XETHRU
BY NOVELDA

XeThru Sensors Introduction

Getting Started with XeThru

XeThru Application Note **by Novelda AS**

Rev. A - October 10. 2017

Summary

This guide will provide you with the necessary information to get you started on your first XeThru project using XeThru software. After reading this the user will have a general understanding of XeThru sensors functionality, output and potential.

1 Introduction

The XeThru sensor modules enable unique sensor solution based on Novelda's Ultra Wide Band impulse radar technology and signal processing. The module emits electromagnetic pulses from a transmitter (TX) and the receiver (RX) samples the reflected energy. After digital signal processing, the XeThru module outputs data through the different communication interfaces. The signal processing or radar user can also choose to read raw radar data and apply custom signal processing algorithms to further enhance the capabilities of the sensors. Combining this with a basic level of programming skills the user is able to create complex, innovative and exciting new products.

From baby-monitoring to occupancy sensing, the XeThru module application areas are vast. XeThru sensors are the key enabler for several research projects and products all over the world. From big research and development companies to student projects and makers. To find out more about research projects and XeThru applications please visit the XeThru blog . Here you can find exciting new project and experiments.

This guide provides step-by-step guidelines to get started using XeThru sensor module. Please visit our community pages on our web page for questions and discussions. Here users can find answers to common questions and can post their own questions and receive support from other users and Novelda employees.

2 Overview

This application note is written for X4 modules and following software versions:

Software	Version	Supported platform	Supported modules
XeThru Explorer	2.5.0	Windows, Ubuntu & OSX	X4M300, X4M200
Module Connector	1.4.2	Windows, Ubuntu, OSX & Rasbian	X4M300, X4M200, X4M03

Keep in mind that certain features in XeThru software may change with new releases. Future versions of XeThru Explorer will support the radar development kit, X4M03.

2.1 Development Flow

This application note will introduce the readers to XeThru sensors in an intuitive order by starting at the highest level of data acquisition and then introduce more complex methods of acquiring data.

Typical d evelopment flow for XeThru sensors:

- Exploring XeThru sensor functionality and do recordings using XeThru Explorer
- Analyse recorded data using desired data analyze software
- Application development with Module Connector
- Embedded host solutions using Module Communication Protocol Wrapper

2.2 Profiles

XeThru sensors containing profiles lets the user access pre-processed data without having to write signal-processing algorithms. Profiles are defined as a specific sensor configuration to be used in a specific use-case. For example, X4M300 use the *presence* profile to output presence monitoring data directly to its host. In addition, modules support accessing raw radar data through the XeThru Embedded Platform (XEP) interface. Using XEP the user can write their own data processing algorithm for specific use cases. Fig.1 shows how hosts communicate with modules. For more information on profiles and specific sensor configurations see X4M300 and X4M200 datasheets.

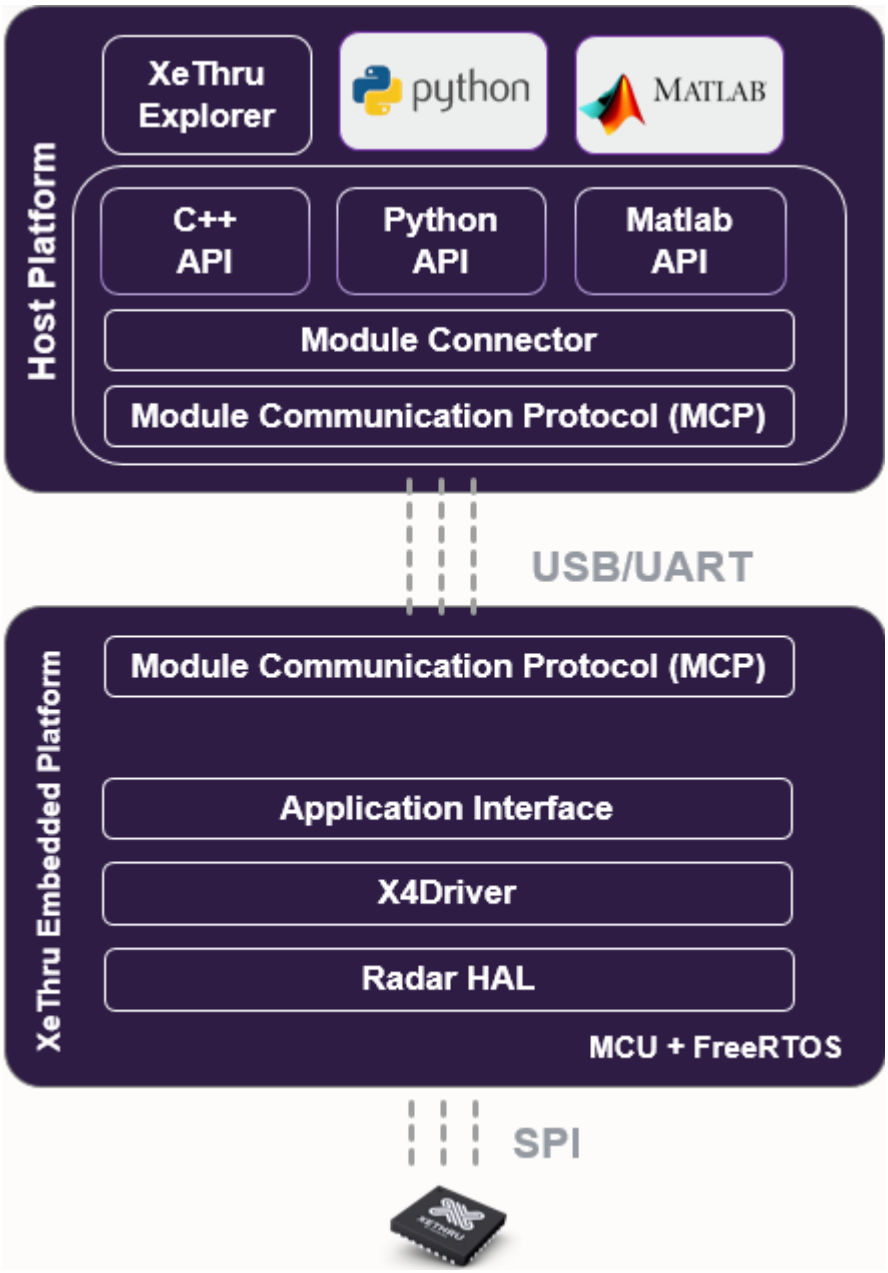


Fig. 1. XeThru software hierarchy.

2.3 Sensor Set-up

Setting up the sensor correctly ensures best results. The sensor should be placed on a stationary surface with the antennas pointing towards the area of interest, see Fig. 2. A tripod or a 3D printed mount will provide a simple and flexible setup. Conductive materials, as metal and water, in front of the antenna will block the signals and distort the sensor output.

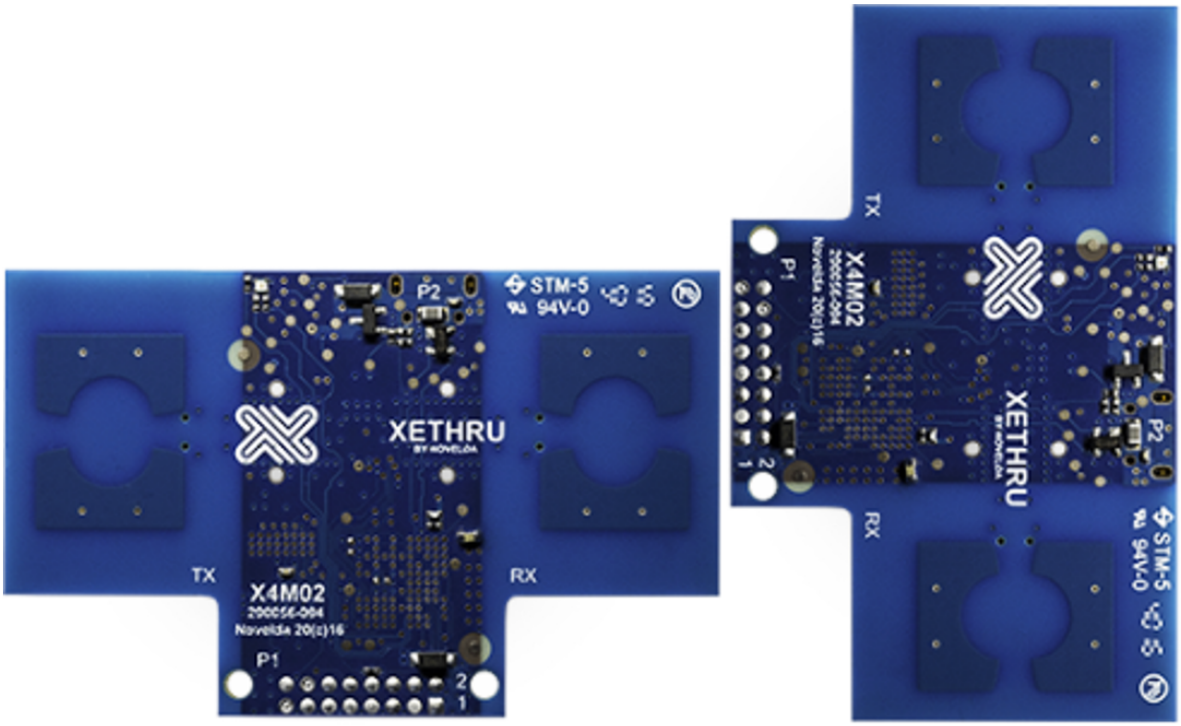


Fig. 2. XeThru sensor antennas are marked with RX (receiver) and TX (transmitter).

3 XeThru Explorer

XeThru Explorer is an easy-to-use graphical user interface that plots the module output. This is the easiest way to get to know the XeThru module's capabilities and outputs and doing first data recordings. All data processing is done on the sensor module itself, XeThru Explorer is simply presenting the data.

After installing XeThru Explorer, follow these steps:

- Connect the XeThru module to a computer with a micro USB cable
- XeThru explorer shows the user which device port the module is connected to, see Fig. 3.

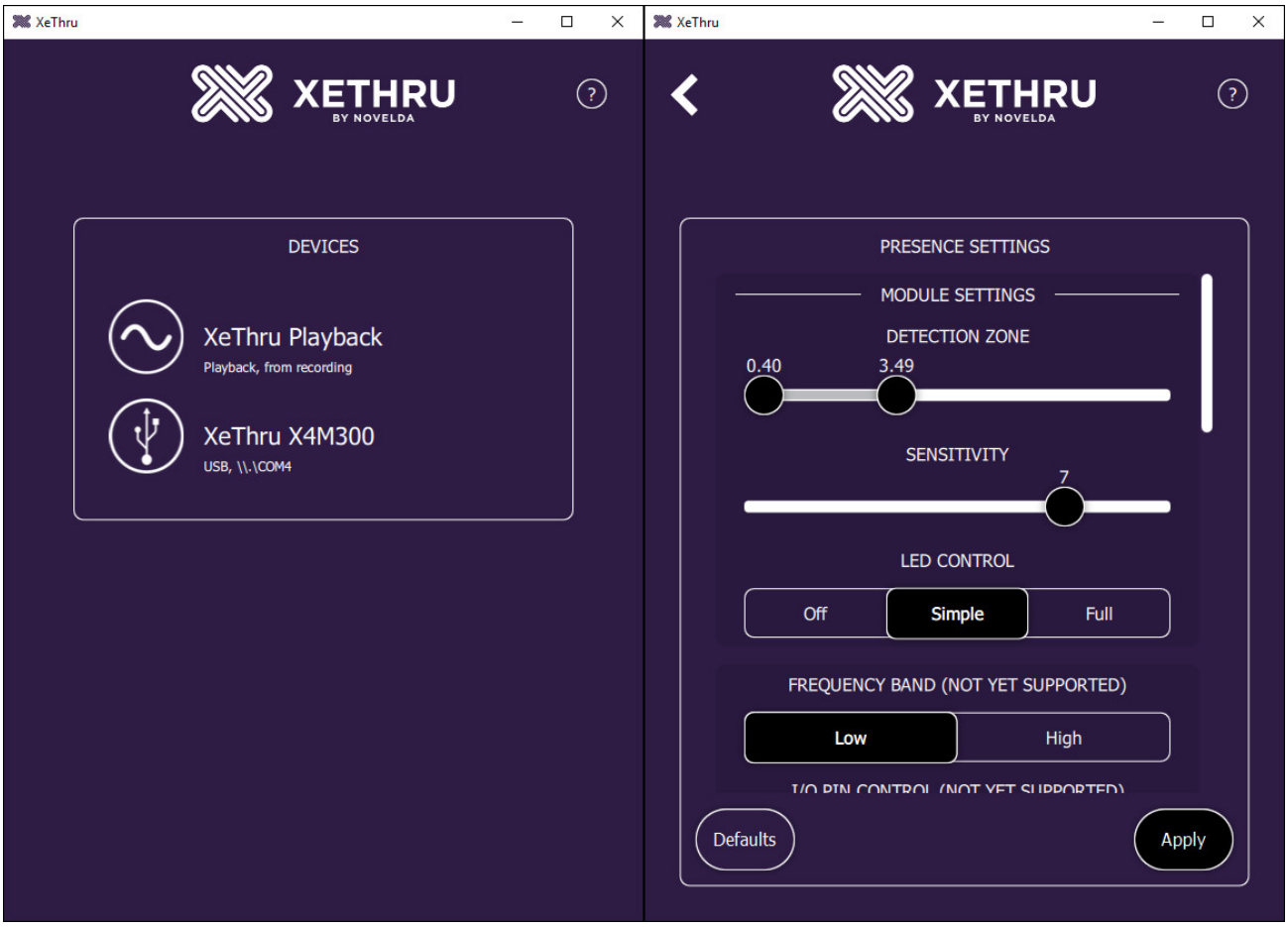


Fig. 3. XeThru Explorer device select and settings menu.

Setup and output:

- After selecting device, XeThru Explorer presents users with a list of settings
- For more details on these settings press the help button
- Spend some time getting used to what the module outputs when there is an object in the detection zone, see Fig. 4 for an example
- Make sure to check out the pulse-Doppler and baseband data output by clicking the buttons labeled PULSE_DOPPLER and BASEBAND
- Try making a recording by clicking the record button
- Playback of recordings is done by clicking "XeThru Playback" after starting XeThru Explorer, see Fig. 3

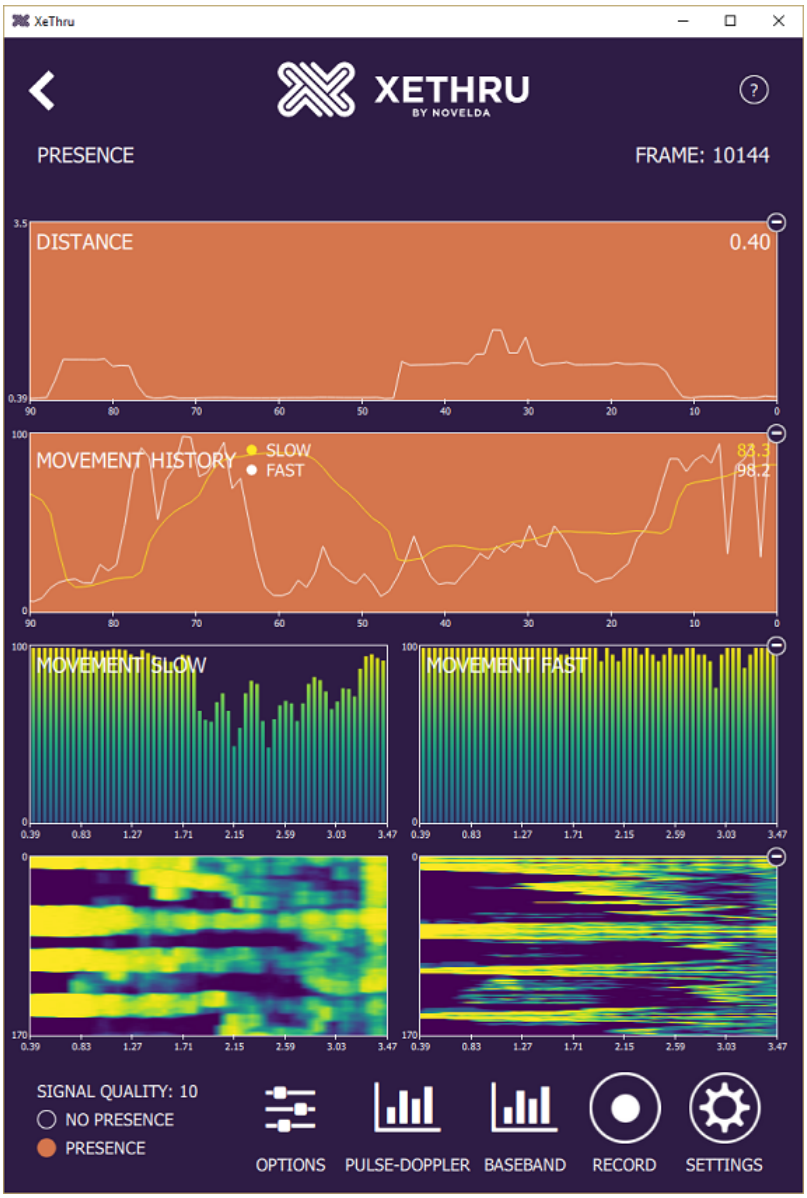


Fig. 4. Example of XeThru Explorer output.

4 Module Connector

Module Connector is a host library that communicates with XeThru modules. Module Connector is a convenient way to start developing applications for XeThru modules. Both the module and Module Connector use the module communication protocol (MCP). MCP is the lowest level communication protocol implementation for the XeThru modules, used on the module firmware and XeThru host components. Module Connector also includes API wrappers for MATLAB, Python and C++ programming languages.

When recording, reading and doing playback of data from the XeThru module it is important to keep three interfaces from the Module Connector in mind:

- Xethru::DataRecorder: The DataRecorder is a high level data recording class. This class generates information and data to generate an exact replica of the bytestream from the module with the DataPlayer class.

- Xethru::DataReader: The DataReader is a high level data reader class. This class uses the files generated by the DataRecorder class as input. It then extracts recorded data into DataRecords. The user can then process the recorded data.
- Xethru::DataPlayer: The DataPlayer is a high level data playback class. This class allows the user to simulate a physical Xethru device from previously recorded data.

To download the Module Connector visit the XeThru resources website . To learn more about Module Connector's functions, variables, constructs etc., unzip the Module Connector package and go to "Moduleconnector\doc\index.html".

The following sections give an overview of how to setup and use the different Module Connector APIs. Generic examples based on the XEP interface is given, for X4M200 and X4M300 functionality, see the examples folders in the Module Connector package.

4.1 C++

4.1.1 Setup C++ API:

For Windows: Start with downloading and installing MinGW-64 and MSYS-64 . Add their /bin directories to the computer PATH.

4.1.2 C++ Example

Now lets try to run XEP_configure_and_run.cpp on the module:

- Compile and link example files by running the makefile found in "/Moduleconnector /examples":
  - > make
- Run the XEP_configure_and_run.exe file from the terminal by writing "XEP_configure_and_run <Device Port>'" where Device Port is the port the module is connected to:
  - > XEP_configure_and_run COM3
- The program will then output raw RF data and can be plotted using a desired plotting package, e.g. Gnuplot
- This program demonstrates how to configure XEP

4.2 Python

4.2.1 Setup python API:

To run examples and python code via the Module Connector it is recommended to download and install Anaconda . Anaconda is a python distribution package which includes most packages needed. If Anaconda is not a option, download additional required packages with the following command:

- > python -m pip install <package>

The following packages are not required for all examples, but most examples include at least one of them. Run the following commands from a command prompt:

- > conda install numpy

- > conda install pyserial
- > conda install configobj
- > conda install matplotlib

Next step is to run the setup.py located in the "Moduleconnector\python<version>" directory. This is done with the command:

- > python setup.py install

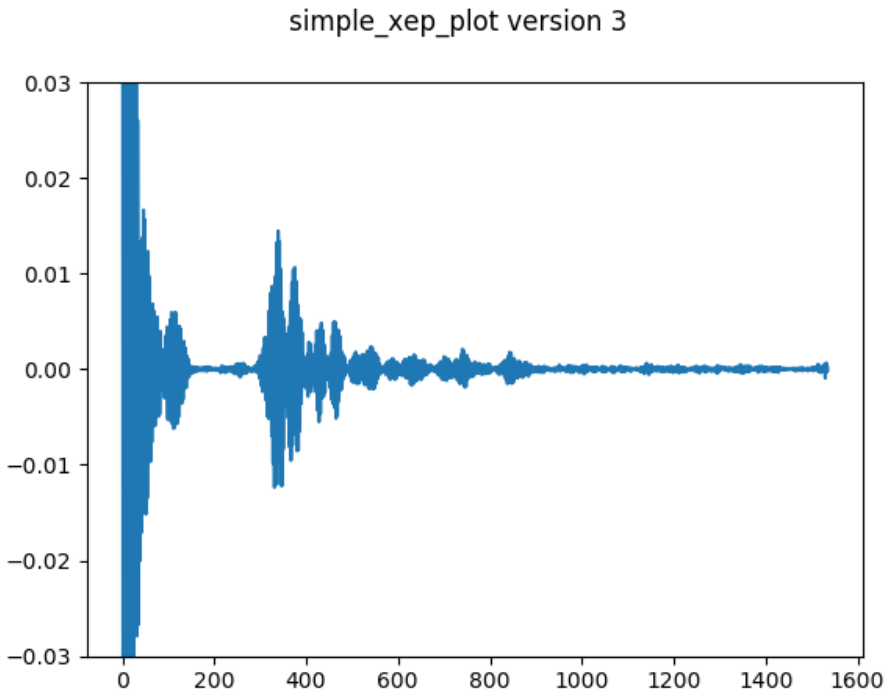
The Python examples are located in the "pymoduleconnector\examples" directory.

- Now cd to “pymoduleconnector\examples”
- When running examples it is important to include the module as argument: python x4m300_presence_simpleoutput.py -d COM3
- Find the module device port with e.g XeThru Explorer
- Not all examples work on all modules, please read the .py example code to find out which module it will work on

4.2.2 Python Example:

Run the example XEP_plot_record_playback.py:

- This example configures the X4 module to XEP, starts raw data acquisition and plots it
- Please read and run the XEP_plot_record_playback.py example
- Try adding -b to change the radar data output from rf to baseband data as shown in Fig. 5
- Try adding -r for recording or -f for playback
- The first strong amplitude is caused by the TX to RX direct coupling



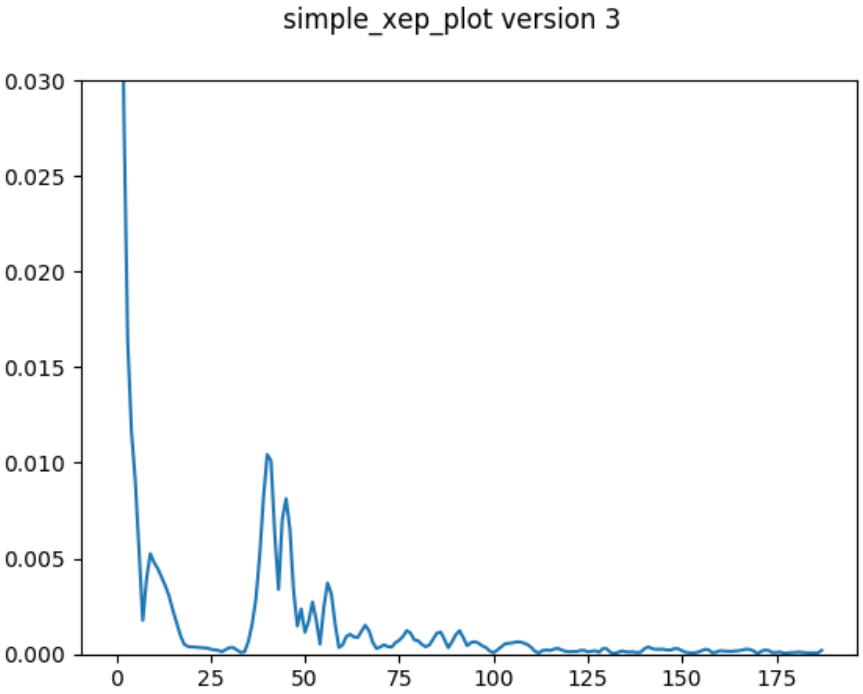


Fig. 5. "XEP_plot_record_playback.py" output. Lower image with baseband enabled.

4.3 Matlab

4.3.1 Setup Matlab API:

Not all Matlab comes with compiler. If the error in Fig. 6 shows up please follow MathWorks installation guide to install compiler.

```
Error using loadlibrary
No supported compiler or SDK was found. You can install the freely available MinGW-w64 C/C++ compiler; see Install MinGW-w64
Compiler. For more options, visit http://www.mathworks.com/support/compilers/R2017a/.

Error in loadlibrary

Error in ModuleConnector.Library/loadlib (line 32)
                [notfound,warnings] =
                loadlibrary(obj.library_name,obj.library_includes,'addheader',obj.library_recording_includes,'addheader',obj.library_datatypes_includes);

Error in ModuleConnector.Library (line 24)
            obj.loadlib();

Error in run_RadarClasses (line 9)
Lib = ModuleConnector.Library;

fx >>
```

Fig. 6: Missing compiler error.

For more documentation on the Matlab wrapper, use the Matlab terminal and cd into the Matlab folder. Then type:

- -> addpath(pwd)
- -> doc ModuleConnector

4.3.2 Matlab Example:

Run the example "XEP_X4_configure_radar.m" on the module:

- Change the variable 'COMPORT' to the device port the sensor is connected to

- Locate the module device port with e.g XeThru Explorer
- Run the example, output is depicted in Fig. 6
- This example demonstrates how to configure the radar chip using XeThru Embedded Platform (XEP)
- The example uses the data float message function to read the raw baseband data
- The recordings from the example are stored in the example folder, try to playback these recordings with the XeThru playback class

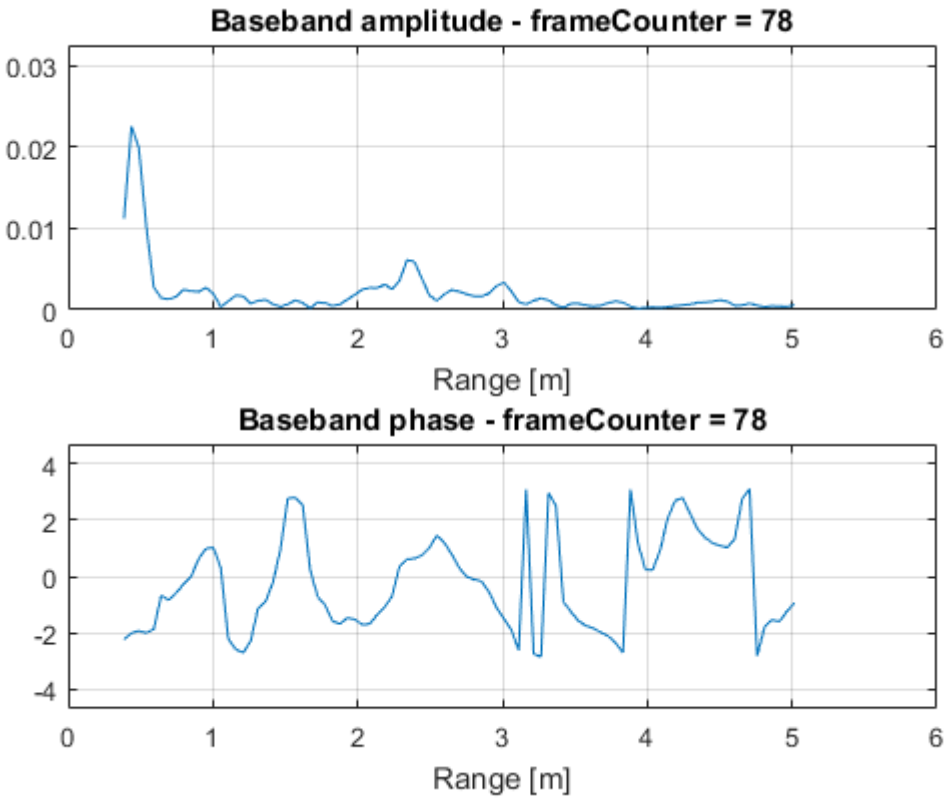


Fig. 7. Output plot of "XEP_X4_configure_radar.m".

5 Module Communication Protocol Wrapper (MCPW)

MCPW offers a source code implementation of sensor communication and comes with examples for different hardware platforms. MCPW uses Module Communication Protocol (MCP), which defines the lowest level communication protocol implementation for the XeThru modules . Typical usage of MCPW is when implementing a host application communicating with a XeThru module and Module Connector is not supported for the host platform.

6 Resources

Software and documentation used in application note:

Description	Link
Host communication software (Module Connector & Module Communication Protocol Wrapper)	https://www.xethru.com/community/resources/categories/host-communication.32/
X4M200 Sleep monitoring Introduction Application Note	https://www.xethru.com/community/resources/x4m200-sleep-monitoring-introduction.110/
XeThru Explorer	https://www.xethru.com/community/resources/categories/xethru-explorer.3/