Level Sensing - Application Notes

Overview

This application note describes how to develop a high accuracy level sensor using uRAD Level Sensing. This level sensor can be used to measure the fill in tanks of solid or liquid materials, river or reservoir level for overflow applications or any application where it is necessary to accurately measure the distance in the frontal direction.

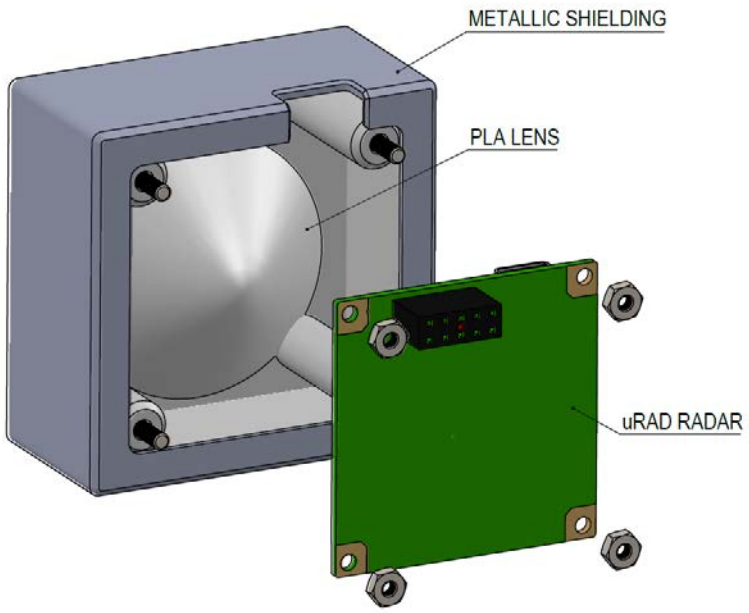
Devices

This application can be developed with our uRAD Level Sensing solution based on uRAD Industrial and uRAD Automotive. These radar models have been chosen, mainly because:

- Operate in the 60-64 or 77-81 GHz band, emission bands for level sensing applications.
- Allows to reach accuracies of ±1mm.
- Results in a very compact equipment.
- A very narrow field of view of less than 6 degrees can be achieved by adding a lens.

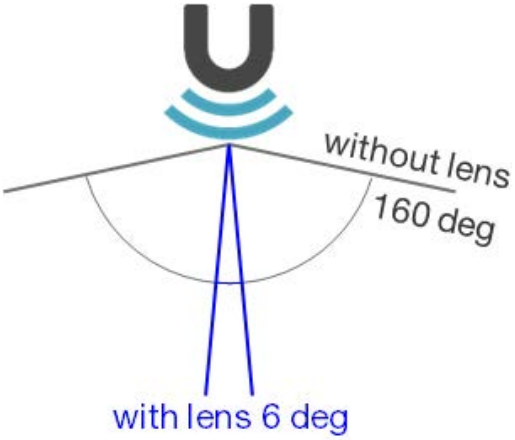
Hardware

The basic assembly is very simple and consists of uRAD board and a PLA box with an integrated lens and metallic shielding in its lateral sides.

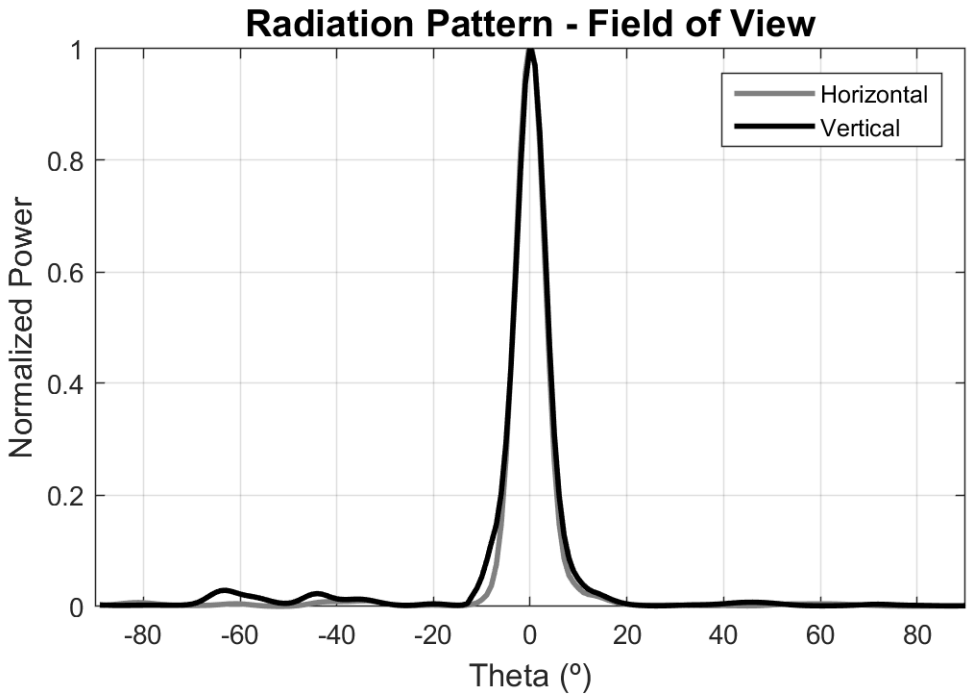


The lens reduces the field of view of uRAD Industrial from 160 degrees to less than 6 degrees. In this way, all the emitted power is focused in the straight direction.

For example, for tank filling applications, this is crucial as the radar must not see the silo walls and therefore, avoid unwanted bounces that do not allow it to measure accurately.

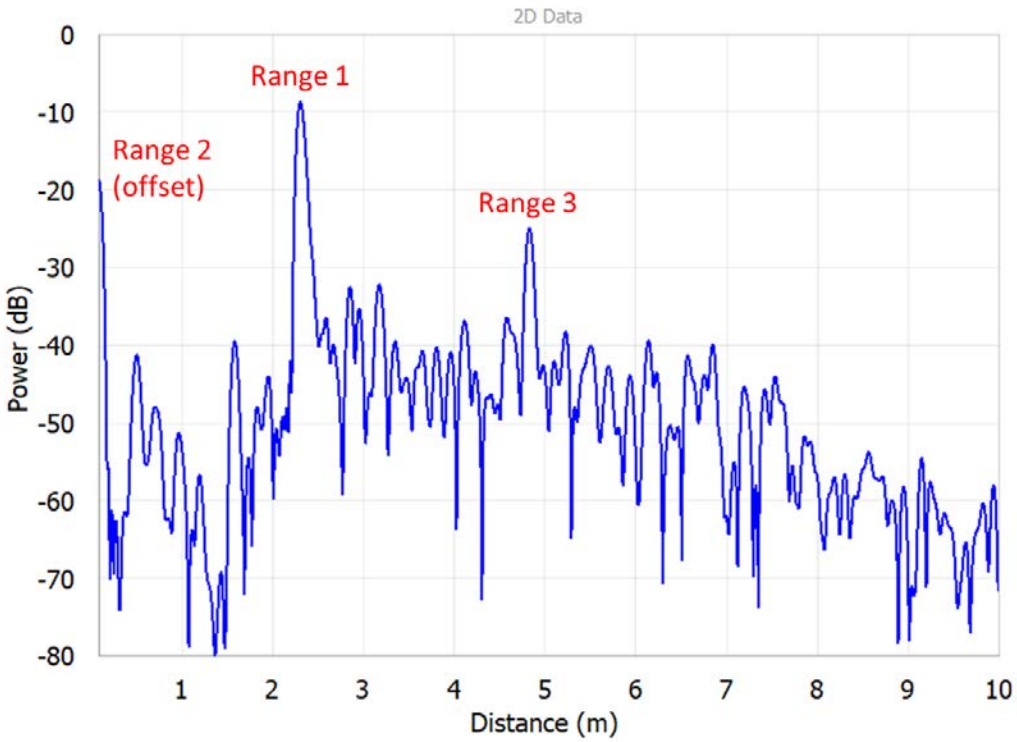


Next, the measured radiation pattern is shown.



Theoretical aspects

The radar calculates the distances to the targets using the Fast Fourier transform (FFT) of the received signal. By looking at the spectrum of the FFT, you can identify higher level peaks, such as reflections from objects or surfaces. The level sensing software allows you to visualize this spectrum and returns the range of the three peaks with the highest level. Here is an example:



In the example, the radar is fixed on a table aiming at the ceiling that is at a distance of 2.5 meters. Three peaks are identified in the graph.

- The most significant one, range 1, at 2.5 meters and that has the greatest amplitude is the ceiling, which provides the greatest reflection.
- There is a second peak, range 2, practically at 0, which does not correspond to any real value. At a very short distance, generally less than 10 cm, there is always a peak that is not real, called offset, which is inherent in radar technology and must be ruled out.
- A third peak, range 3, at 5 meters, corresponds to the double reflection produced by the table and the ceiling, which the radar is also capable of detecting.

The sensing software returns the distance of the three peaks with the greatest amplitude, but **the configuration allows you to select the range of distances to search for the three peaks**. Therefore, it is possible to easily eliminate the offset from the results or discard all those peaks outside the range of interest.

Functioning

First of all, the firmware is uploaded to uRAD board. We have developed a specific firmware for this application that is distributed free of charge with purchase. The process of uploading new firmware to uRAD is explained in the general user manual.

The sensor working is quite simple. The master device, that controls uRAD, sends the configuration commands to the radar by USB or UART. Once uRAD received the commands, starts sending the measurements values.

Most of the commands are set by default. Only a few have to be configured. In the examples, we have done this process transparent for the user. Therefore, it has to be configured:

```
#### CONFIGURATION PARAMETERS ####
model = 'IWR'            # 'IWR' or 'AWR'
maximum_distance = 12    # maximum distance to measure. From 12 to 150 meters
range_min = 0          # lower limit of the range of interest
range_max = 44           # upper limit of the range of interest
sampling_rate = 2        # min = 2, max = 20 samples per second
offset = 0               # apply an offset to all range measured values
```

- **model:** 'IWR' if your radar model is uRAD Industrial and works at 60 GHz, or 'AWR' if it is uRAD Automotive working at 77 GHz
- **maximum_distance**: radar can be configured to measure a maximum distance from 12 meters up to 150 meters. **It is better to choose the lowest possible value to have the best resolution.**
- **range_min** and **range_max**: distance range of interest. Limit the results to this range of interest.
- **sampling_rate**: measurements per second the radar sends to the master device. From 2 up to 20 samples per second.
- **offset**: add an offset (in meters) to the measurement values.

Programming Examples

Along with the hardware and firmware, various Python and C ++ scripts are also provided to control the radar, either by its USB connector or by the pin connector via UART.

- **level_sensing_UART.py**

This script is useful when sensor is controlled by the pin connector using a single UART channel. Only the distance results are sent (**FFT plot and amplitude values are not received**).

Some extra configuration parameters have to be set:

```
#### OUTPUT RESULTS ####
saveResults = True          # Save results in .txt file
printResults = True         # Print results in the console
numberOfPeaksToSave = 3     # Number of peaks to save (max = 3)
fileSizeMinutes = 2         # Minutes to split the .txt file

#### UART PORT ####
port_name = '/dev/serial0'

#### RESET ####
reset_pin_number = 6    # GPIO pin number of the Master Device
reset = True
```

- **saveResults**: results are saved inside *output_files* folder in ***YYYY_MM_DD_HH_mm_results.txt*** file. Each results file has a header name with date and time.
- **printResults**: print results on the Python console.
- **numberOfPeaksToSave**: choose the number of peaks (most significant distance values) to save. Maximum 3.
- **fileSizeMinutes**: split the results file by minutes, to limit the size of each file.

Also, the UART port name and whether you want to use a reset pin (and its pin number) has to be configured.

- **level_sensing_USB.py**

Similar to the previous one but controlled by USB. It does not include the reset pin. **Therefore, each time you want to run again the code, you have to manually reset the radar with the physical reset button**.

```
#### OUTPUT RESULTS ####
saveResults = True          # Save results in .txt file
printResults = True         # Print results in the console
numberOfPeaksToSave = 3     # Number of peaks to save (max = 3)
fileSizeMinutes = 2         # Minutes to split the .txt file
```

```
#### USB PORT ####
configPort_name = 'COM1'
dataPort_name = 'COM1'
```

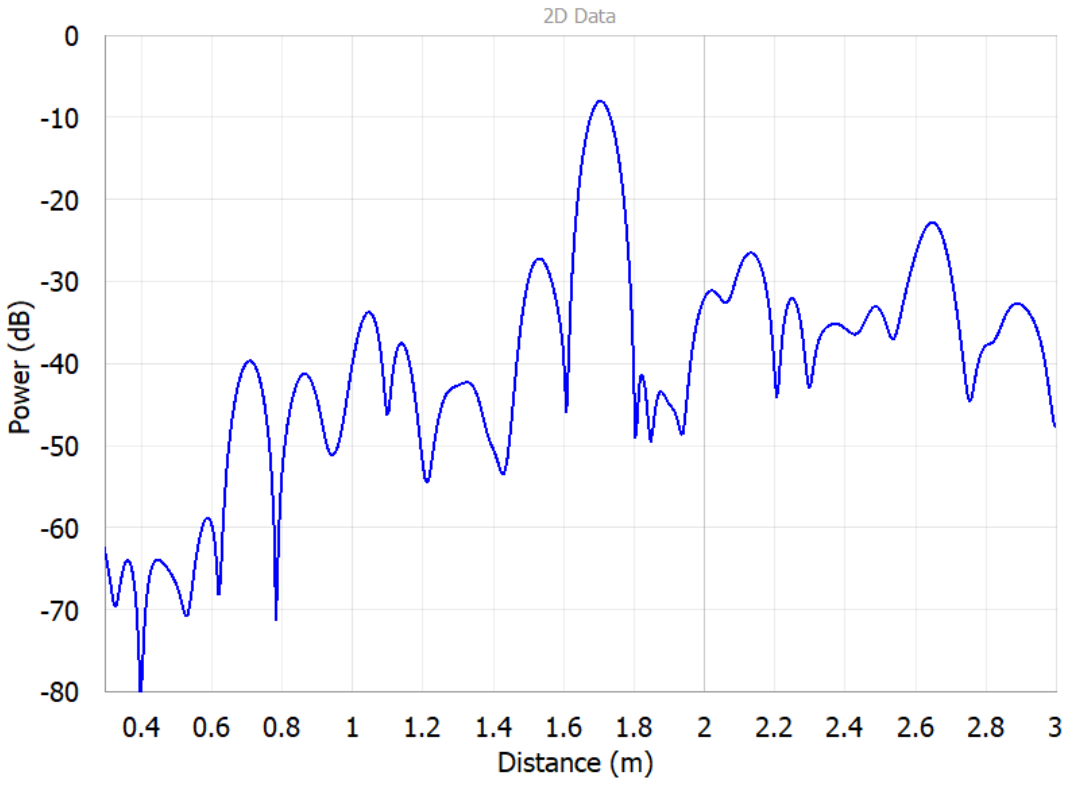
- **level_sensing_UART_GUI.py**

The most complete one because allows you to obtain **distance results and also amplitude of the peaks**. The configuration commands include to send the IQ values for the FFT plot. Therefore, the peaks plot (FFT) can be visualized if selected. This is especially useful for calibration.

```
#### OUTPUT RESULTS ####
calculateAmplitude = True    # Calculate amplitude of the peaks besides range
saveResults = True           # Save results in .txt file
printResults = True          # Print results in the console
plotSpectrum = True          # Show Amplitude Vs Distance plot
numberOfPeaksToSave = 3      # Number of peaks to save (max = 3)
fileSizeMinutes = 2          # Minutes to split the .txt file

#### UART PORT ####
port_name = '/dev/serial0'

#### RESET ####
reset_pin_number = 6     # GPIO pin number of the Master Device
reset = True
```



- **level_sensing_USB_GUI.py**

Similar to the previous one, but via USB. **Remember that every time you want to launch the code again, you must manually reset the radar with the physical reset button**.

```
#### OUTPUT RESULTS ####
calculateAmplitude = True    # Calculate amplitude of the peaks besides range
saveResults = True           # Save results in .txt file
printResults = True          # Print results in the console
plotSpectrum = True          # Show Amplitude Vs Distance plot
numberOfPeaksToSave = 3      # Number of peaks to save (max = 3)
fileSizeMinutes = 2          # Minutes to split the .txt file

#### USB PORT ####
configPort_name = 'COM2'
dataPort_name = 'COM1'
```

- **level_sensing_function.py**

This example is a script that shows how to create a simple function to take a single measurement (averaged over the number of samples you want) each time the function is called. Therefore, it allows the user to take only one measurement at the desired time.

- **level_sensing_UART.cpp**

This C++ script is equivalent to the Python script ***level_sensing_UART.py***. The difference in this script is that, instead of defining the configuration parameters, these have to be directly introduced (the equivalent number) in the configuration commands.

The configuration commands, in the code, are these lines:

```
"flushCfg\n", \
"dfeDataOutputMode 1\n", \
"channelCfg 1 1 0\n", \
"adcCfg 2 1\n", \
"adcbufCfg 0 1 1 1\n", \
"profileCfg 0 60 7 7 114.4 0 0 31.23 1 512 5000 0 0 48\n", \
"chirpCfg 0 0 0 0 0 0 0 1\n", \
"frameCfg 0 0 10 0 500 1 0\n", \
"lowPower 0 0\n", \
"guiMonitor 1 0 0 0 0 1\n", \
"RangeLimitCfg 2 1 0.1 12.0\n", \
"sensorStart\n"
```

The 5 relevant numbers to set are highlighted in yellow that corresponds to:

- start frequency according to ***model***, 60 for IWR or 77 for AWR.
- chirp slope according to ***maximum_distance***, from 1.87 (150 meters) to 31.23 (12 meters). Both values are related with the formula:

$$chirp\_slope\ [MHz/\mu s] = \frac{374.7405725}{maximum\_distance\ [m]}$$

- time between measurements according to ***sampling_rate***, from 50 (20 samples/s) to 500 (2 samples/s). Both values are related with the formula:

$$measurement\_periodicity\ [ms] = \frac{1000}{sampling\_rate}$$

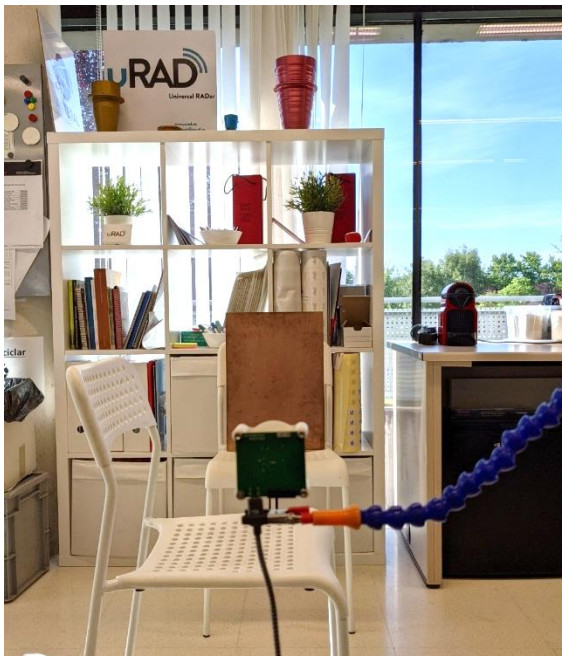
- *range_min* and *range_max*.

Moreover, a similar code called ***uRAD_LevelSensing.ino*** is also provided for Arduino platform

Example with real measurements

Next, a real case is presented.

The setup is as follows. The radar is placed at a distance of 1.7 meters from a metal sheet. Halfway through, a crystal will be placed. Three situations are going to be measured: without glass, with the glass perpendicular to the radar and with the glass tilted.



The radar is connected to a computer. It is powered directly by the USB where it is also configured and received the data.

The script used is ***level_sensing_USB_GUI.py*** since we want to show the peak graph in addition to saving the results.

The configuration is:

```
#### CONFIGURATION PARAMETERS ####
model = 'IWR'            # 'IWR' or 'AWR'
maximum_distance = 12    # maximum distance to measure. From
range_min = 0.3          # lower limit for the range of interest
range_max = 3            # upper limit for the range of interest
sampling_rate = 2        # min = 2, max = 20 samples per second
offset = 0               # apply an offset to all range measured values
```

We connect the radar and see the COM ports that have been assigned. We configure the rest of the script parameters as follows:

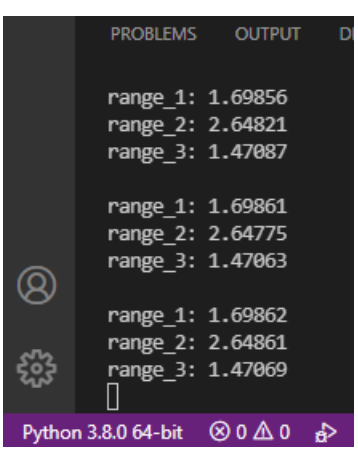
```
#### OUTPUT RESULTS ####
calculateAmplitude = True    # Calculate amplitude of the peaks besides range
saveResults = True           # Save results in .txt file
printResults = True          # Print results in the console
plotSpectrum = True          # Show Amplitude Vs Distance plot
```

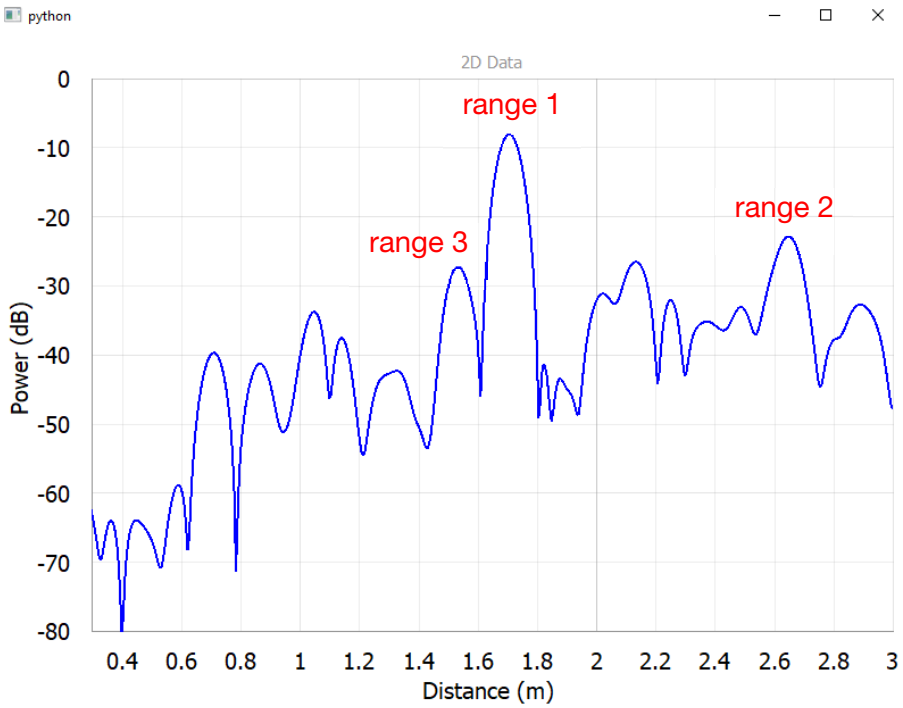
```
numberOfPeaksToSave = 3      # Number of peaks to save (max = 3)
fileSizeMinutes = 2          # Minutes to split the .txt file

#### USB PORT ####
configPort_name = 'COM1'
dataPort_name = 'COM1'
```

- **Without glass**

After running the script, we obtain the following measured results:



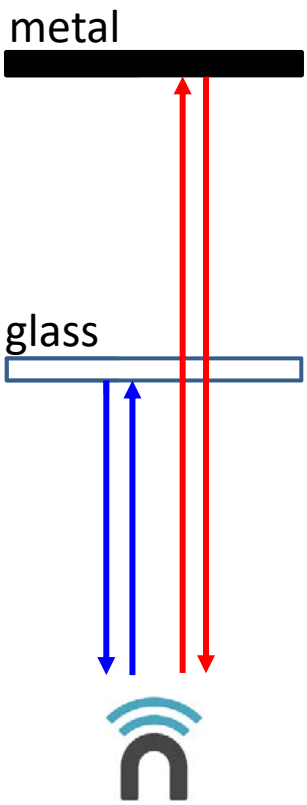


Looking at the results both in the Python console and in the graph, we see the three peaks that are identified. The results are ordered from highest to lowest amplitude of the peak.

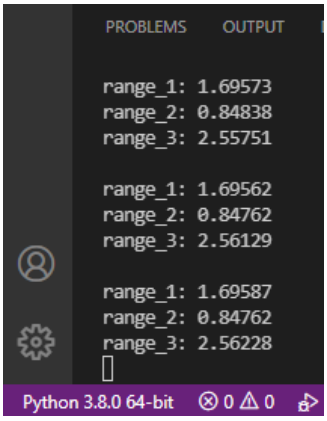
At the distance of 1.69856 meters, the metal has been measured with an amplitude well above the other two peaks, which do not actually correspond to any surface of interest.

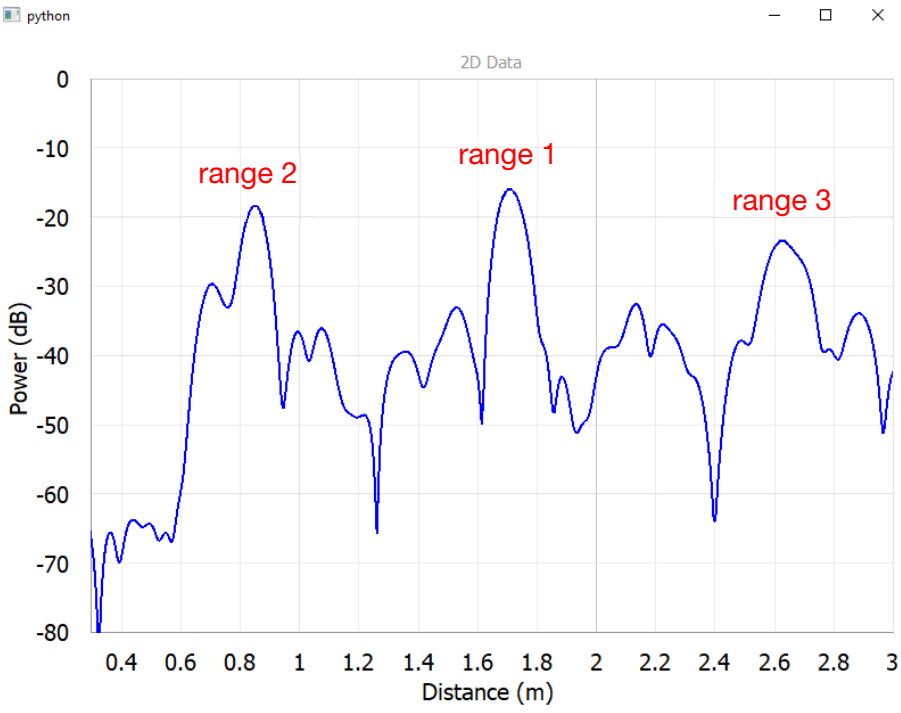
The rest of the peaks of the spectrum, including range 2 and 3, are produced by the FFT processing itself and by rebounding from the environment.

- **Perpendicularly placed glass**



The obtained results are:



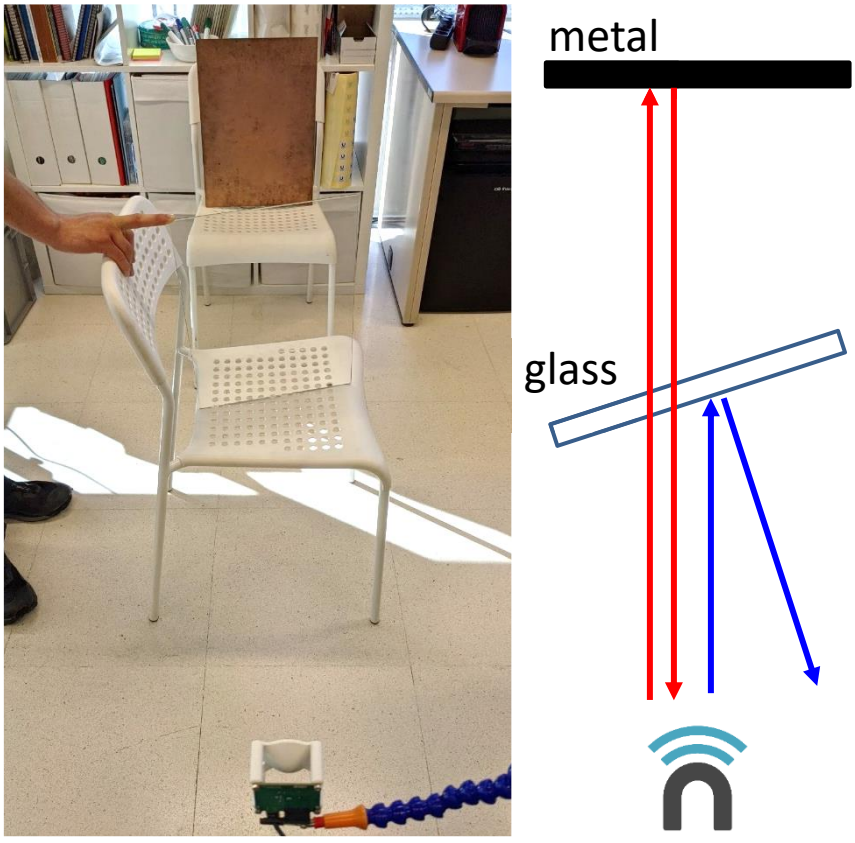


In this case, how the glass appears at a distance of 0.84838 meters and the metal continues to appear at a distance of 1.69573 meters is observed. In this case, the amplitude level of both peaks is very similar, so the value of range 1 could be jumping between both values or even attribute the peak of greater amplitude to the glass. In addition, the peak corresponding to the metal has decreased in amplitude as a consequence of the attenuation of the wave through the glass.

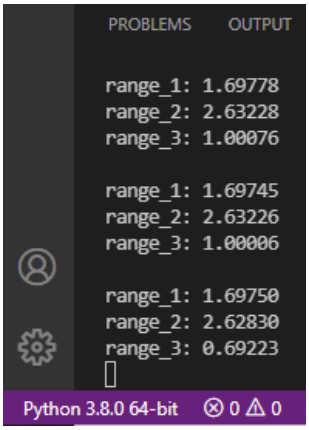
- **Tilted placed glass**

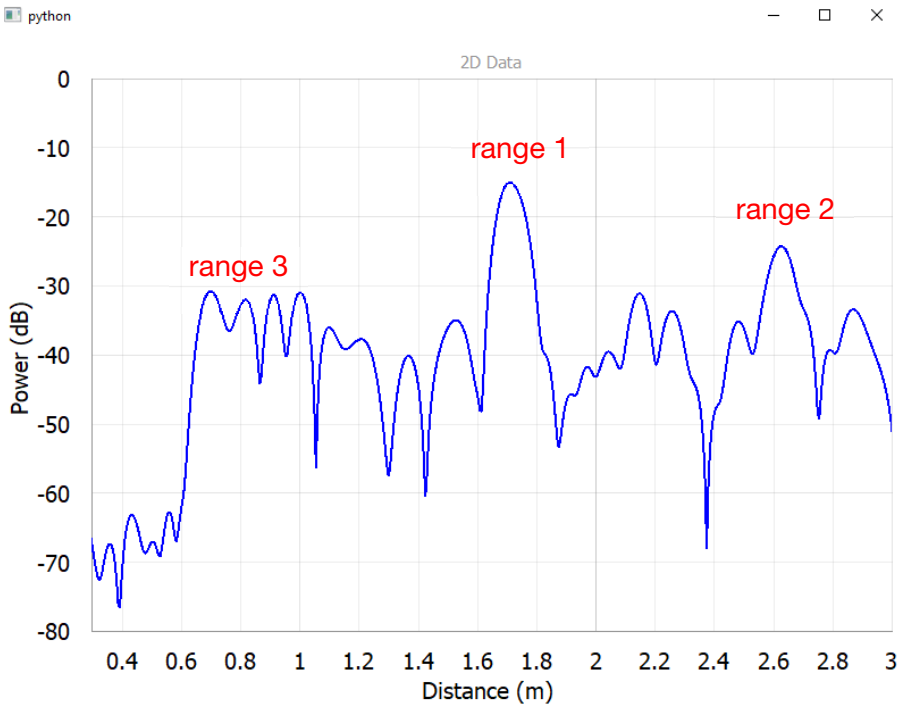
In this case, we tilt the glass to observe its peak disappears. With a tilt of 20-30 degrees, it is enough for the peak to disappear.

This can be very useful, for example, in real situations where it is required to isolate the radar with a cover or where it is unavoidable to pass through various materials. In that case, a tilt on that surface will mitigate the effect of the reflection.



By tilting the glass, most of the reflected power does not come back to the radar, as shown in the image above.





How the peak of the glass has been completely mitigated is observed. In this new situation, range 1 is unequivocally identified with the metal and its amplitude being much higher than the rest of the peaks.

The text file generated with the results, is saved in the *output files* folder. The format is as follows:

```
1.69778 2.63228 1.00076 1622787713.261
1.69745 2.63226 1.00006 1622787713.762
1.69750 2.62830 0.69223 1622787714.261
1.69761 2.63123 0.69210 1622787714.762
1.69776 2.63211 1.00037 1622787715.260
```

The values of each measure are saved in each row, one frame per row, ordering by columns:

```
range1   range2   range3   time_stamp
```