## 

## 

## 

## Software Documentation

Project Name: Millimeter Wave Radar Imaging

Technical Advisor: Dr.Muhammad Faeyz Karim

Sponsor Name: Dr. Logan Porter

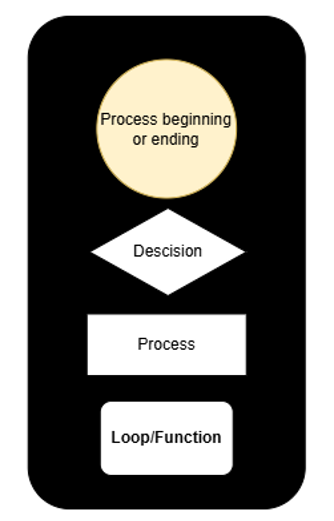
Team Name: Crani-Detect

Revision: 5.0

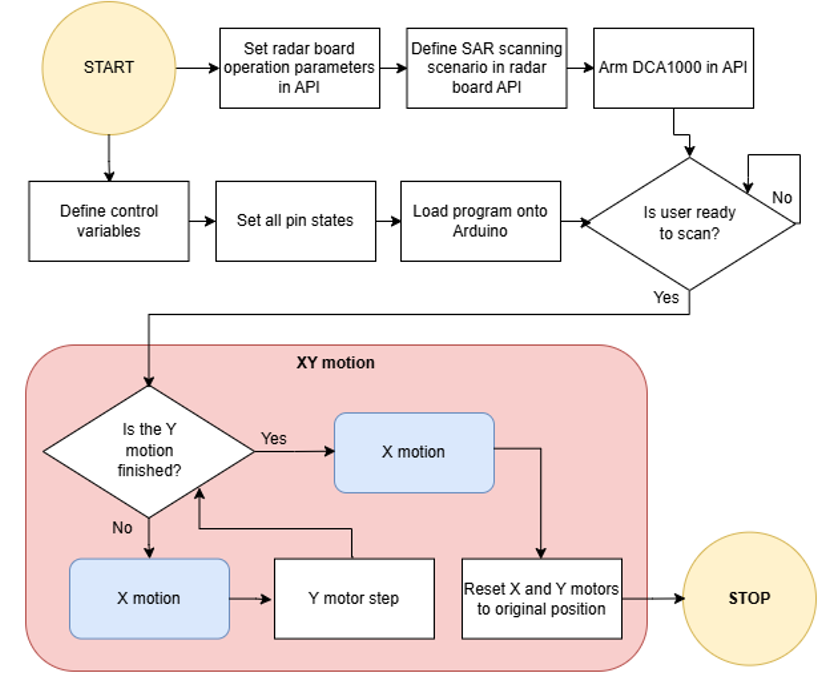
Date: 11/25/2024

## 

The software for our mmWave radar imaging scanner can be characterized by two main sections: scanner movement and image production. This software flowchart outlines the main processes and decision points involved in producing images using a MIMO-SAR (Multiple Input Multiple Output - Synthetic Aperture Radar) style scanner. The key for this chart can be found above in Figure A below.



*Figure A: Software Flow Chart Key*



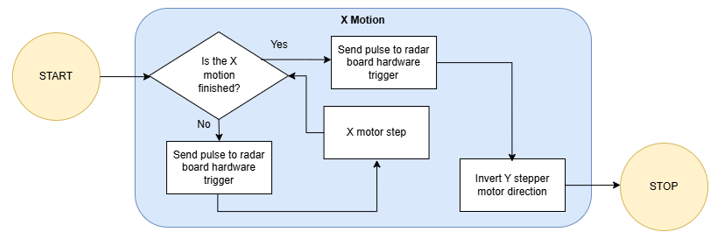
*Figure B: Software Flow Chart*

The entire software process is a result of the combined utilization of three software beds: mmWave Studio, Arduino IDE, and Matlab. The process shown in Figure B characterizes the scanner movement as well as data acquisition, which is done using a combination of both mmWave Studio and Arduino IDE. The API provided by Texas Instruments (TI mmWave Studio) is utilized at the very beginning of this process in order to configure chirp profiles and frame configurations, ensuring the radar operates according to the desired specifications during the scan. Once the radar board is configured, the user will arm the radar board for sensing through mmWave Studio. Data acquisition is also managed through mmWaveStudio, where the software works in tandem with the DCA1000 board to capture raw ADC data from the radar sensors. This raw data is stored in binary format on a local PC, ready for further processing.

Arduino IDE will be utilized exclusively to perform the process defined in the XY motion block which is colored in red. This loop has the purpose of controlling the movement of the scanning system and signaling the radar sensing board to begin capturing data. After updating the control variables to match the desired SAR scanning scenario, the program will be loaded onto an Arduino UNO. After this point, once the UNO is powered on the motors will begin to move and execute the defined SAR scan after a short delay. Once the scan is completed and data is captured, MATLAB is utilized for processing and image reconstruction. The raw ADC data from mmWave Studio is imported into MATLAB, where the data is converted into a 3-D data cube. With this information a Fourier-based image reconstruction algorithm can be utilized to produce high-resolution 2D or 3D images.

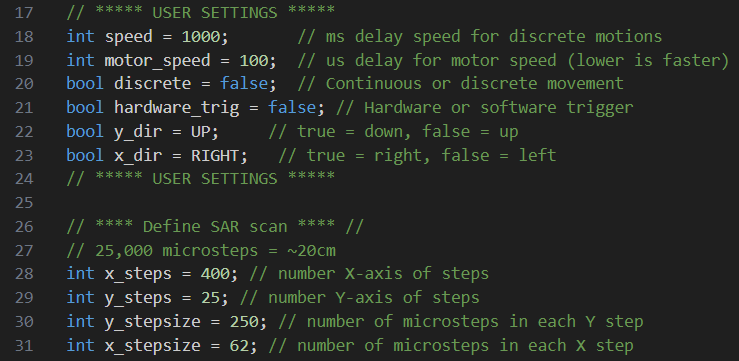
### Scanner control

The process for scanner movement starts at the lowest level with the X motion control shown in Figure C. In each iteration of this process (or each unique X-axis location) the UNO will send a pulse to the radar board, signaling it to begin a frame capture. After this the scanner will move a user-specified distance (defined in the movement control variables) in the X direction by pulsing the X axis stepper motor driver. During this process the DCA1000 acquires the scan data (I/Q data) from the radar board and sends this information to the local PC via ethernet. This will continue until the last iteration, in which the scanner will capture a frame at the last unique X-axis position and invert the direction of the X stepper motor. This is vital so that the scanner is able to quickly cover the entire synthetic aperture by moving in a SAR scanning pattern similar to the one shown in Figure E.



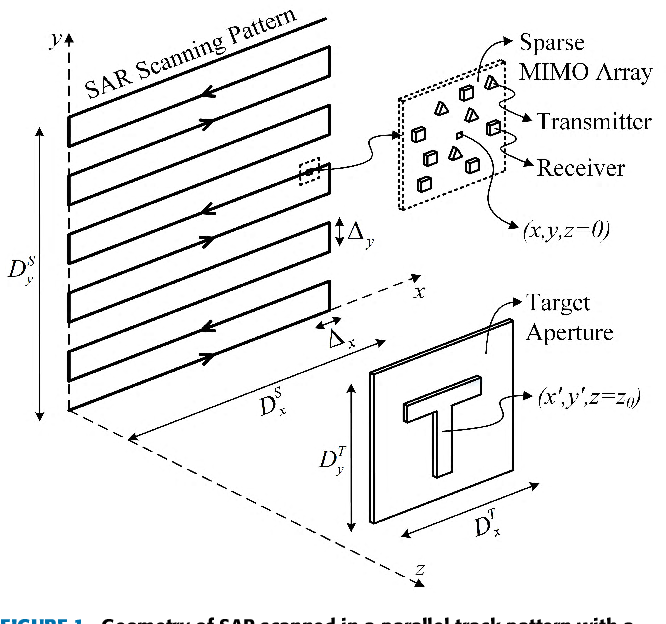
*Figure C: X Motion Software Flow Chart*

After the completion of all desired X-axis steps, the scanner will move a user-specified distance in the Y direction by pulsing the Y-axis stepper motor driver. After this, the X-axis scan will be repeated until the entire target aperture is scanned, characterizing the logic for the entire XY motion shown in Figure X. Once the XY motion is finished and the scan completed, the X and Y direction stepper motors will move the radar board back to its original starting position.

**

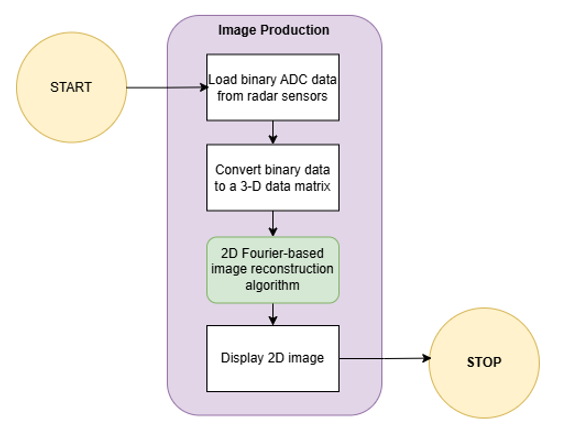
*Figure D: Snapshot of Arduino Control Variables*

A snapshot of the modifiable control variables in the Arduino code is shown in Figure D. The user may set a delay between discrete motions as well as the microsecond delay for the microstep pulses sent to the stepper motor driver. Boolean values may be set to enable discrete steps as well as hardware trigger pulsing between X-axis steps. The user can additionally set the initial directions for the X and Y motions for an XY SAR scanning motion.

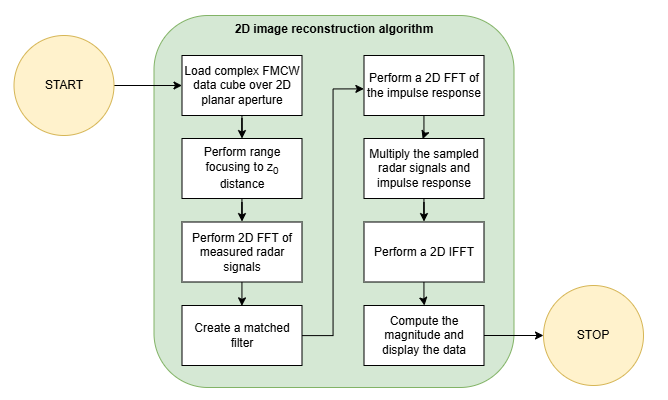


*Figure E: Example of SAR Scanning Pattern Using MIMO scanner*

### Image Production



*Figure F: 2D Image Production Flow Chart*



*Figure G: 2D Image Processing Algorithm Flow Chart*

This section will discuss the SAR Imaging package developed by Muhammet Emin Yanik and Prof. Murat Torlak which we are utilizing to create 2D images in MATLAB. The image production process begins with the optional preprocessing of raw ADC data captured from radar sensors. In our experiments we did not include any preprocessing of our raw data, however calibration and synchronization procedures where gain and phase mismatches in the MIMO radar elements are corrected could improve the image production quality. The user should note as well that the radar data must be synchronized with the mechanical scanner’s movement to maintain accurate spatial sampling. This can be achieved using the hardware trigger functionality of the radar board which is mentioned briefly in the scanner control section.

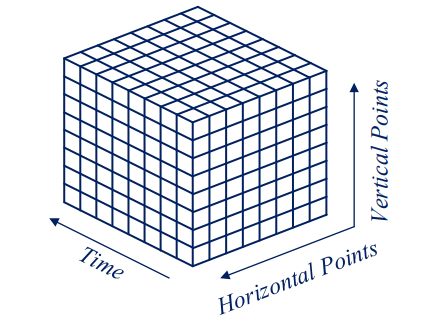
The image reconstruction process then proceeds by converting the data into a 3D cube which is represented in Figure H. This information stored in this cube is the number of samples by the number of vertical points by the number of horizontal points. The prior user-defined target distance, x sampling distance, and y sampling distance are critical to the success of the next part of this process. After coinciding the target and aperture coordinates, the 2-D reflectivity image can be formulated as



where FT2D and FT2D-1 denote 2-D Fourier and inverse Fourier transform operations over the 𝑥 − 𝑦 plane, 𝑓(𝑥, 𝑦) is 2-D target reflectivity function, 𝑠(𝑥, 𝑦) is the measured radar signals, and ℎ(𝑥, 𝑦) is the impulse response or the point spread function of the imaging system calculated for each (𝑥, 𝑦) measurement point as

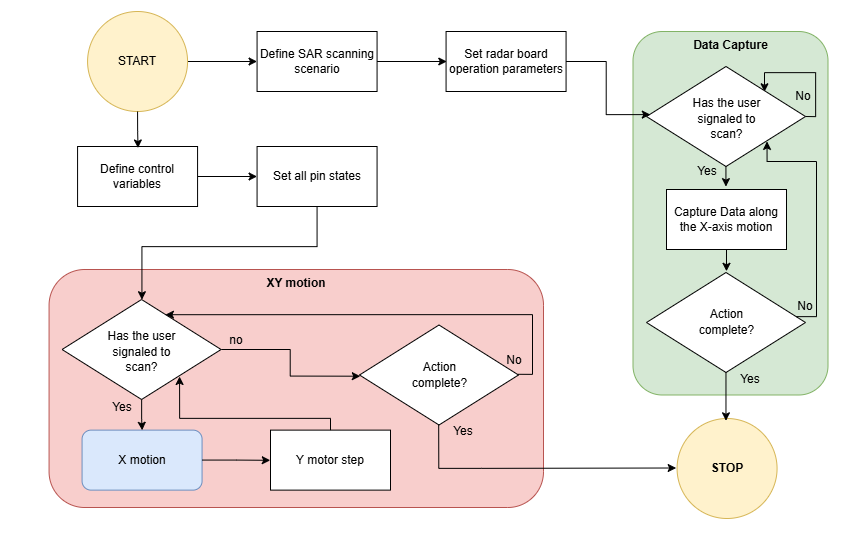


Using the built in MATLAB 2-D FFT function, the reconstruction algorithm can be summarized as shown in the flow chart represented in Figure G.



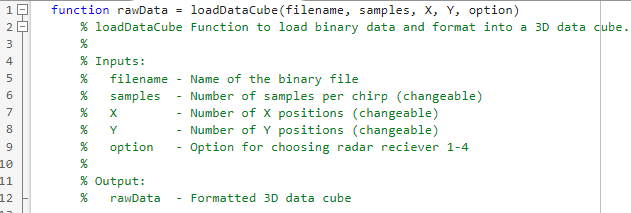
*Figure H: 3D Data Cube Utilized for Image Processing*

Due to the fact that our team could not utilize the functionality of the hardware trigger, the software design previously described could not be used effectively. We discovered this is because the synchronization of scanner movement and data acquisition is imperative for creating useful images from SAR scans. In order to account for improved synchronization while maintaining as much of the software design as possible we came up with the following solution.



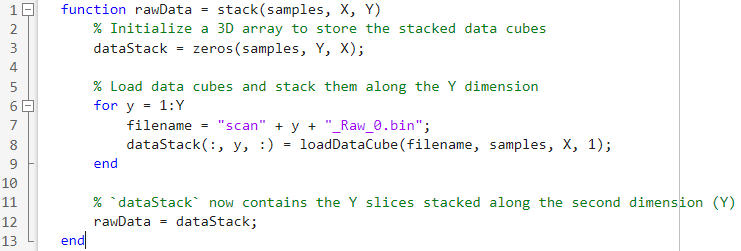
*Figure I: Software flow chart*

The flow chart shown in figure I characterizes the actual software process we utilized in our experiments to create our 2D images. This process requires the user to manually activate the scan on the radar board at the same time as the Arduino is manually activated. To do this we created a simple UI so that the user can communicate with the arduino through serial communications. This way a user may send a signal from their PC to the arduino to start a scan movement. With the mmWave Studio API, users may also send a signal from their PC to the radar board using the software trigger feature to start the scan data acquisition. By utilizing two PCs (one on the Arduino and the other on the IWR1443.DCA1000), the enter button could be hit on both PCs at the same time to ensure that spatial data is being sampled correctly. In this new version of the program, the XY motion consists of an X motion back and forth of the user-defined distance. This X motion and API software trigger can be invoked infinitely until the user is finished scanning their target.

**

*Figure J: Custom MATLAB Function for Loading Raw Data*

Our team created two custom MATLAB functions to process the raw data and to stack the Individual scans to create a data cube as shown in Figure H. If the program previously described is utilized using the hardware trigger only the loadDataCube function shown in Figure J needs to be invoked. This program reads the binary data and arranges the data such that every other Y slice is indexed in reverse order. This is done so that the spatial data matches the SAR scanning pattern shown in Figure E. The user may also choose 1 through 4 depending on which receivers are configured using the API.



*Figure K: Custom MATLAB Function for Scan Stacking*

In figure K, the stack function can be invoked to stack multiple individual scans as we did in our scans. To use this function the user must name each file as scan with an indexing integer (ex: scan1, scan2, scan3… scanN) when they are capturing data in the radar API. This way scan data can quickly be transferred from mmWave Studio to MATLAB where it can be processed immediately.

Files needed for software:

Mmwave studio

<https://www.ti.com/tool/MMWAVE-STUDIO>

matlab runtime engine (must download version 8.5.1 32 bit)

<https://www.mathworks.com/products/compiler/matlab-runtime.html>

Mmwave sdk

<https://www.ti.com/tool/MMWAVE-SDK>

xds emulation software

<https://software-dl.ti.com/ccs/esd/documents/xdsdebugprobes/emu_xds_software_package_download.html>

Matlab (newest version should work)

<https://www.mathworks.com/products/matlab.html>

Arduino IDE

<https://www.arduino.cc/en/software>