



Chirp: Distributed Radar Network

Our Team



Jason Wang

MQTT Development



Andrey Otvagin

Jetson/Node Development



Vihan Jayaraman

Wireless Infrastructure



Oviya Seeniraj

Time Sync & Real-Time
Adaption



William Ni

Integration

Motivation: RF Sensing = Seeing without Light

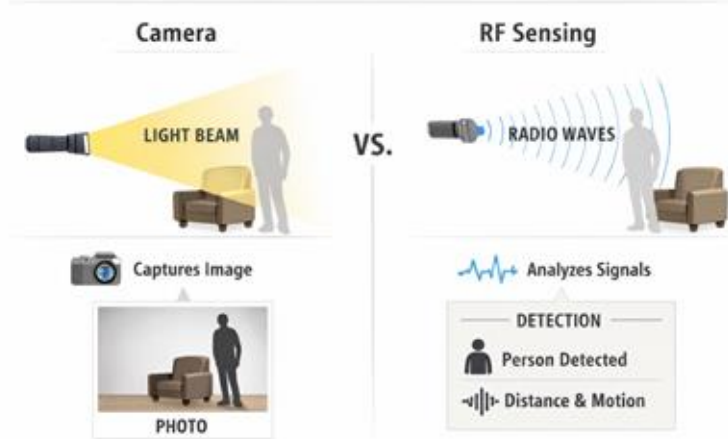


Fig. Camera vs RF Sensing

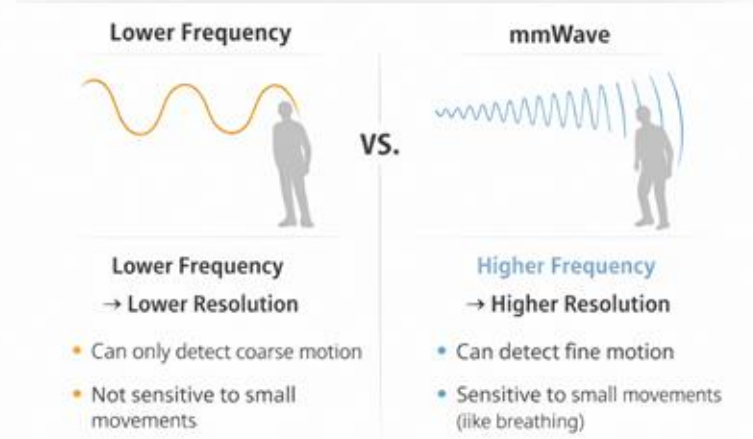
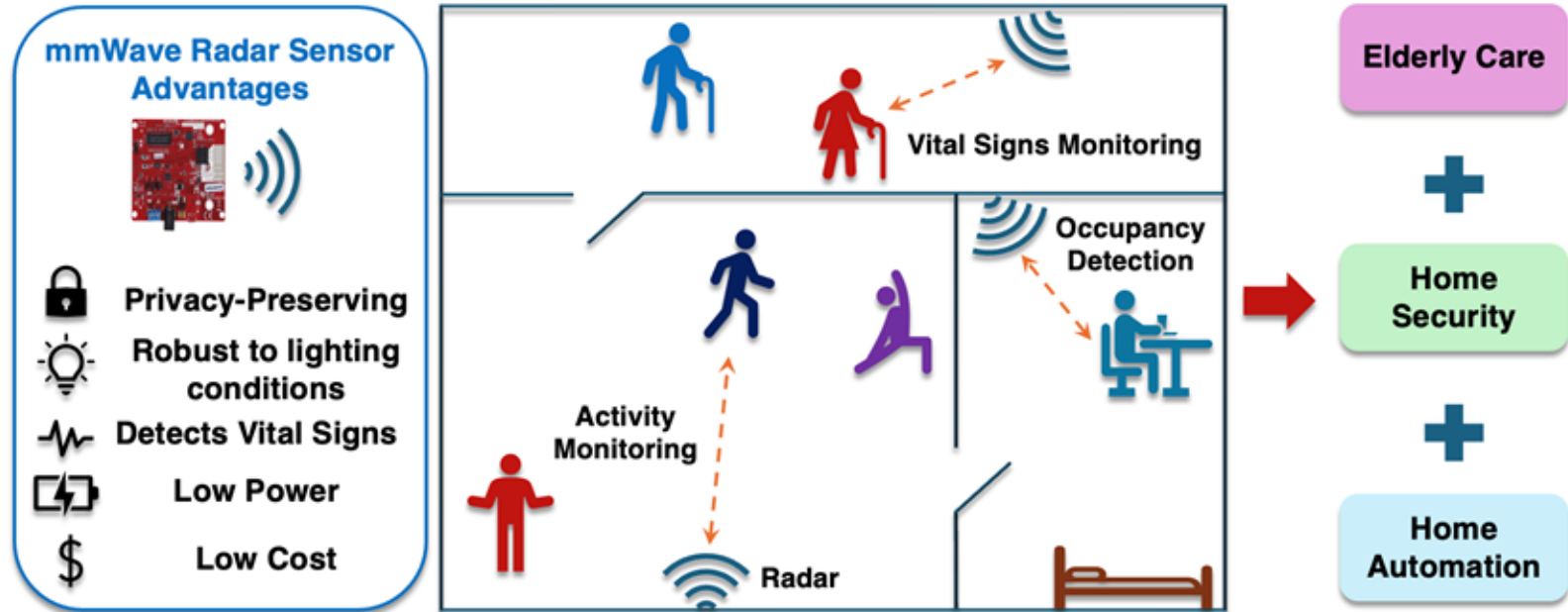


Fig. Sensing in mmWave vs Lower Frequency bands

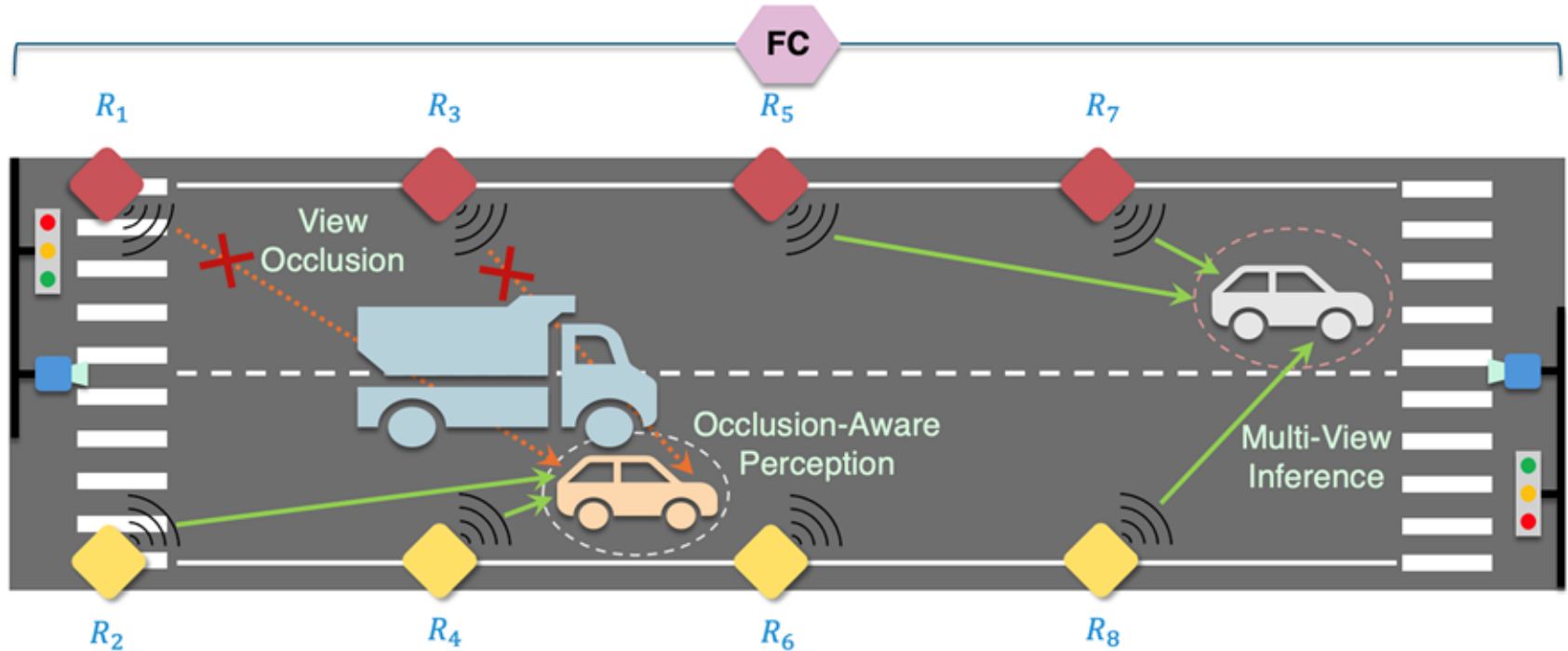
Motivation: mmWave Applications

Smart Homes enabled by mmWave Radar Sensors



Motivation: mmWave Applications

Collaborative Sensing on Roadside Deployments

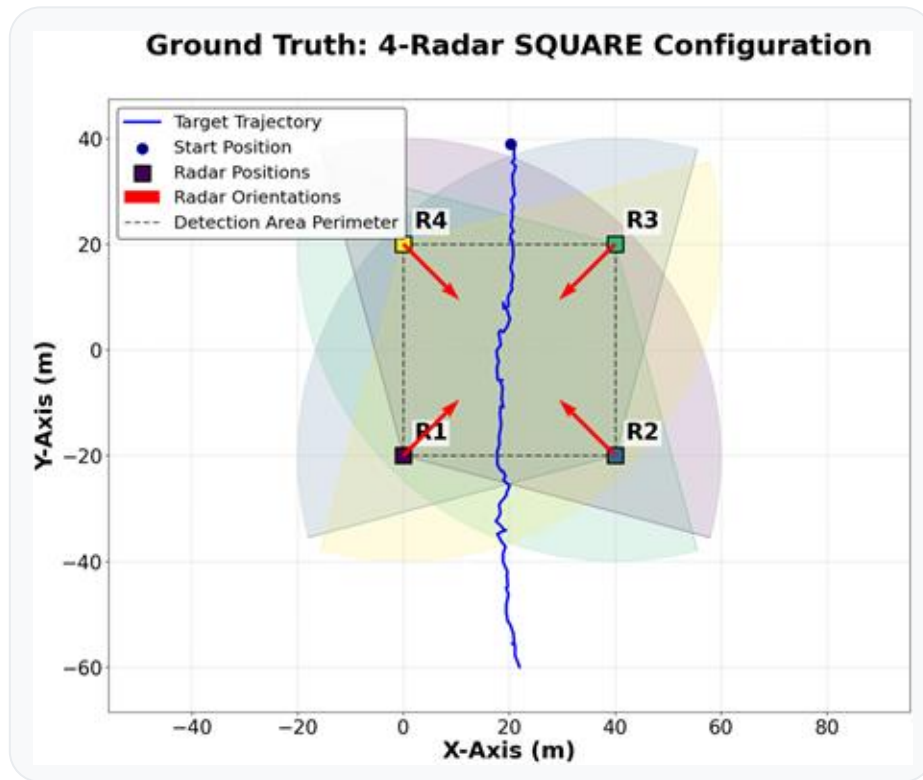


Goal & Purpose

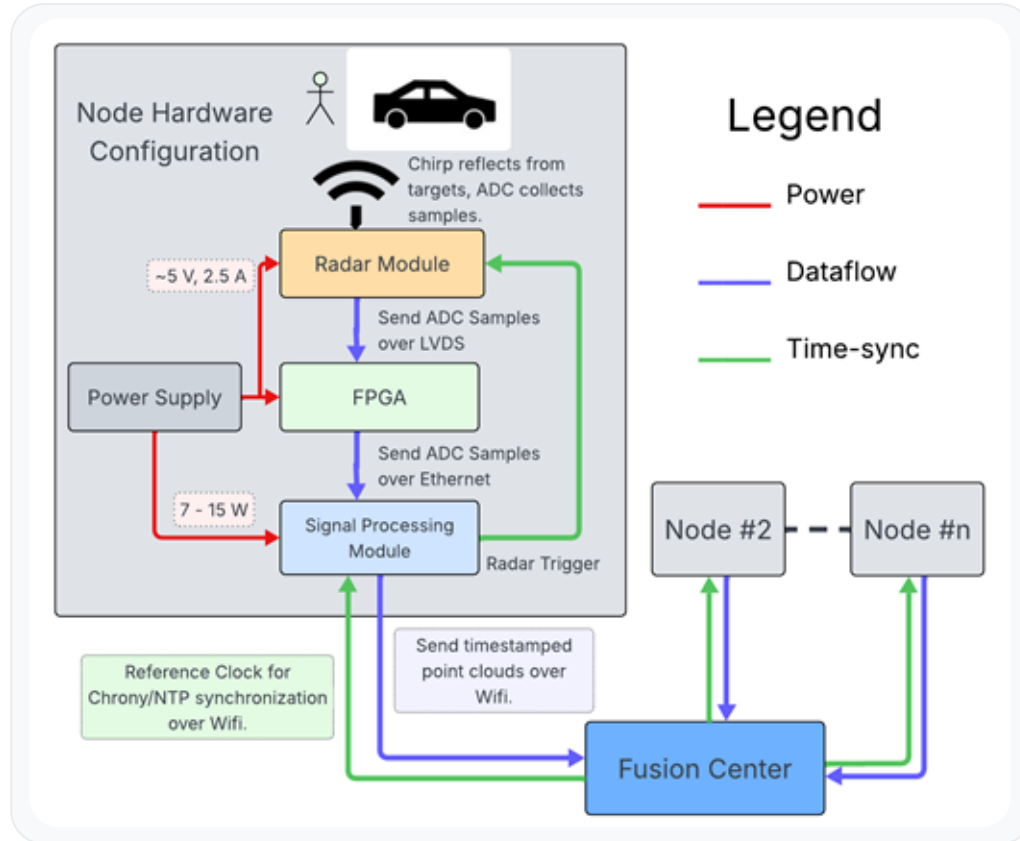
Demonstrate *Spatial Calibration* for distributed radar nodes

Achieved through 3 main steps:

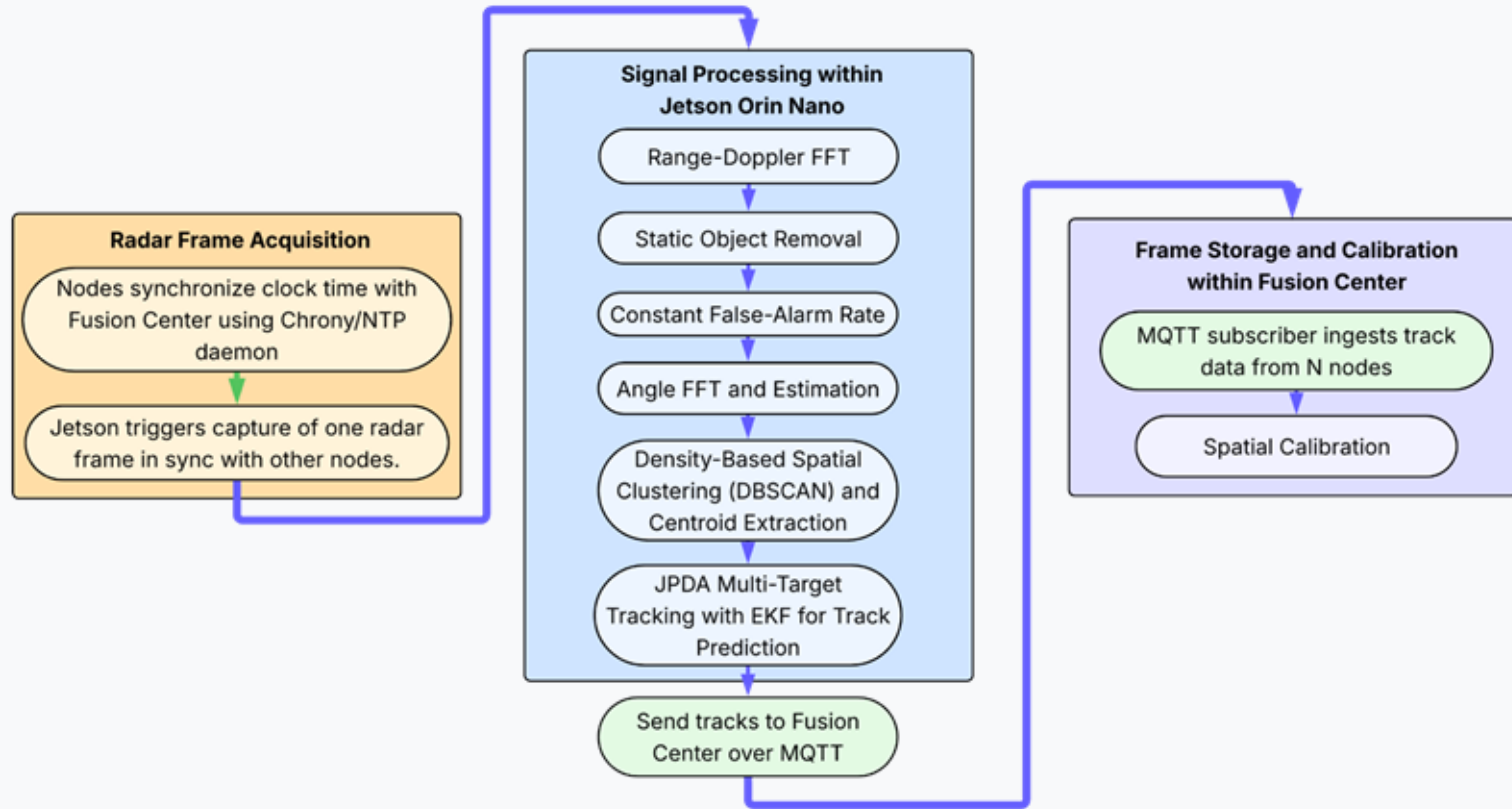
- Time Synchronization
- Reference Tracking
- Relative Pose Estimation



Block Diagram



Dataflow Overview



Dataflow Overview: Acquiring Radar Frames

Step 1

Radar Frame Acquisition

Nodes synchronize clock time with Fusion Center using Chrony/NTP daemon

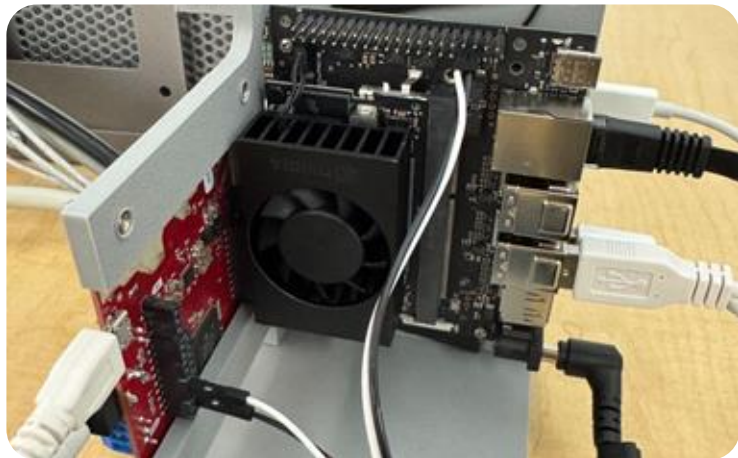


Jetson triggers capture of one radar frame in sync with other nodes.

Time Synchronization through Chrony NTP Daemon

Precise Synchronization

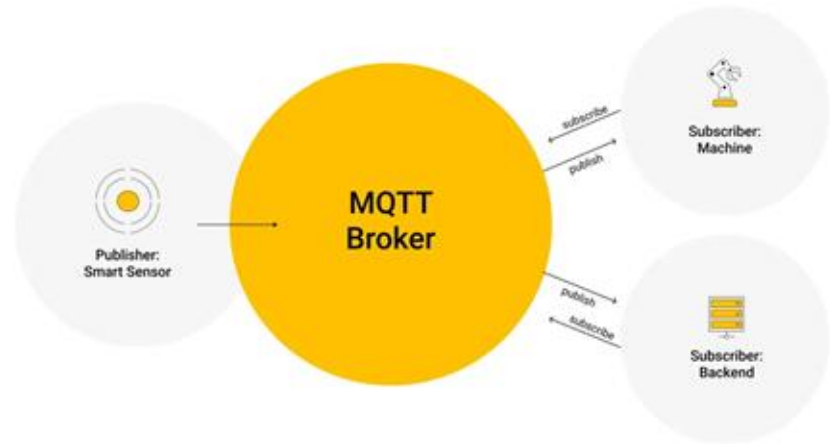
- Spatial Calibration requires that nodes capture frames at the same time
- Synchronize Nodes using “chrony”
- Setup Radars to capture a frame via a GPIO pulse
- Chrony NTP server served off of Fusion Center
- **~ 3 ms accuracy**



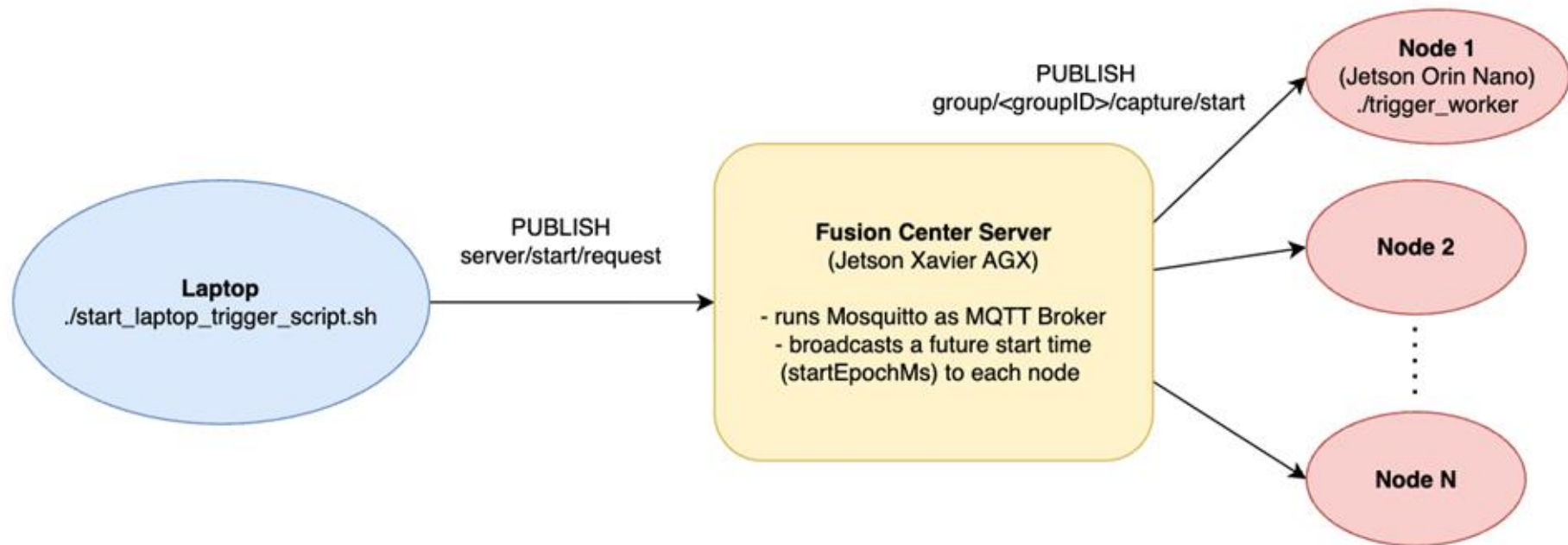
MQTT Overview for Communication between Nodes

What is MQTT?

- **Message Queueing Telemetry Transport:** A lightweight, common protocol in IoT.
- **Pub/Sub Model:** Nodes communicate by publishing or subscribing to specific “topics”.
- **Centralized Routing:** The MQTT Broker handles all message distribution between nodes.

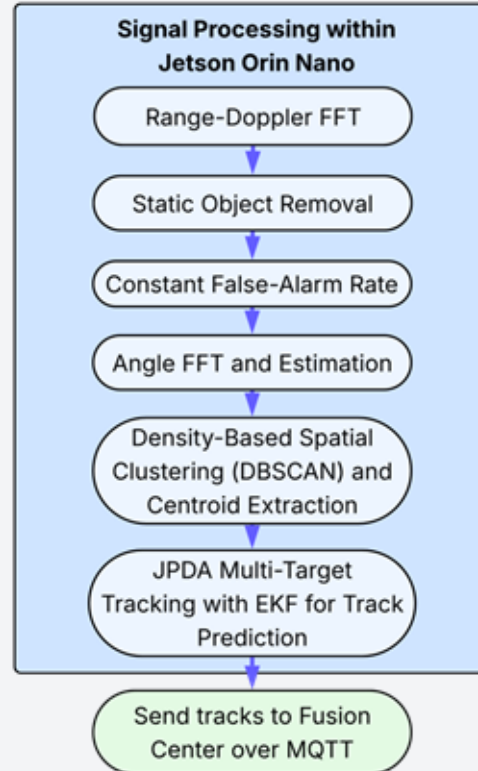


MQTT for Synchronized Trigger from Laptop



Dataflow Overview: Processing Radar Data

Step 2



Radar Signal Processing Pipeline Purpose

CORE OBJECTIVE

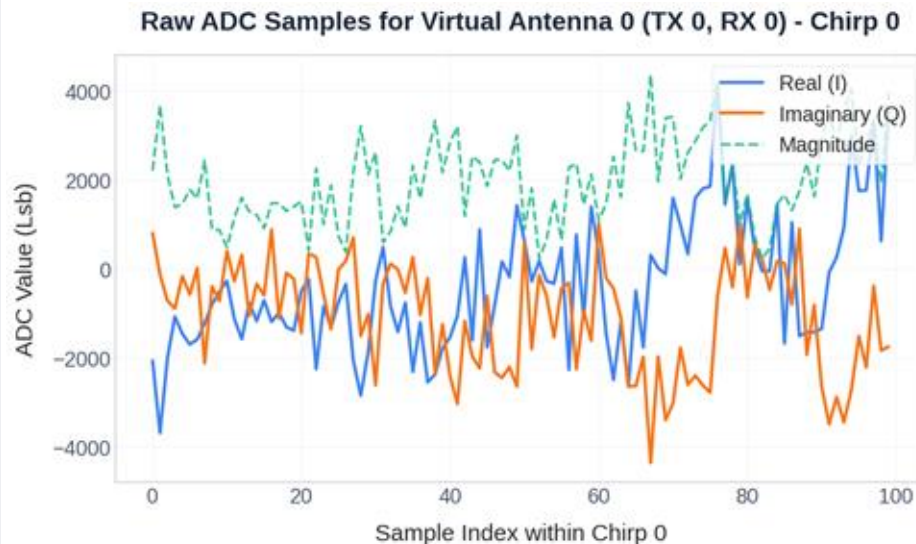
Stream of Raw ADC samples →
Object Location & Velocity

Example Output: Human 1

Position: x: 1 m, y: 3 m

Velocity: v_x: 1 m/s, v_y: 0.5 m/s

Timestamp: 12h:34m:56s



Pipeline: Raw Data to Range Doppler Map

STEP 01

Input Stream

Data provided as a stream of raw ADC samples

STEP 02

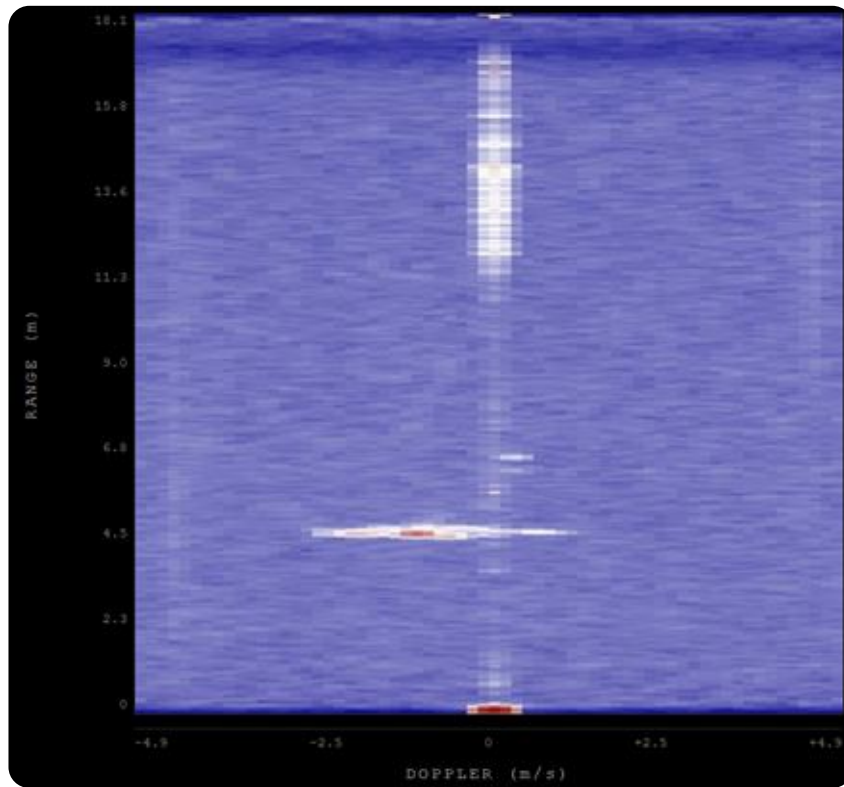
Processing

2D FFT to generate a Range Doppler Map

STEP 03

Static Filtering

Subtract power of stationary background to isolate moving targets



Visualizes magnitude per Range-Doppler bin

Pipeline: CFAR for Points of Interest

STEP 04

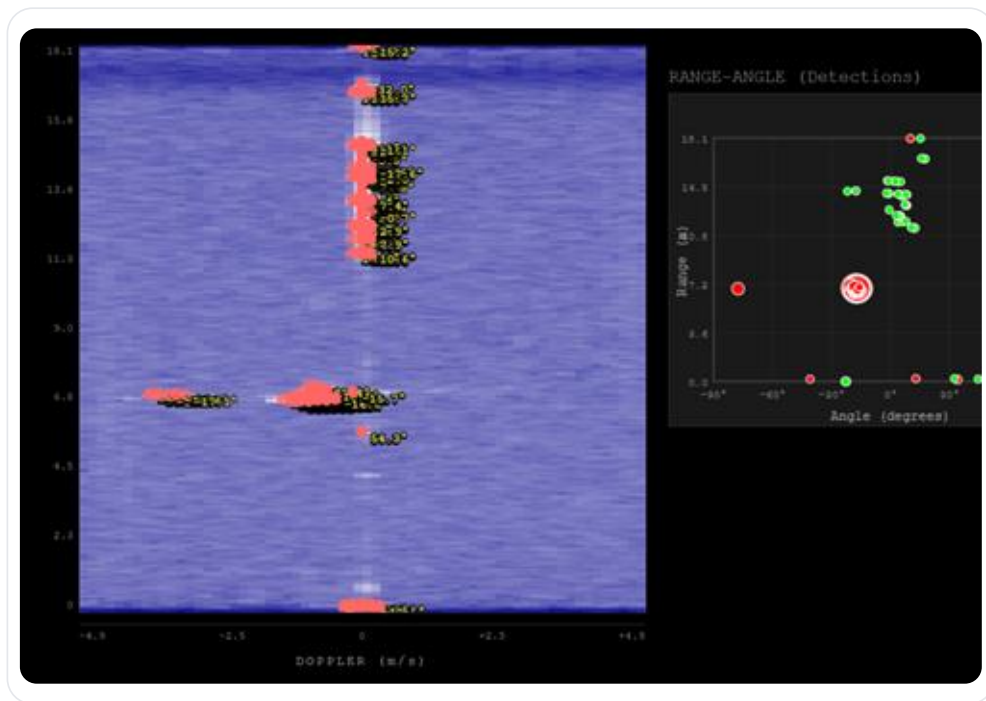
Detection

CFAR to identify potential points of interest

STEP 05

Angle Estimation

Calculate the angles of each of those points of interest

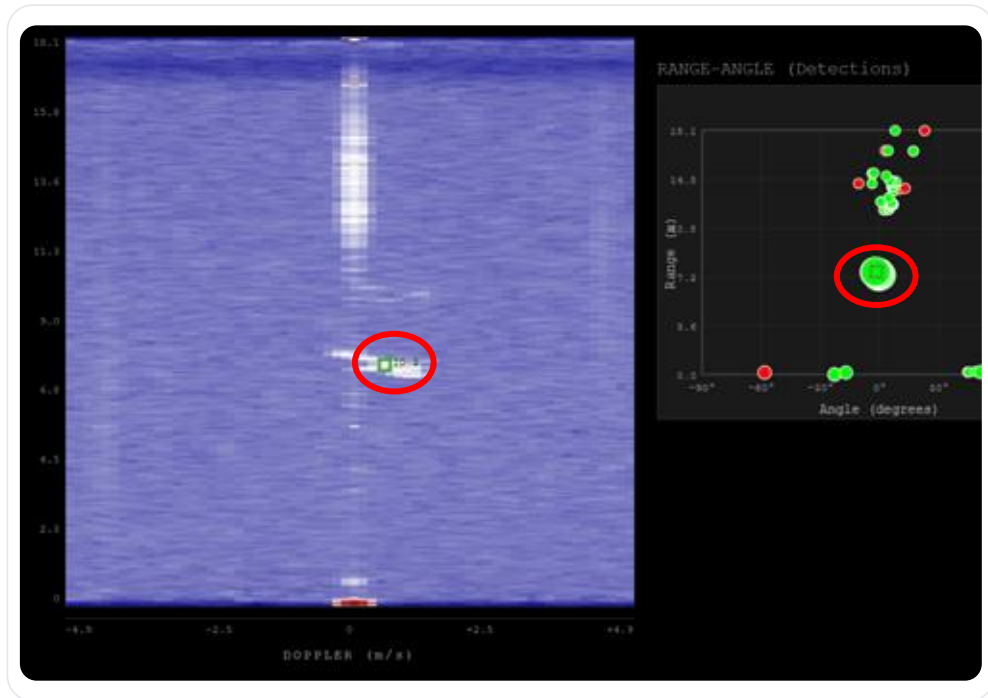


Pipeline: CFAR Point Clustering into Centroid

STEP 06

Clustering

Cluster CFAR Points (Range, Doppler, Angle)
into Centroids using DBScan

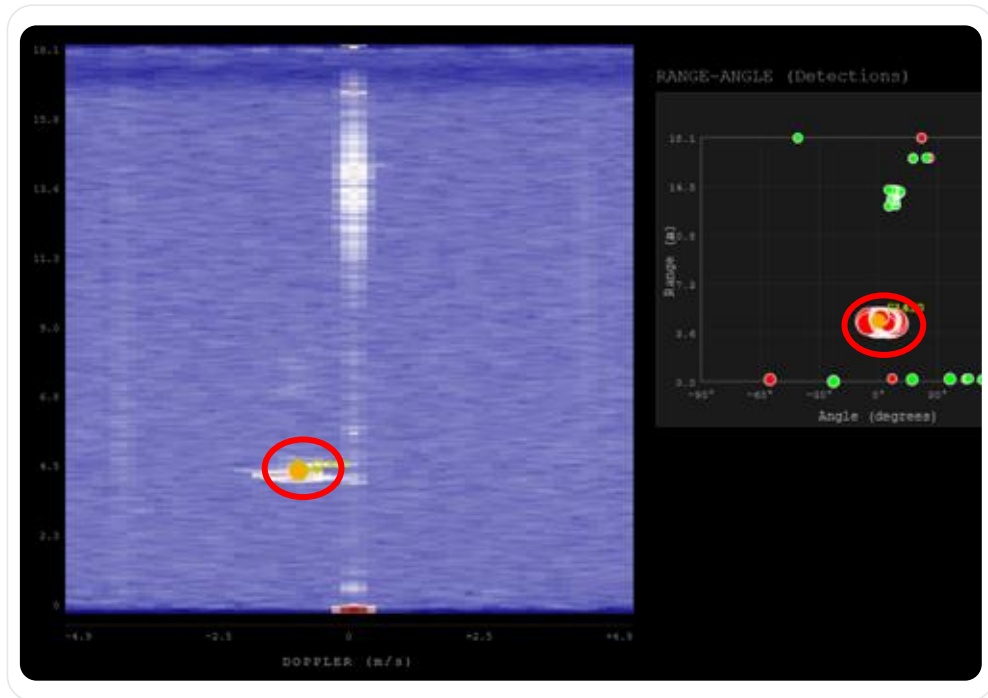


Pipeline: Filtering Centroid Noise

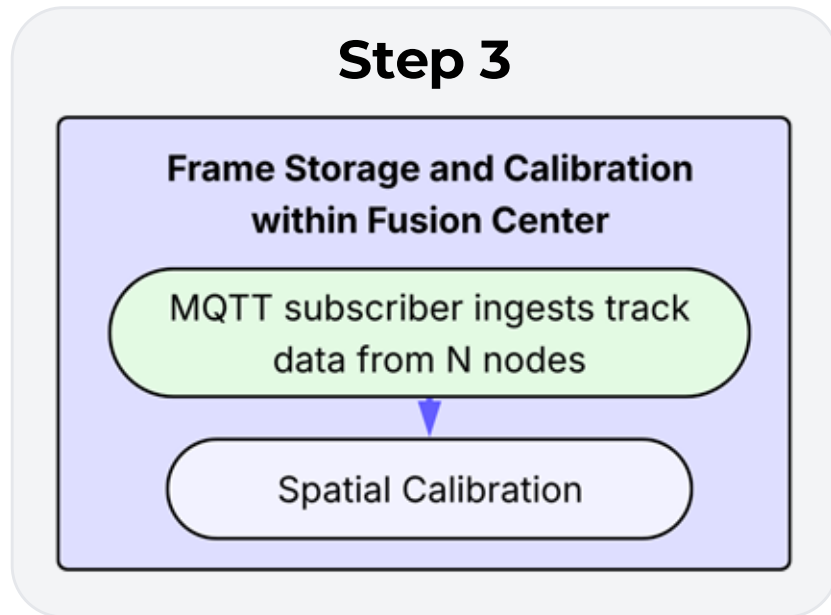
STEP 07

Noise Filtering

Filter Centroids across time using Extended Kalman Filtering (EKF) and Joint Probabilistic Data Association (JPDA)



Dataflow Overview: Spatial Calibration



Spatial Calibration

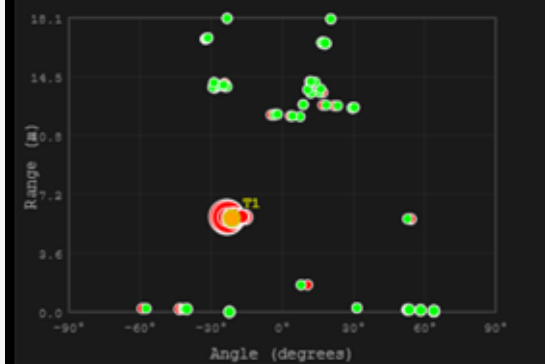
Goal:

- Map all the trajectories together to create fused map
- Requires figuring out the translation & rotations

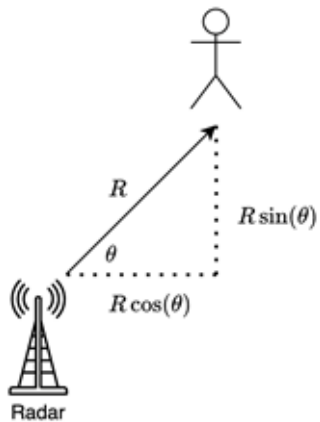
How?

- Publish frames w/ detection data
- Source common frames
- Turn detection list into centroids (blobs of 'objects')
- Algorithms outputting rotation and translation estimation
- Transpose node centroids w/ output to create bird's eye map

RANGE-ANGLE (Detections)



Object
 $(x, y) = (R \cos(\theta), R \sin(\theta))$



Fusion Center Front End

- Hosted on fusion center
- Main Features:
 - Pull from Github
 - per-Node RDM Visualizer (UI)
 - Local Logs
 - Start Calibration
 - Displays Calibration Results
- Allows future capstone team/researchers to quickly iterate and verify software.



Fusion Center Back End

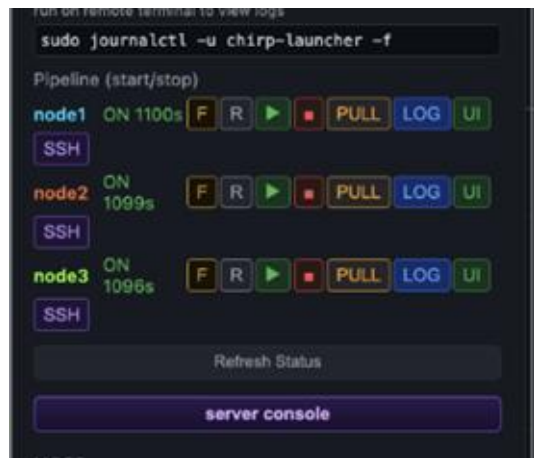
- Dockerized stack for easier deployment
 - Dashboard service for user interface
 - Calibration service for running node calibration
 - Bridge service for mqtt message ingest
 - NanoMQ mqtt broker service for handling mqtt transactions
 - Postgres service for storing centroids and tracks from nodes

IMAGE	COMMAND	CREATED	STATUS
server-dashboard	"python -u dashboard..."	6 hours ago	Up 6 hours
server-bridge	"python -u subscribe..."	7 hours ago	Up 7 hours
server-calibration	"python -u controlle..."	7 hours ago	Up 7 hours
postgres:16-alpine	"docker-entrypoint.s..."	7 hours ago	Up 7 hours (healthy)
emqx/nanomq:latest	"/usr/bin/docker-ent..."	7 hours ago	Up 7 hours (healthy)

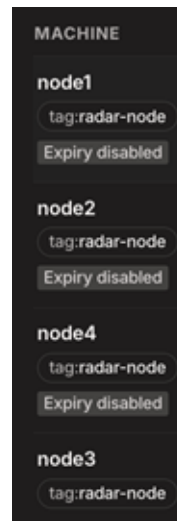
Fusion Center Features

```
May 30 22:02:22 tien1 chirp-launcher[12351]: self.end_headers()
May 30 22:02:22 tien1 chirp-launcher[12351]: File "/usr/lib/python3.10/http/
server.py", line 535, in end_headers
May 30 22:02:22 tien1 chirp-launcher[12351]: self.flush_headers()
May 30 22:02:22 tien1 chirp-launcher[12351]: File "/usr/lib/python3.10/http/
server.py", line 539, in flush_headers
May 30 22:02:22 tien1 chirp-launcher[12351]: self.wfile.write(b"".join(sel
f._headers_buffer))
May 30 22:02:22 tien1 chirp-launcher[12351]: File "/usr/lib/python3.10/socket
server.py", line 826, in write
May 30 22:02:22 tien1 chirp-launcher[12351]: self._sock.sendall(b)
May 30 22:02:22 tien1 chirp-launcher[12351]: BrokenPipeError: [Errno 32] Broke
n pipe
May 30 22:02:22 tien1 chirp-launcher[12351]: -----
May 30 22:02:37 tien1 chirp-launcher[12349]: 2026-05-30 22:02:37,263 [LAUNCHER
] INFO Connected to broker 169.231.45.81:1883
May 30 22:00:56 tien3 chirp-launcher[7239]: File "/usr/lib/python3.10/sockets
erver.py", line 466, in server_bind
May 30 22:00:56 tien3 chirp-launcher[7239]: self.socket.bind(self.server_ad
dress)
May 30 22:00:56 tien3 chirp-launcher[7239]: OSError: [Errno 98] Address already
in use
May 30 22:00:57 tien3 chirp-launcher[7240]: 2026-05-30 22:00:57,054 [SERVER] IN
FO Starting Optimized Asyncio/Aiohttp Radar Plotter on port 5001
May 30 22:00:57 tien3 chirp-launcher[7240]: ===== Running on http://0.0.0.0:
5001 =====
May 30 22:00:57 tien3 chirp-launcher[7240]: (Press CTRL+C to quit)
May 30 22:02:22 tien3 chirp-launcher[7236]: 2026-05-30 22:02:22,267 [LAUNCHER]
WARNING Unexpected disconnect rc=Unspecified error flags=DisconnectFlags(is_d
isconnect_packet_from_server=False)
May 30 22:02:37 tien3 chirp-launcher[7236]: 2026-05-30 22:02:37,300 [LAUNCHER]
INFO Connected to broker 169.231.45.81:1883
```

Web-accessible logs, per node



With Tailscale VPN and DNS resolution, nodes automatically connect to Fusion Center once powered on through daemon service.



DNS resolution

Spatial Calibration Demonstration

Chirp Debug

Uncalibrated

NODES

node3 C:1 T:0

node1 C:1 T:1

node2 T:1

CALIBRATION

Group: default

Frames: 150

Start Calibration

NODE CONTROL

Run on remote terminal to view logs

sudo journalctl -u chirp-launcher -f

Pipeline (start/stop)

node1 ON 1114s F R PULL LOG UI

SSH

node2 ON 2236s F R PULL LOG UI

SSH

node3 ON 1114s F R PULL LOG UI

SSH

Refresh Status

server console

MQTT

Pipeline (start/stop)

Refresh Status

MQTT

Host: nanomq

Port: 1883

User: chirp

Status: connected

CONTROLS

☒ Grid ☒ Node pos

☒ Clusters ☒ Tracks

☐ Trails ☐ Doppler color

Zoom: 4 m/div

STATS

FPS: 25

Points: 3

Nodes: 3

groups: node1,node2,node3 cmd: -

4:18:27 PM

25 FPS

Checking initial track detection

1114

1125

Toggle OV2640 settings

Save





Thank you for listening!

Feel free to ask any questions.

Special thanks to: Anirban Banik, Tien Nguyen, Prof. Isukapalli, Prof. Madhow