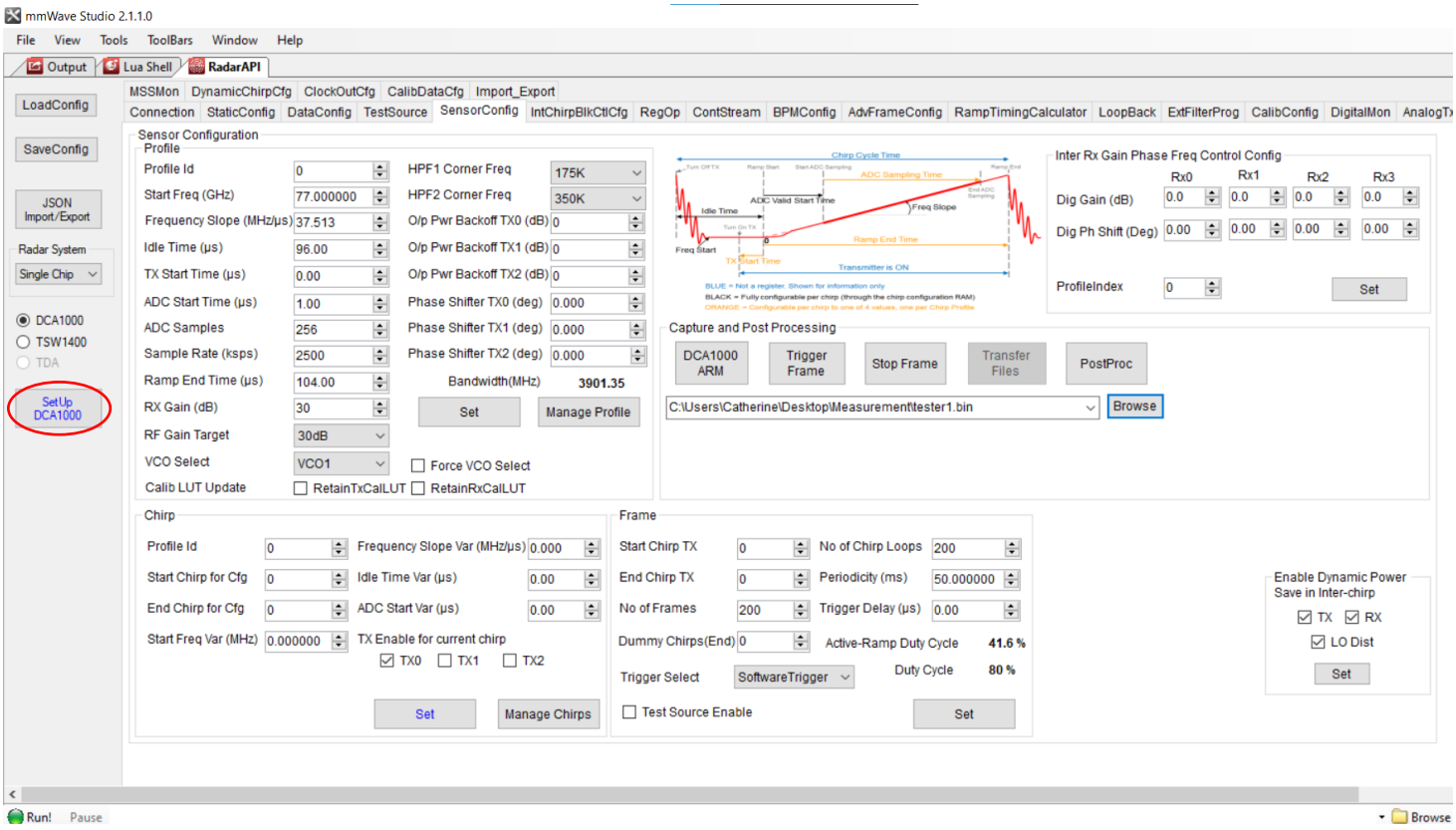
- Under ‘Basic Config’, the number of transmitter and receiver channels can be selected



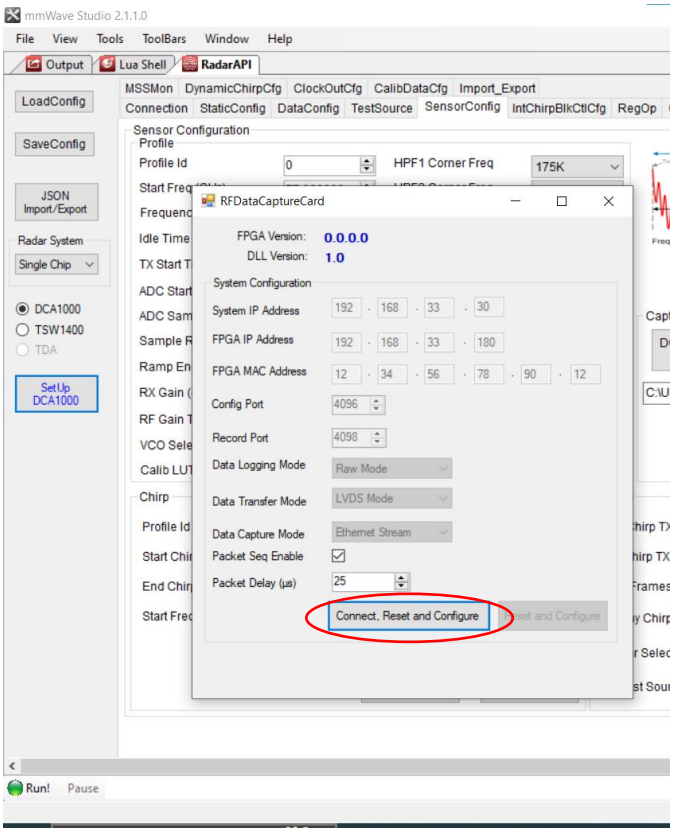
7. Select ‘SensorCongfig’ to configure the chirp. The values used are shown in image below:
   - Click ‘Set’ after filling out the ‘**Profile**’, ‘**Chirp**’ and ‘**Frame**’ sections

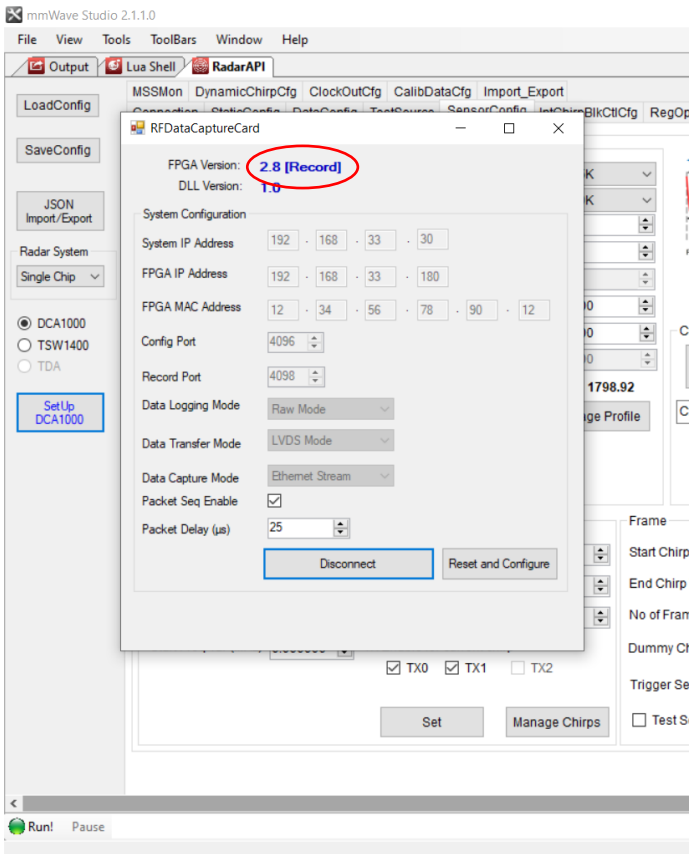


8. To set up the DCA1000 capture card, click ‘**Set Up DCA1000**’

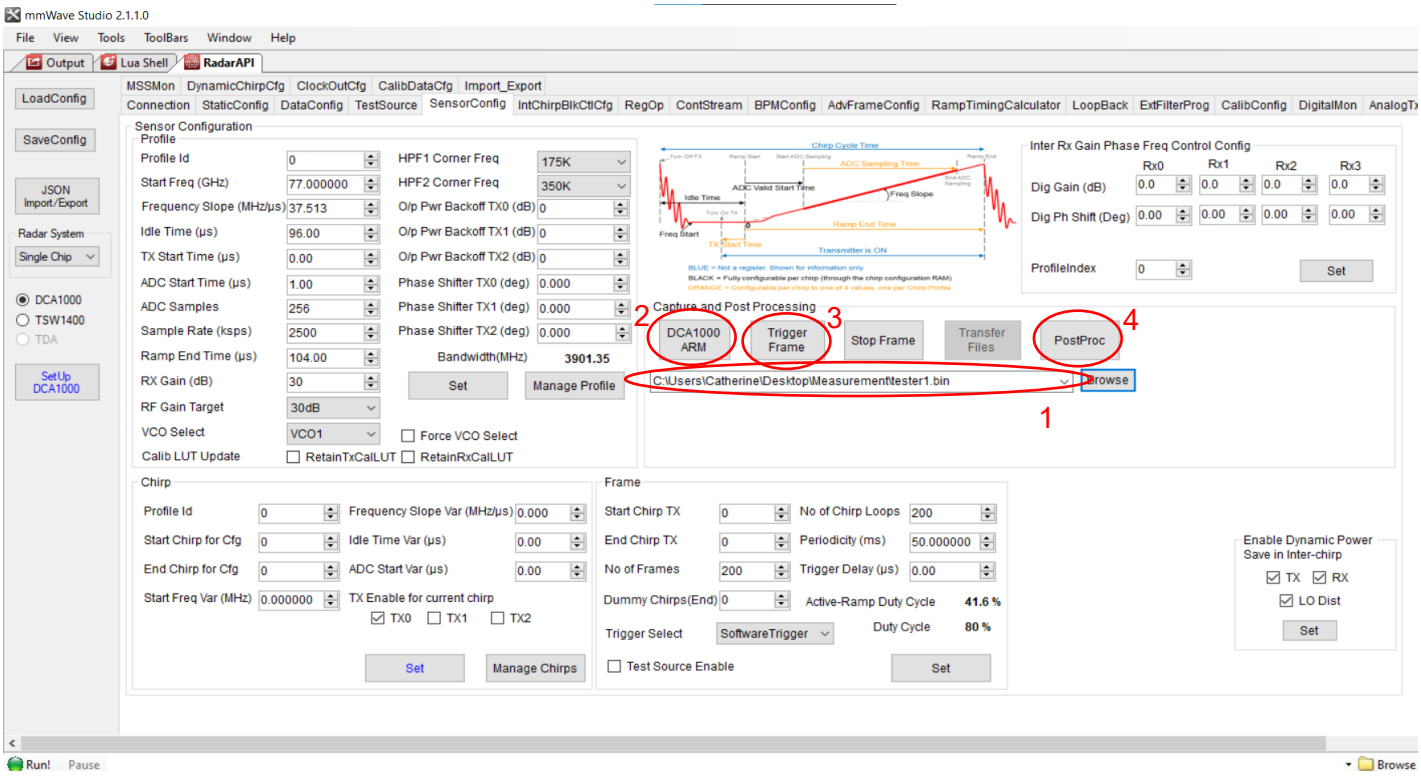


9. A pop up window will open. Click the '**Connect, Reset and Configure'**, the FPGA Version should now display '**2.8 (Record)**'. Close the pop up window

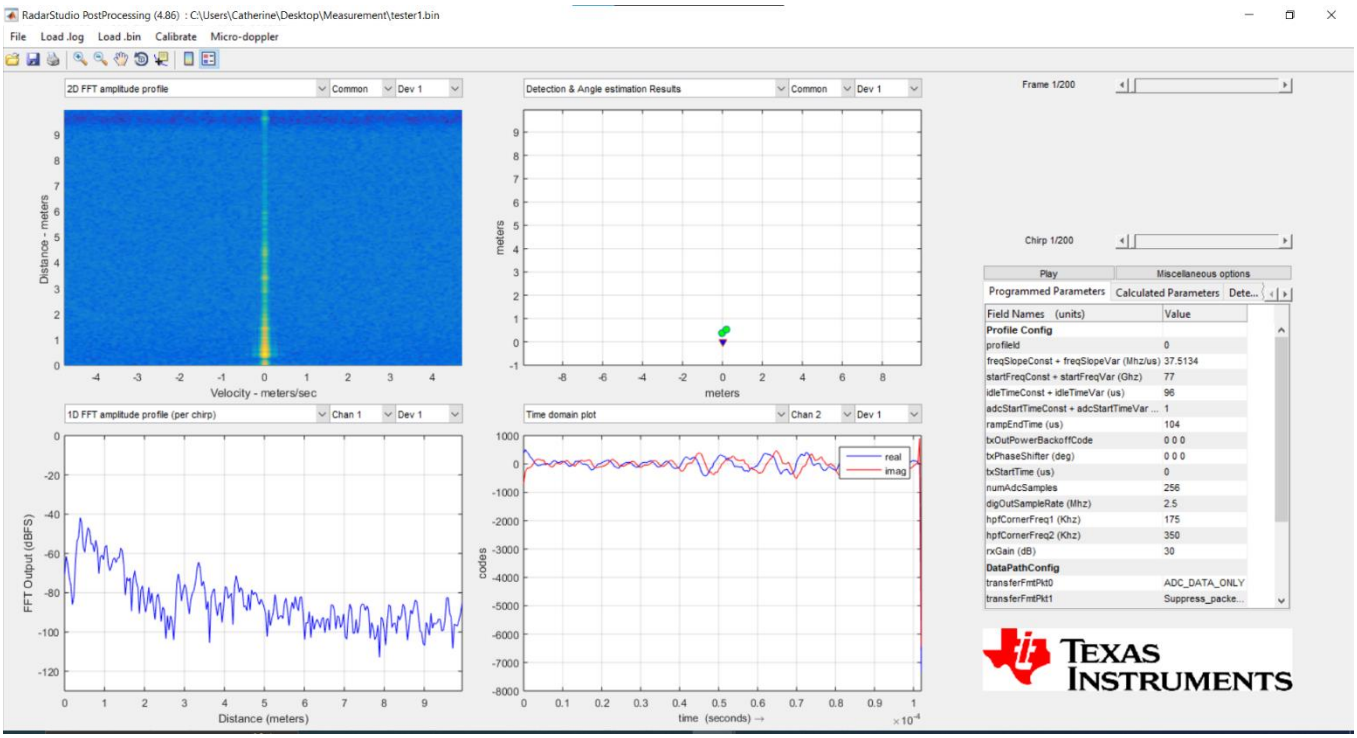




10. To capture the data, enter the location the binary file will be saved, and then enter the name for the binary file followed by .bin in the 'Capture and Post Processing' section . Click the '**DCA100 ARM**', followed by '**Trigger Frame**'. Wait for the data to be captured. To check if the data has all been captured, go the 'Output' tab. After the data been captured, a binary file will be saved in the selected location. To see the post-processing performed by the software click '**PostProc**'



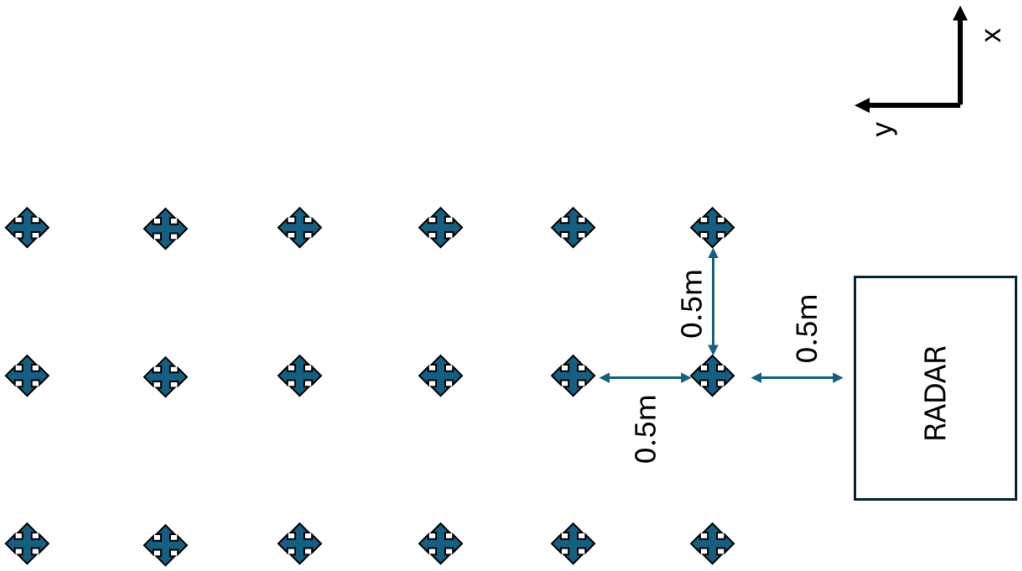
11. These are the results obtained when the radar is pointed 32cm from a wall



Data Collection

First test:

- To collect data, a **1m by 3m** grid was marked out using masking tape at each corner
- A cross was marked out on the masking tape at 0.5m intervals as shown below
- The range was measure from the midpoint of the 4 receiver to each cross
- 2 transmitter and 4 receivers were used
- A metal tin was placed at these crosses and range data were captured using the mmWave studio software

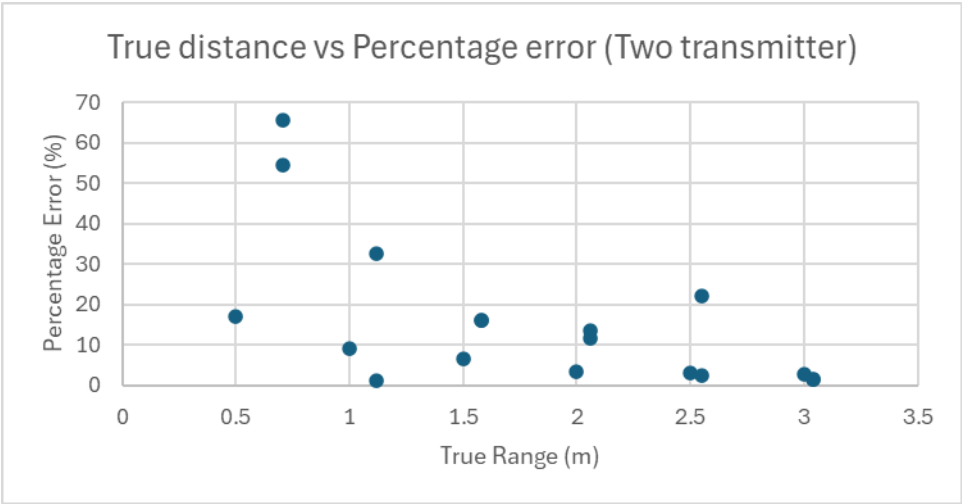


Second test:

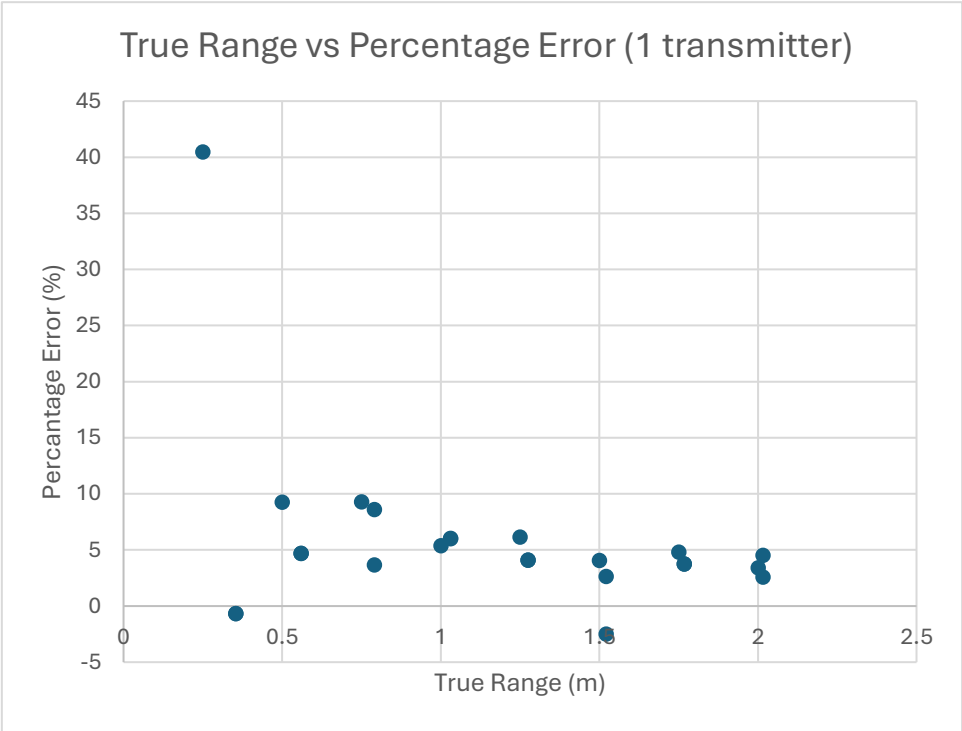
The same layout was used for 1 transmitter and 4 receivers, but with a grid size of 0.5m by 200m, and a 0.25m interval in the x and y directions.

Data from testing

- For each measured point, the range was extracted from the 1D FFT amplitude plot.
- The plot below shows the true range vs percentage error for the first test, which was conducted using 2 transmitter and 4 receivers



- The plot below shows the true range vs percentage error for the second test, which was conducted using 1 transmitter and 4 receivers

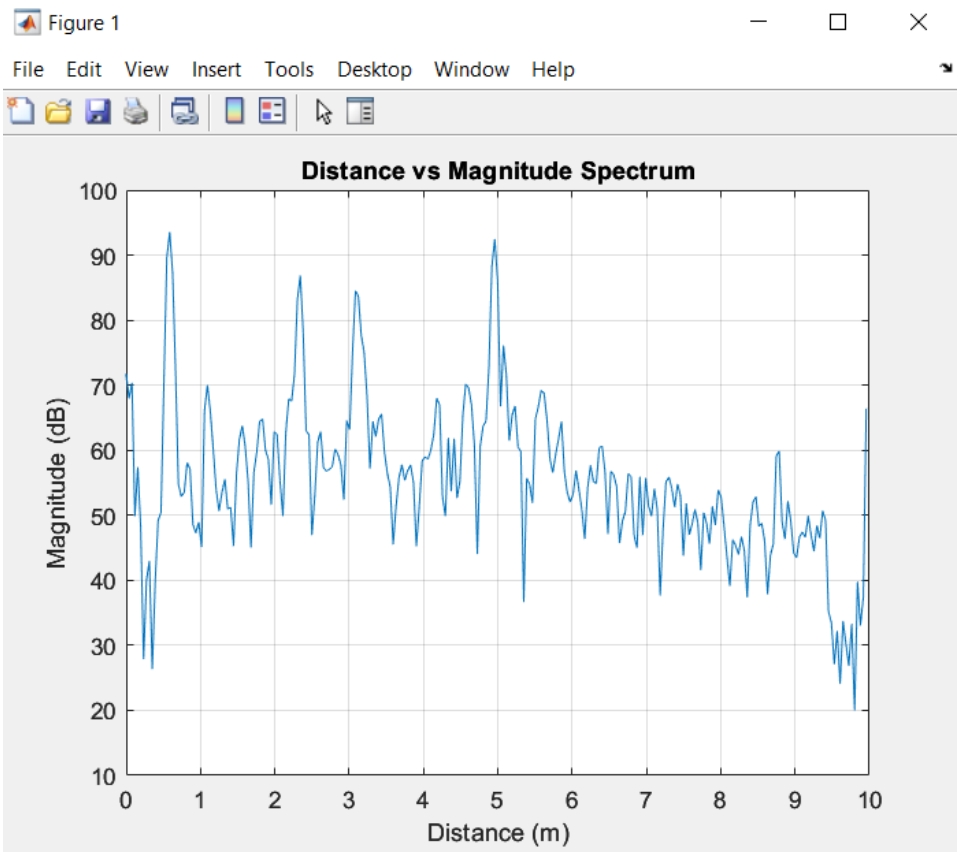


Preprocessing data

1. Read data from the binary file.
2. Combine the real and imaginary parts of the ADC data into complex data.
3. Reshape the data so that the rows represent ADC samples and the columns represent the chirp number.
4. Reshape the data so that the rows represent the receiver number and the columns represent ADC samples.
5. Create a matrix for each receiver, where the rows represent the ADC samples and the columns represent the chirp number.
6. Crop the data if there is an initial delay
7. Centre the data for each receiver around 0
8. Apply Hanning window to remove spectral leakage

MATALB files used for preprocessing are ‘readDCA1000.m’ and ‘multifile_readDCA1000.m’,

Below is a plot showing data for the first chirp, with a metal tin placed at x=0 and y=0.5 (range of 0.5 m). There is a peak at around 0.5 m, but there are also additional peaks, which are caused by some objects in the background.



Recreating the 1D FFT amplitude graph

1. Apply the FFT to the windowed data for each receiver
2. Find magnitude in dB of the range-time data
3. Calculate the frequency axis and then the range bin

$$d = \frac{c * f_b}{2 * slope}$$

$$slope = \frac{BW}{T}$$

Where c = speed of light (3×10^8)
f_b = beat frequency Hz
B = bandwidth (Hz)
T = ramp end time or duration of chirp (s)

4. Plot the range vs. magnitude (dB).

Object range prediction

1. Apply the FFT to the windowed data for each receiver
2. Find the location of the peak when the peak is exceeds a specified threshold
3. Calculate the range of the object using:

$$d = \frac{c * f_b}{2 * slope}$$

$$slope = \frac{BW}{T}$$

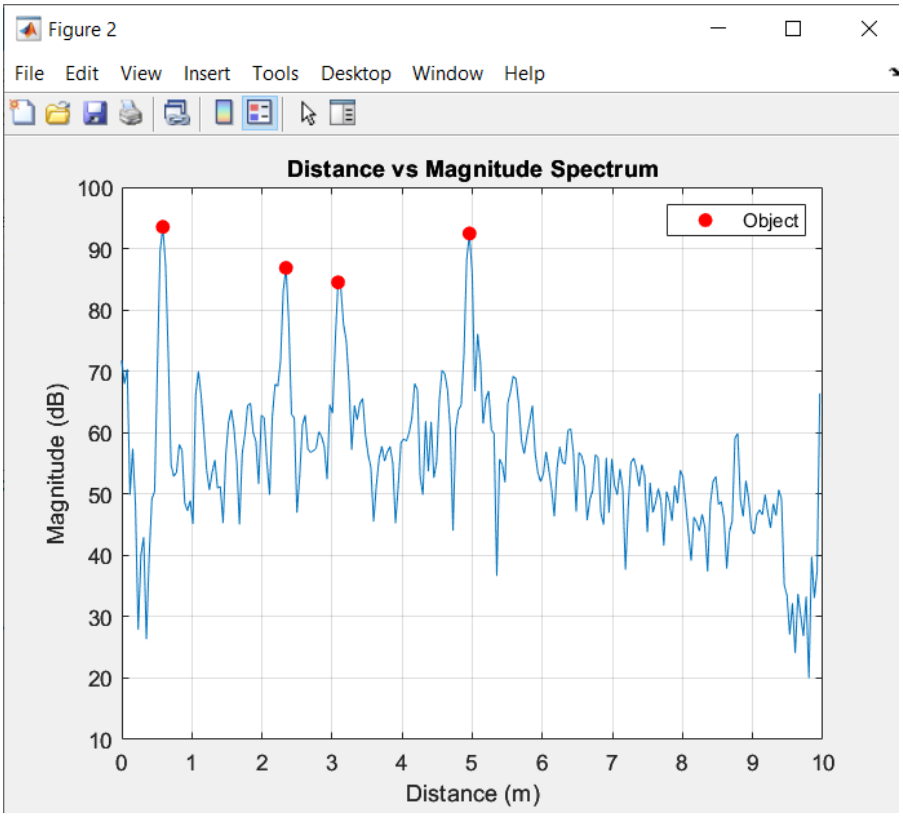
Where c = speed of light (3×10^8)
f_b = beat frequency Hz
B = bandwidth (Hz)

T = ramp end time or duration of chirp (s)

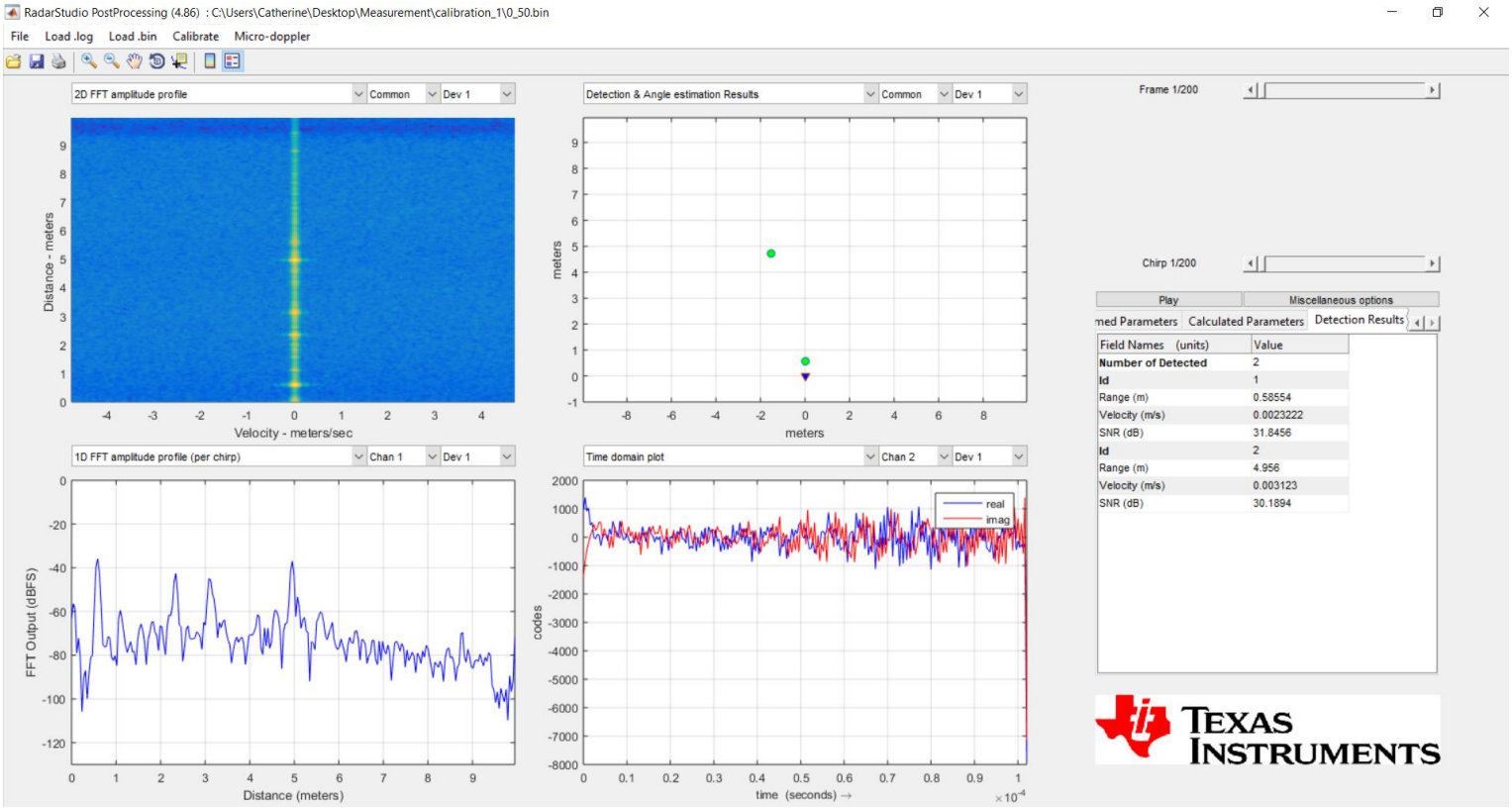
5. Mark the range of the detected objects on the range vs. magnitude (dB) plot.

The MATLAB file used for range prediction was **'FFT_amplitude_1D_profile.m'**.

The plot below shows distance vs. magnitude using the first chirp from the raw ADC data, when the metal tin was placed at x=0 m and y=0.5 m. The data is from receiver 1.



The plots from mmWave studio using the same data is shown below. The plot for the 1D FFT amplitude is very similar to the one generated using MATLAB. The only difference is the y-axis.

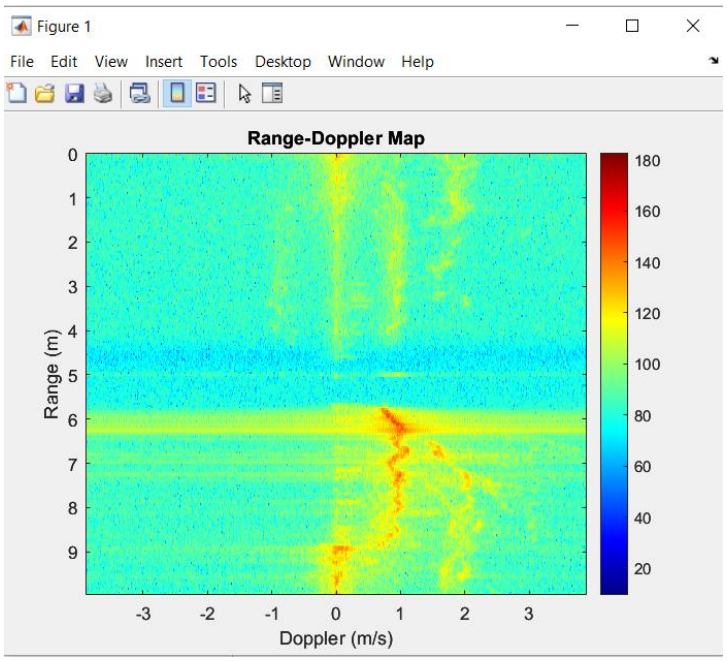


Velocity prediction

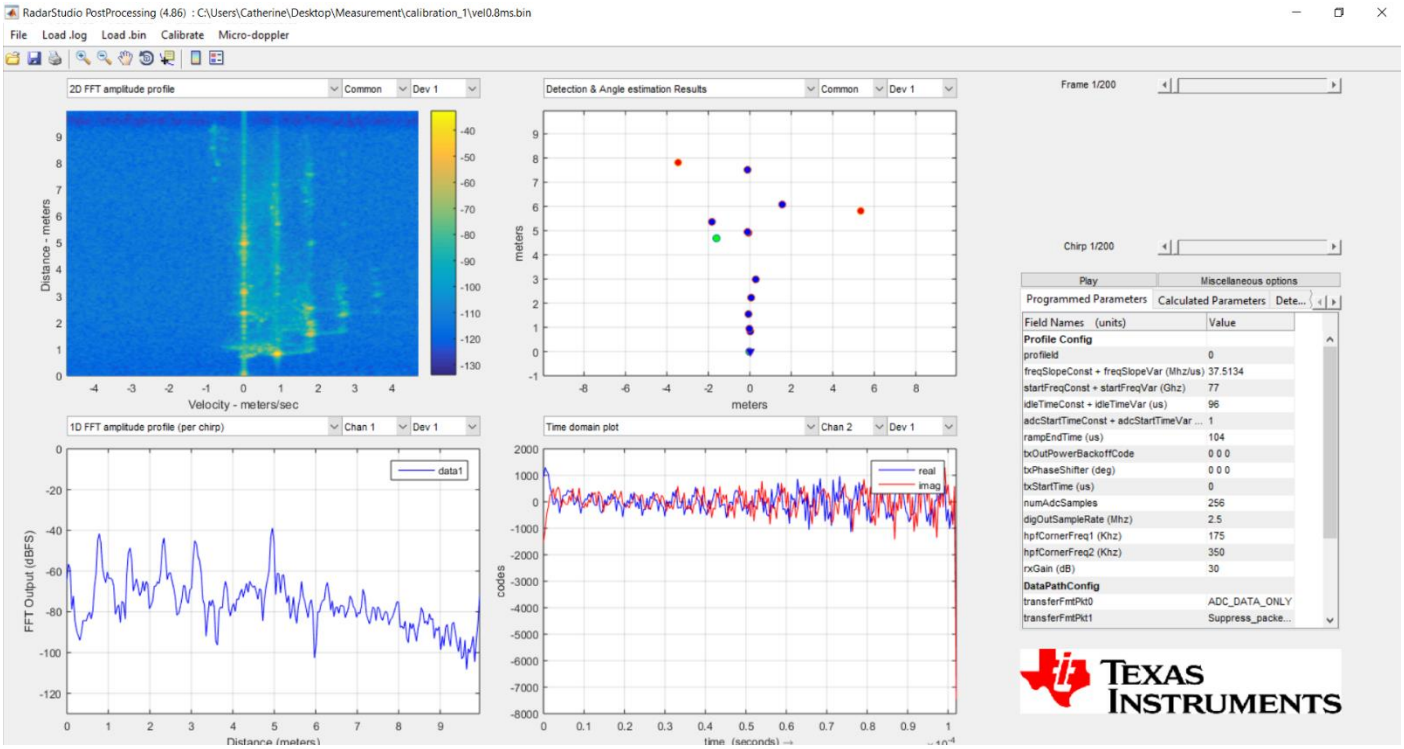
1. Apply the FFT across the ADC samples
2. Apply the FFT across the chirps
3. Calculate the frequency axis at 0
4. Convert Doppler frequency to velocity
5. Plot using imagesc

The MATLAB file used for velocity prediction was **'Doppler.m'**.

The imagsc plot below uses the first chirp of ADC data for a person walking from 0.5m to 4m at a speed of roughly 0.8m/s



Below is the plots generated from mmWave Studio:



Problem:

- The 2D FFT amplitude profile graph from mmWave Studio do not look the same as the one coded from MATLAB.
- Not sure why, maybe because the data from one receiver was used
- Uncertain about how to obtain the true velocity of an object for calibrating the radar.

Angle of Arrival using one transmitter (MUSIC algorithm)

1. Created a 4 by 256 matrix using ADC samples from the first chirp of each receiver, where 4 is the number of receivers and 256 is the number of samples
2. Compute the correlation matrix
3. Compute the eigenvalues and eigenvectors
4. Sort the eigenvalue from smallest to largest and rearrange eigenvector depending on the order of the eigenvalue
5. Extract the eigenvectors correspond to the noises (all the eigenvectors up to the number of objects)
6. Compute the steering vector using:

$$\begin{bmatrix} 1 \\ \exp\left(-j2\pi(1)\frac{d\ sin\theta}{\lambda}\right) \\ \dots \\ \exp\left(-j2\pi(M-1)\frac{d\ sin\theta}{\lambda}\right) \end{bmatrix}$$

Where d is the spacing between receivers

$$\lambda = \frac{c}{operating\ frequency}$$

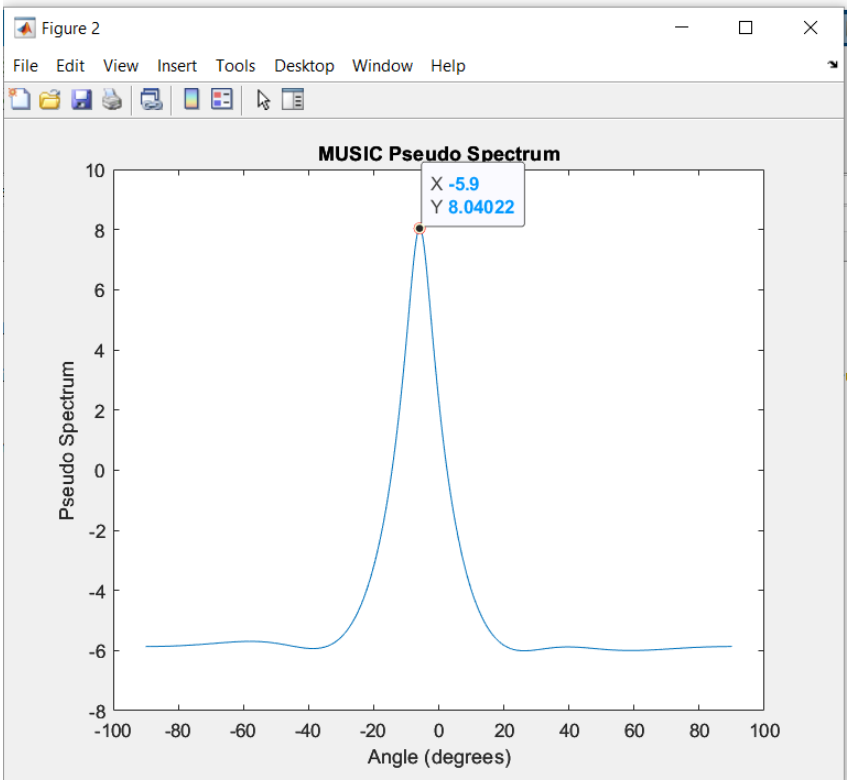
M is the number of receiver channels

7 Compute the pseudo-spectrum
8 Plot the pseudo spectrum

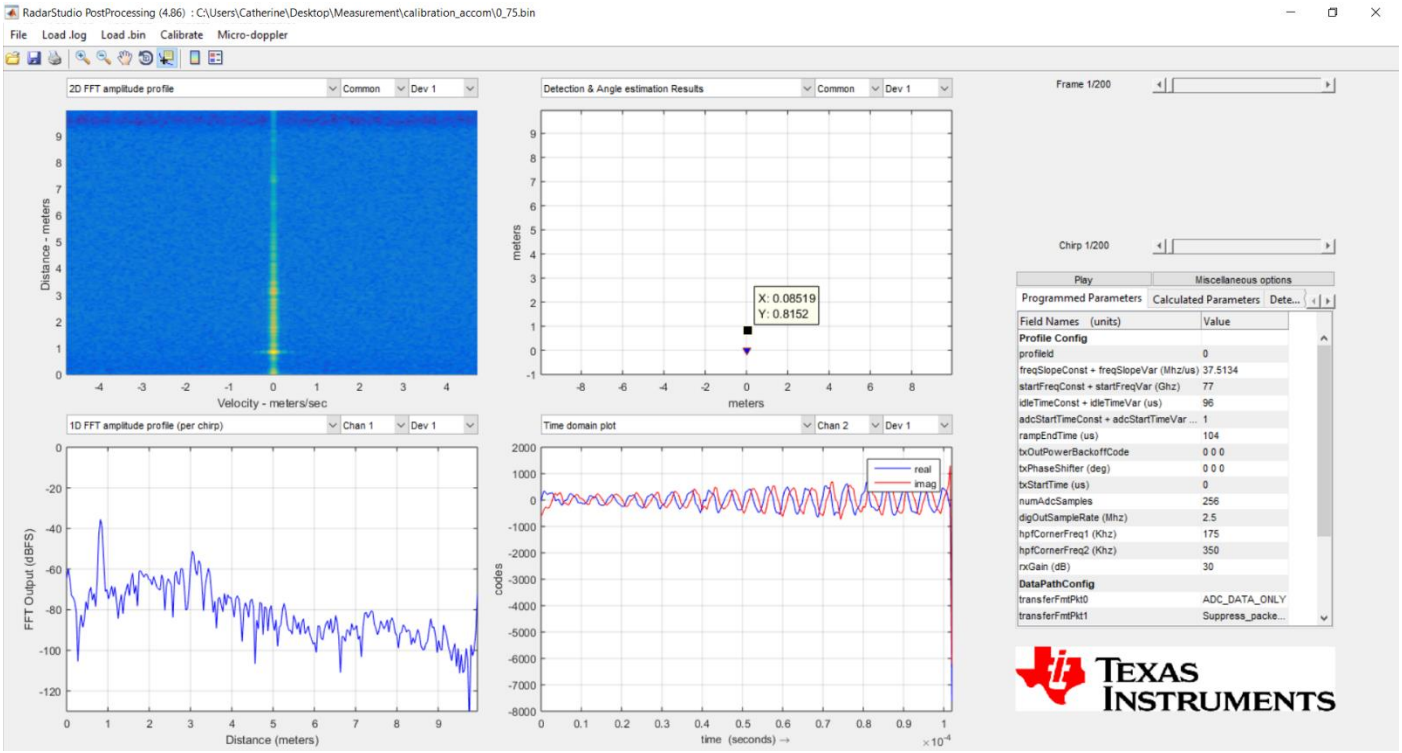
To test if the MUSIC algorithm was working correctly, a signal with known phases was passed through the algorithm to see if it would output the correct angle of arrival. The MATLAB code used for this test was '**Create_signal.m'** and '**Angle_tester.m'**.

The MATLAB code used for angle of arrival prediction was '**Angle_prediction_for_1_transmitter.m'**.

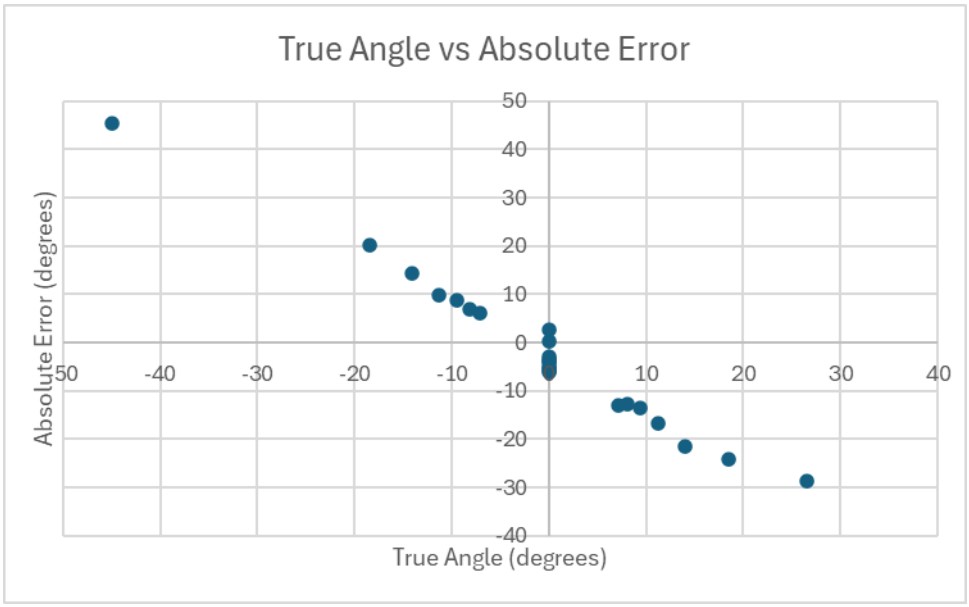
The plot below shows the pseudo-spectrum, which is used to depict the angle of arrival. A metal tin was placed at x=0 m and y=0.75 m. The predicted angle of arrival is 5.9°, because when the data was collected, the positive direction was taken to the right. However, the MUSIC algorithm assumes the positive x-axis is to the left, so the angles from the MUSIC algorithm need to be inverted.



The image below show the plots generated from mmWave. Using Pythagoras, the angle of arrival can be calculated from the x and y coordinates in the Detection and angle estimation plot and it was found to be 6.0°



Using the algorithm, data from the first chirp of Test 2 (1 transmitter and 4 receiver) was processed through the algorithm, and a plot was obtained, as shown below:



Problems:

1. During Test 2, it was noticed that a sheet of metal on the door, where the radar was facing, caused random points on the detection and angle estimation plot in mmWave Studio. To address this, a backpack was placed in front of the metal to block it. After this adjustment, instead of multiple points

being detected, only one point was detected, representing the backpack. Therefore, when the metal tin was placed at either x = -0.25 or x=0.25, two object were detected. This is because when the metal tin was placed at x=0, it hid the backpack. This maybe the reason why the MUSIC algorithm performed terribly when $x \neq 0$, the MUSIC algorithm might be detecting the angle of the backpack and not the metal tin.

2. MUSIC algorithm can only detect up to M-1 objects, where M is the number of receivers

Next step

1. Get the MUSIC algorithm working for two transmitters.
2. Apply velocity prediction over multiple chirps
3. Reproduce the detection and angle prediction plot from mmWave Studios