Technical Documentation for mmWave Radar Pi-Hat

1. Introduction

a) Background

Millimeter-wave (mmWave) sensing offers numerous advantages, making it a key technology for a wide range of applications. Its high distance resolution and sensitivity enable precise object detection and tracking, even in challenging environments such as low visibility or high clutter. These characteristics have positioned mmWave sensors as a preferred choice for automotive, industrial, and consumer applications.

b) Motivation

Although mmWave sensing is a promising technology, the usability of existing mmWave development boards remains a significant challenge. Many of these boards, despite their excellent performance, are difficult to set up and operate. For example:

- Texas Instruments (TI) provides high-performance mmWave development boards, but their usability is hindered by the need for additional components, such as a separate data acquisition board. Furthermore, the system often requires a computer to control the setup, adding complexity and cost. The proprietary control software provided by TI is not open-source, making direct data access and further development cumbersome. The software also suffers from compatibility issues, requiring specific CPUs and operating systems to function properly.
- Infineon offers mmWave development boards that are slightly more user-friendly than TI's. However, they still face similar issues. The boards require a computer for control and also face software compatibility issues. Infineon provides visualization tools, but there is significant margin to improve its visualization effect and usability. In addition, the boards rely on USB connections, which in practice are unstable; slight movements of cables or the boards often result in errors in communication, reducing its robustness.

Overall, current mmWave development boards are plagued by poor usability, limited compatibility, and high costs. These factors make them unsuitable for rapid prototyping or for beginners, such as students or hobbyists, who are just getting started with mWave sensing.

c) Objective

Our goal is to address these challenges by developing a user-friendly mmWave development board that integrates seamlessly with the Raspberry Pi ecosystem. By simplifying the setup process and reducing reliance on external components or computers, we aim to make mmWave sensing accessible to a broader audience. This board is designed to enrich the Raspberry Pi community, enabling more people to explore and utilize this powerful sensing technology with ease.

2. Overview

a) Key Features

Seamless Integration: Designed as a GPIO HAT for easy plug-and-play compatibility with Raspberry Pi.

Raspberry Pi Ecosystem: Operates entirely within the Raspberry Pi platform.

Advanced Radar Technology: Built with the Infineon BGT60TR13C mmWave radar chip.

Comprehensive Data Handling: Enables real-time data acquisition, processing, and visualization.

Flexible Data Management: Supports both local data collection and raw data forwarding via UDP.

Versatile Examples: Includes four usage examples—three focused on real-time visualization and one for offline data processing.

User-Friendly Configuration: Compatible with Infineon Radar Fusion GUI (version 3.5.4) for generating radar configuration files.

b) Specifications

Antenna Configuration: 1 transmitter and 3 receivers.

Operating Frequency: 58–63.5 GHz.

Raspberry Pi Compatibility: Raspberry Pi 4B and Raspberry Pi 5.

Communication Interface: SPI for interaction with the Raspberry Pi.

3. System Architecture

a) Hardware

i. AAA

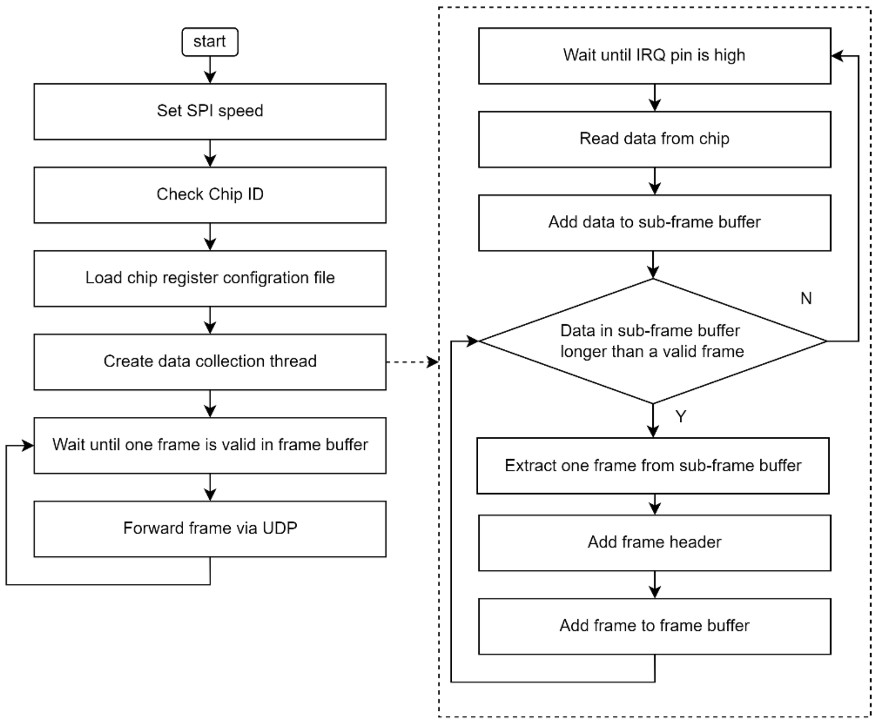
ii. BBB

iii. CCC

b) Software

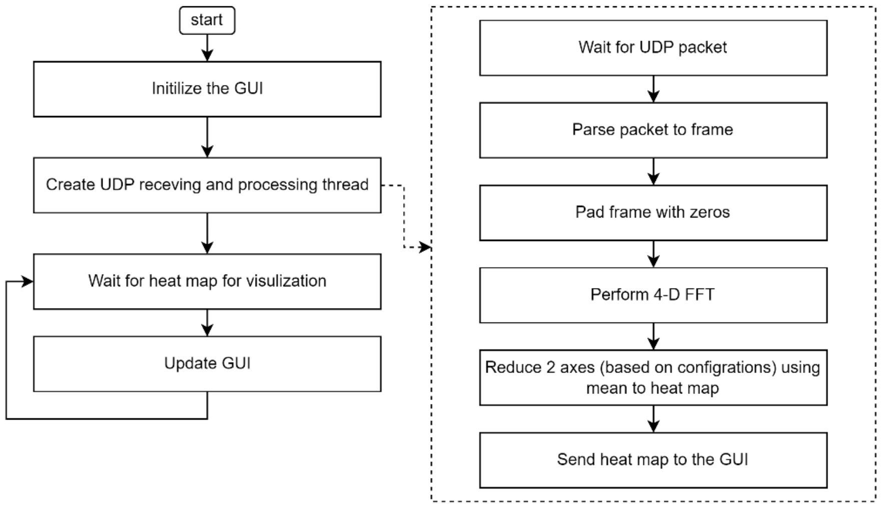
i. Driver and UDP streaming

The main driver is located in “utility/BGT60TR13C.py”, utilizing the “spidev” library for communication and “gpiozero” for GPIO management. The “udp_streaming.py” script interacts with the driver to transmit data via UDP. UDP is chosen for its low latency, continuous data flow, and flexibility in data routing. When the receiver's IP address is set to “127.0.0.1” (localhost), the receiver software operates on the same computer. Alternatively, setting it to the IP address of another PC within the network enables data routing to that device for further processing. This separation of data collection and processing is especially beneficial for real-world applications, such as uploading data to a server over a WiFi network for advanced algorithmic processing.


start
Set SPI speed
Check Chip ID
Load chip register configration file
Create data collection thread
Wait until one frame is valid in frame buffer
Forward frame via UDP
Wait until IRQ pin is high
Read data from chip
Add data to sub-frame buffer
Data in sub-frame buffer longer than a valid frame
N
Y
Extract one frame from sub-frame buffer
Add frame header
Add frame to frame buffer

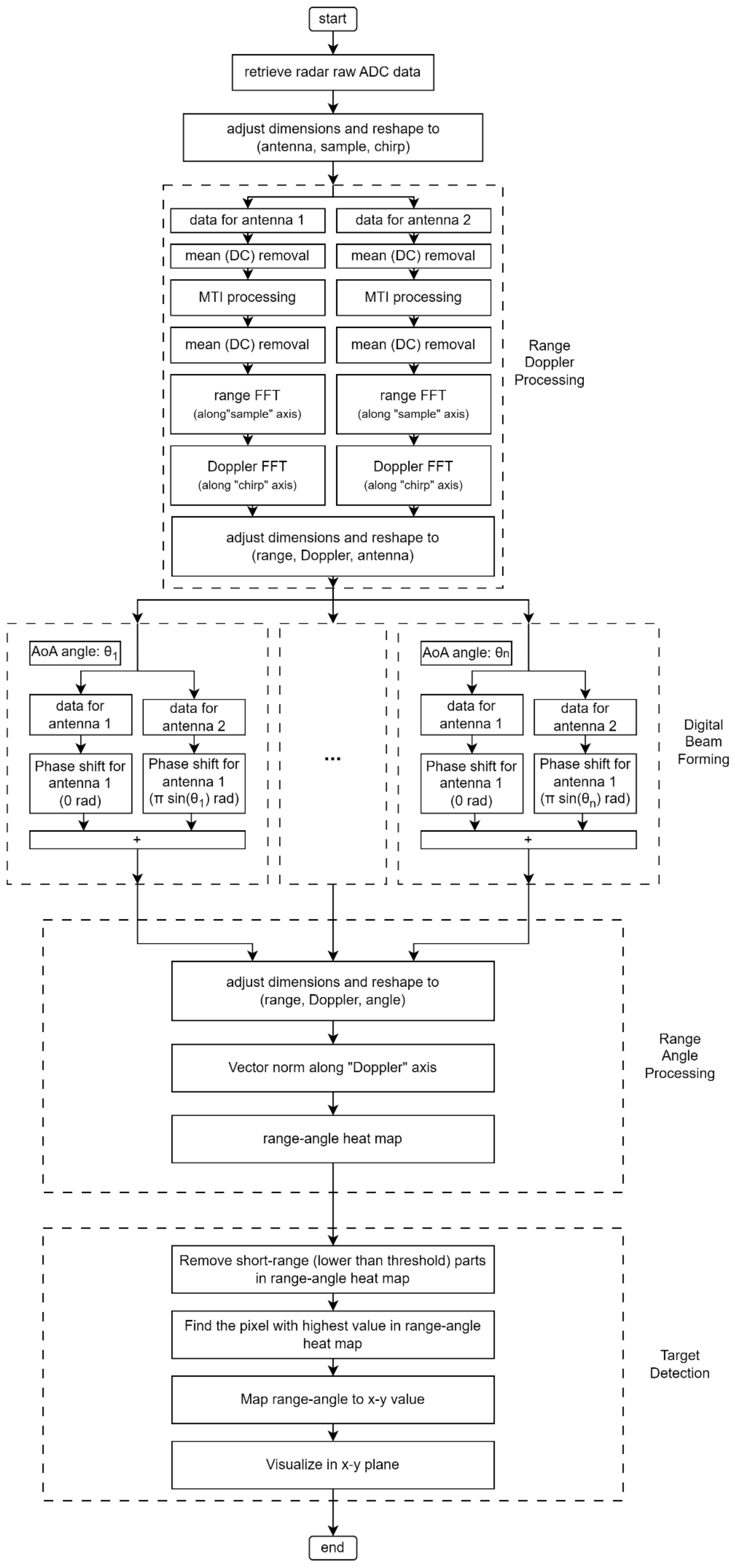
ii. Receiving, processing and visualization

The program receives data from UDP and visualize the mmWave sensing results. It first creates a GUI and then creates another thread to handle incoming data. The thread listens to a UDP port and wait for a packet. Once received, it would parse it to a frame and then pad it with zeros to improve visualization resolution. The main processing is 4-D FFT using "FFTW", which can use all CPU cores. The 4 dimensions are "Doppler", "Range", "Azimuth" and "Elevation", respectively. Finally, for 2-d heat map visualization, 2 axes are reduced using mean. For example, for range-doppler, "Azimuth" and "Elevation" are reduced, leaving "Doppler" and "Range".


start
Initilize the GUI
Create UDP receving and processing thread
Wait for heat map for visulization
Update GUI
Wait for UDP packet
Parse packet to frame
Pad frame with zeros
Perform 4-D FFT
Reduce 2 axes (based on configrations) using mean to heat map
Send heat map to the GUI

iii. Target Tracking

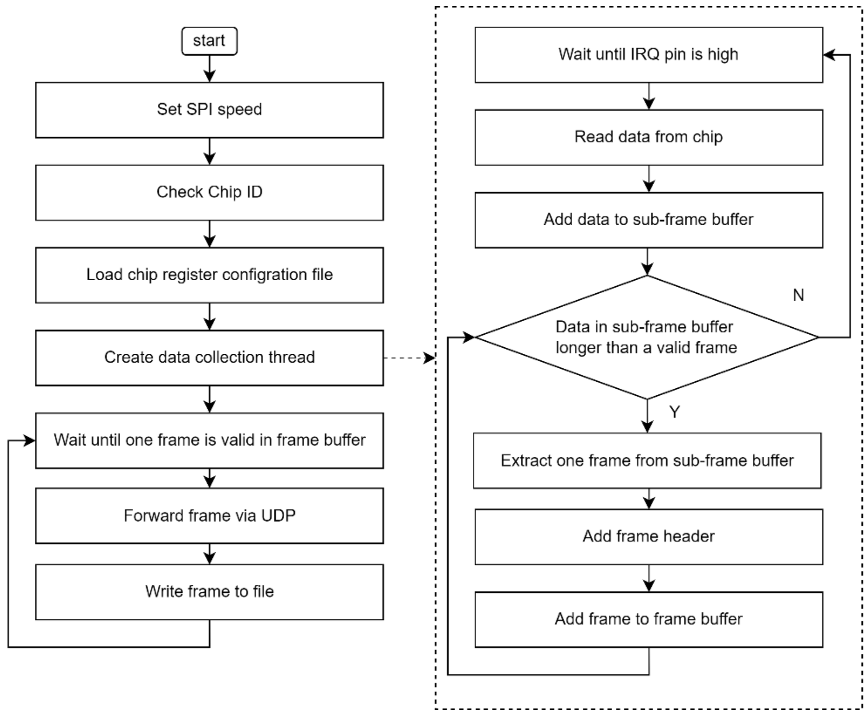
Infineon has provided an example of target tracking in *radar SDK* but not suitable for Raspberry Pi. It is adapted and optimized significantly to fit this pi-hat.


start
retrieve radar raw ADC data
adjust dimensions and reshape to (antenna, sample, chirp)
data for antenna 1
data for antenna 2
mean (DC) removal
MTI processing
mean (DC) removal
range FFT (along"sample" axis)
range FFT (along "sample" axis)
Doppler FFT (along "chirp" axis)
adjust dimensions and reshape to (range, Doppler, antenna)
Range Doppler Processing
AoA angle: θ1
data for antenna 1
data for antenna 2
Phase shift for antenna 1 (0 rad)
Phase shift for antenna 1 (π sin(θ1) rad)
+
...
AoA angle: θn
Phase shift for antenna 1 (π sin(θn) rad)
Digital Beam Forming
adjust dimensions and reshape to (range, Doppler, angle)
Vector norm along "Doppler" axis
range-angle heat map
Range Angle Processing
Remove short-range (lower than threshold) parts in range-angle heat map
Find the pixel with highest value in range-angle heat map
Map range-angle to x-y value
Visualize in x-y plane
Target Detection
end

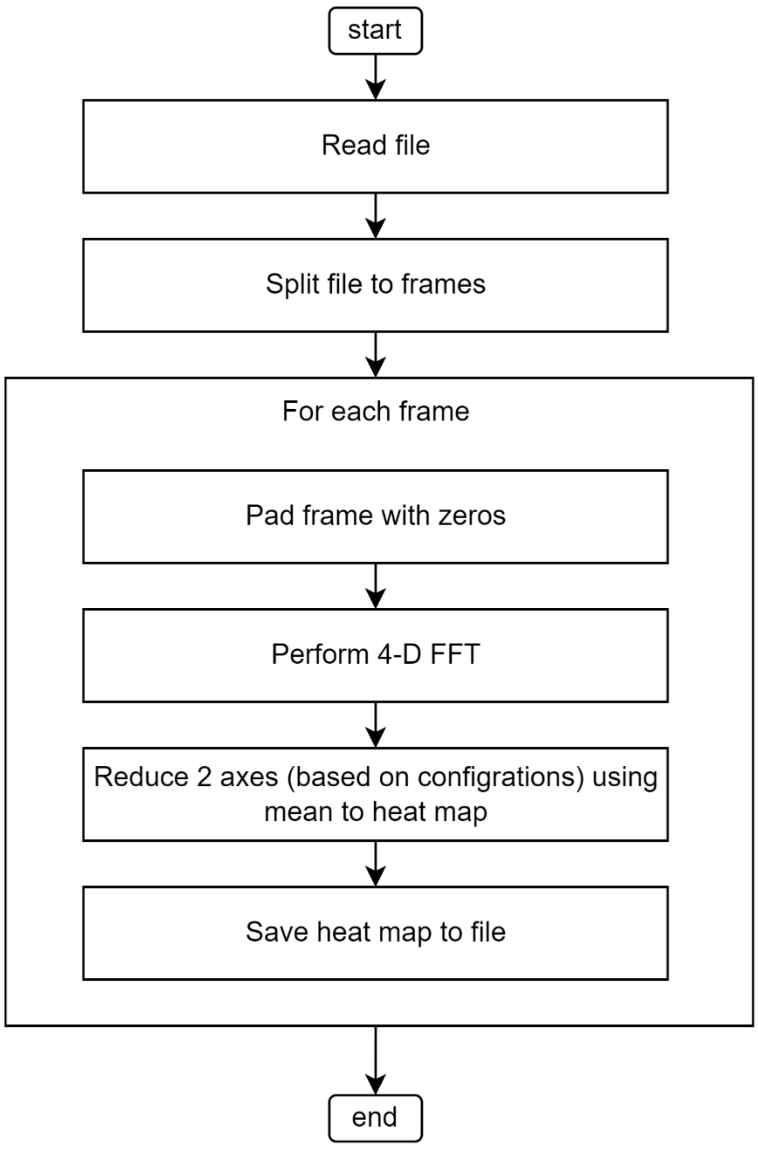
iv. Offline Processing

Offline processing uses almost the same workflow as real-time processing. It just split the data collection and processing to two separate programs.

The data collection resembles UDP streaming by adding write to file.


start
Set SPI speed
Check Chip ID
Load chip register configration file
Create data collection thread
Wait until one frame is valid in frame buffer
Forward frame via UDP
Write frame to file
Wait until IRQ pin is high
Read data from chip
Add data to sub-frame buffer
Data in sub-frame buffer longer than a valid frame
N
Y
Extract one frame from sub-frame buffer
Add frame header
Add frame to frame buffer

For processing, it still uses the 4-D FFT based processing. 2 axes are reduced according to configuration for visualization.



4. User Guide and Examples

a) Set everything up

A Raspberry Pi image is provided with everything set to make it plug and play.

i. Download the image

ii. Unzip the image using “7-zip”

iii. Write the image file to a micro-SD card (e.g. using “Win32DiskImager”)

iv. Boot the Raspberry Pi using the micro-SD card.

v. It will reboot automatically on the first boot.

vi. Set the screen resolution to 1080p.

b) Run GUI-based examples

i. There are 3 executable scripts and one readme file on the screen. Please follow the readme file to run the executable scripts. Basically, double click the executable scripts and then the click “execute in terminal” would start the example.

ii. The project is in “/home/pi/Documents/MMW-HAT-Release”. In each example folder, the python code “run_example_xxxx.py” will start the UDP streaming and visualization in 2 different processes. Run “python run_example_xxxx.py” in terminal/console can also start the example.

iii. In the range-doppler example, in “example_1_range_doppler/visualization_range_doppler.py”, three configurations are provided for different ranges. Uncomment the one you want to use and comment the rest then restart the example.

iv. As there are only 2 horizontal antennas, the azimuth resolution is limited and it may be difficult to identify target directly from the azimuth heat map in example 3.

c) Run offline processing example

   i. In “example_4_offline_processing”, run “python data_collection.py” to collect data. Close the terminal or use “ctrl+c” to terminate the program.

   ii. In “example_4_offline_processing”, run “python offline_processing.py” to process data. Replace the filename in “offline_processing.py” with the new data file before running it. The processed data will be saved in the “example_4_offline_processing/mmw_proc/” folder.

d) Additional information

   i. With the given image, the username and password are given as follows. Username: pi. Password: MMW-HAT

   ii. To set up a Raspberry Pi without using the Image provided,

      1. Enable the SPI

         a) Enable SPI in interface option within “raspi-config” tool.

      2. Run the following command in console to install required packages

```
sudo apt-get update
sudo apt-get install -y python3-numba
sudo apt-get install -y python3-pyqtgraph
sudo apt-get install -y python3-pyfftw
```

      3. Get the code from Github

         a) The code is available at Github: xxxxxxx.xxxxxx.github.com

5. Precautions

a) **Compliance with Local Radio Regulations**
Ensure that the use of the mmWave radar development board complies with local radio frequency management regulations. Avoid interference with other mmWave devices operating in the vicinity by adhering to the designated frequency bands and power limits.

b) **Electrostatic Discharge (ESD) Protection**
Avoid directly touching the components on the development board to prevent damage caused by electrostatic discharge (ESD). Always handle the board with care, and consider using an anti-static wrist strap or working in an ESD-safe environment.

c) **Heat Management and Safety**
During operation, the Raspberry Pi and mmWave radar development board may reach temperatures exceeding 60°C. Avoid touching the devices while they are powered on to prevent burns. Ensure adequate ventilation to mitigate overheating.

d) **Use of Official Accessories**
To ensure stable performance and avoid potential compatibility issues, please use official Raspberry Pi accessories. This includes, but is not limited to, **power adapters**, HDMI cables, keyboards, and mice.