TI mmWave Radar Set Up

1 Set Up

1.1 Github Download Instructions

This document is designed to provide you with essential information and resources for the course. In this section, you will find instructions on how to download course materials and resources from GitHub.

To get started with the course, you need to download the necessary course materials from our GitHub repository. There are multiple ways to download the repository.

1.1.1 Fork Via Terminal (Useful to Share Code within Groups)

1. Open up terminal (if on Linux or MacOS) or download Git Bash for Windows.
2. Go to the directory where you would like to clone the project repository to (use `cd folder-name`).
3. At the top of the GitHub repository select Fork, and select yourself as the owner. You can also change the name of your project if you would like (useful to differentiate between groups).
4. Click create fork.

More detailed instructions on how to fork and eventually how to clone from your forked repository are available here.

To clone this new forked repository into your computer:

1. Click on the green code button on your repository front page.
2. Copy the SSH link for Cloning.
3. In terminal or Git Bash, type `git clone git@github.com:<owner name>/comm-proj-radar.git`, making sure to do this in the directory you want to save the repository to.

1.1.2 Download Via Zip

1. Visit the link to your GitHub repository.
2. Click on the "Code" button, and select "Download ZIP" to download a zip archive of the repository.
3. Once the download is complete, extract the files to your local machine.

Now you have all the course materials ready for your study and projects.

1.2 Repository Organization

This GitHub repository contains:

- In folder `homework` the Jupyter notebooks you will have to do for homework.
- Notebooks for capturing and reading data. In the files: `capture_data.ipynb` and `visualize_data.ipynb`.
- A script to have live streaming of the radar (located at: `/streaming/realtime_streaming.py`).
- In folder `processing` you will find Python scripts used to process the data. Which are used in radar capture and formatting code.
- In folder `streaming` you will find Python scripts used to have a continuous stream of data coming from the radar and plot it with `realtime_streaming.py`.
- In folder `scripts` you will find everything linked to the setup of the radar.

1.3 TI Software Setup

In this section, you will learn how to set up the required Texas Instruments (TI) software for your course. This software is essential for developing applications related to radar technology. mmWaveStudio is the software that will allow us to format and run the radar boards.

1.3.1 Requirements

- Windows machine with 2 USB ports and an ethernet port.
- XWR1843BOOST board.
- DCA1000EVM.
- One 5V power adapter.
- 2 USB to micro-USB cables.
- 1 Ethernet cable.

If you lack some of the ports, you will have to use one of the adapters that we will provide. If your machine is not running on Windows, you will have to use a laptop that will be at your disposition. This is because MMWave studio software is only supported on Windows.

1.3.2 Installing TI Software

Follow these steps to install the necessary TI software:

1. Install mmWave studio 2.1.1.0 (or whatever is newest and compatible with XWR1843BOOST).
2. Install MATLAB runtime R2015aSP1 (8.5.1) in 32-bit: MATLAB-RUNTIME.
3. Install XDS Emulation Software (32-bit version): XDS.
4. Configure network settings to communicate with the DCA board by setting up the Ethernet network setting to (192.168.33.30 / 255.255.255.0). See example in Fig. 1.
5. Install Python, we recommend using Anaconda to manage Python packages and environments.
6. Install additional required packages via `environment.yaml` or `requirements.txt`.

Figure 1: Ethernet specifications.

2 TI Hardware Setup

The TI hardware setup is also shown via video at this Google Drive link.

To have the radar working, plug the 2 micro USB cables, the ethernet cable to your laptop. The red radar board should be powered with a 5V power supply and the switch on the DCA1000EVM (green board) should be set to 5V_RADAR_IN so that the radar board is powering the green board. Be careful that the micro USB plugged into the DCA1000EVM should be put into RADAR_FTDI, not the FPGA_JTAG.

2.1 Running the Board with LUA Files

Lua files essentially program these radar boards. It is a programming language that we will use to set up the radar's parameters and to capture data. This language is not complicated, and we provide scripts such that you mainly have to change some variables, however, if you want to do more extensive changes to the radar's settings you will have to understand a bit better how the files work.

The provided Lua code automatically runs all the commands required, so there is no need to deal with the mmWave interface other than hitting "Run". A base set of code is given at https://github.com/hailanzs/comm-proj-radar.

Two files are given:

- `1843_config.lua` formats the radar waveform onto the board.
- `1843_record.lua` is used to capture data.

Take the following steps to capture data:

1. **Run** mmWave Studio (`C:\ti\mmwave_studio_02_00_00_02\mmWaveStudio\RunTime\mmWaveStudio.exe`) as administrator mode (being admin might not matter).
2. Open the **device manager** to check the port number of XDS110 Class Application/User UART (`COM [Number]`). The port number is found under XDS110 port in 'device manager', see Fig. 2.

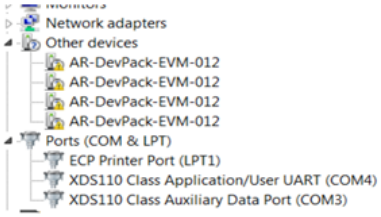

Figure 2: Device manager ports showing radar connections.

3. Ensure that the COM port matches the COM PORT coded in `1843_config.lua`, and verify that the locations of `RADARSS_PATH` and `MASTERSS_PATH` correspond to your paths for `radarssxwr18xx_radarss.bin` and `masterssxwr18xx_masterss.bin` files.

   These files should be located in:

   ```
   C:\ti\mmwave_studio_02_01_01_00\rf_eval_firmware\(radarss)\..
   ```

   ```
   C:\ti\mmwave_studio_02_01_01_00\rf_eval_firmware\(masters)\..
   ```

4. Load the setup file, `1843_config.lua`. To load the configuration, click **Browse** at the bottom, select the file, and click **Run** (Green button at the left bottom).

   This file formats the radar board with the specified chirp parameters and number of frames to capture.

   TIP: In the LuaShell in mmWaveStudio, you can type `help` and the command name (e.g., `help ar1.ChirpConfig`) to get the input description.

5. To capture data, run `1843_record.lua` (modify the path to match your computer's configuration).

6. Once it is working (a green LED at the corner of the DCA board will be blinking, indicating that measurement is in progress), the data is saved to the specified `SAVE_DATA_PATH` in the command:

   ```
   ar1.CaptureCardConfig_StartRecord(SAVE_DATA_PATH, 1)
   ```

   The data is stored as a binary file which you will have to post process in order to get the data into arrays filled with the raw data to then do any sort of signal processing on.

3 Understanding Radar Data

The radar data is streamed out via the Ethernet cable connected to the DCA1000. It saves the file in a file with a `.bin` extension in the location specified in your Lua scripts. This binary data is saved in a format based on the radar board type we have, and everything is interleaved, with complex and real parts separated. In order to get it to a point that is useful for us to manipulate, we need to reformat the data. To do this, we use a package from TI (see `processing/singlechip_raw_data_reader_example.py`). This reformats our binary data into a file with extension `.mat` that we can then load into Python (or Matlab) for visualization and processing.

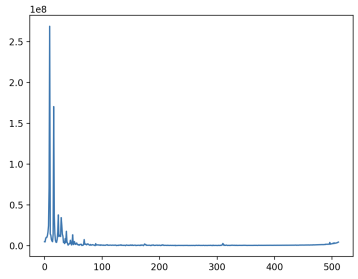

Figure 3: Range plot.

To reformat the data, you must run the following line (with the correct paths, assuming everything was set up according to this documentation):

```
save_adc_data(raw_data_file, directory_of_project,
              name_of_the_captured_data_folder,
    name_of_the_JSON_scripts_exported_without_extension)
```

This code will save the reformatted data into your capture data directory. We recommend that you keep your measured data organized. See `visualize_data.ipynb` for a better understanding of how to reformat the data.

4 Processing Captured Data

4.1 Data Visualization

Once you have reformatted the data you can now process it any way that you like! To first visualize the range FFT directory in line with the radar, perform an FFT along the ADC sample dimension (see `visualize_data.ipynb`) and sum all of the transmitter and receiver channels. We have provided some of our own data for you to play around with in the folder `data`.

The code `visualize_data.ipynb` shows a basic data processing pipeline. It check if not already reformatted, then it reformats the data using the JSON files, and plots the range FFT. You should end up with a plot looking like the one in Fig. 3.

5 Realtime Data Processing

Sometimes it might be useful to stream the data and immediately perform some basic processing on top of it, allowing you to visualize the signals in real-time! We have provided a basic structure for doing this, which requires a slightly different setup compared to capturing the data and processing it offline.

The real-time data processing is located in the folder **realtime-streaming/**.

A high-level overview of the steps required to run real-time streaming:

1. Run the radar so that it continuously captures data (this is done by setting the number of frames in the Lua script to be 0).

2. Close mmWaveStudio **and** terminate the corresponding task in Task Manager (it will have the name `DCA1000EVM CLI`). If this is not done, the data capture may freeze at some point since both the real-time streaming code and mmWaveStudio will attempt to read from the same port simultaneously, causing conflicts.

3. Run `realtime_streaming.py`. A pop-up window will appear with the range FFT plotted in real-time.

As you become more familiar with the code, you can modify it to plot other types of processing in real-time, such as phase over time, Doppler FFT, heatmaps, etc.

At a high level, real-time streaming reads from the Ethernet port connected to the radar. However, in this case, the raw data for each transmitter and receiver pair will be interleaved, requiring some data manipulation.

```
dca = DCA1000()
dca.sensor_config(chirps=3, chirp_loops=1, num_rx=4, num_samples=512)
```

After acquiring the data, the script performs basic processing (such as range FFT) and places it onto a queue that refreshes every few milliseconds. The main function in `realtime_streaming.py` reads the data, updates the plot, and so on. You are welcome to modify the script to include additional plots for real-time visualization, such as phase information!