

[BP] InCabin Project

목차

1. 서론

[1.1 프로젝트 목표](#)

[1.2 IWR6843 개요 및 주요 스펙](#)

2. 개발 환경 세팅 : 별첨 서류 확인 참고

[2.1 하드웨어 및 소프트웨어, 툴체인 설치](#)

3. 예제 실행 : 별첨 서류 확인 참고

3.1 Uniflash를 이용한 펌웨어 플래싱

3.2 Visualizer를 통한 시각화

4. 데이터 파싱

[4.1 데이터 구조 및 형태](#)

[4.2 데이터 파싱](#)

5. DCA1000을 이용한 raw 데이터 수집

[5.1 DCA1000 환경 설정](#)

[5.2 데이터 수집](#)

6. 펌웨어 개발 환경 세팅 : 별첨 서류 확인 참고

7. 데이터 수집

[7.1 실험 시나리오 설계](#)

[7.2 실험 환경 세팅](#)

1. 서론

1.1 프로젝트 목표

프로젝트 목표

IWR6843AOP 센서로 부터 받아온 실시간 데이터를 기반으로 학습된 AI 모델을 사용하여, 차량 실내 상황 모니터링 어플리케이션 구축

어플리케이션 필수 기능

- 각 좌석의 성인/사물/유아 존재 탐지
- 각 좌석의 성인/유아의 심박수 및 호흡 탐지
- 실시간 시각화를 통한 모니터링 시스템

AI 모델 개발 및 구축

- 대용량 데이터셋 수집
- 학습 모델 선정 또는 개발
- 학습 (파라미터 피팅)

1.2 IWR6843 개요 및 주요 스펙

데이터 시트 → [IWR6843AOP_Datasheet](#)

Parameters	Value
Frequency range	60 - 64 GHz
Number of receivers	4
Number of transmitters	3
ADC sampling rate(kspS)	25000
TX power (dBm)	10
Arm CPU	Arm Cortex-R4F at 200MHz
Coprocessors	Radar Hardware Accelerator
DSP type	1 C67x DSP @600MHz
Interface type	CAN-FD, I2C, LVDS, QSPI, SPI, UART
RAM(kByte)	1792
Operating temperature range (C)	-40 - 105

2. 개발 환경 설정 & 3. 예제 실행 (별첨 서류 확인)

[IWR6843AOPEVM Setup Guide](#)

By Yoonsang Lee 5 min See views Add a reaction

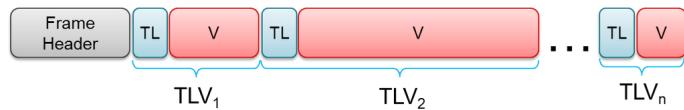
Powered by Confluence

4. 데이터 파싱

4.1 데이터 구조 및 형태

데이터 구조

IWR6843AOP는 TLV (Type Length Value)라는 형태로 데이터 전송



매 프레임마다 위와 같은 형태로 데이터 packet을 전송

- Frame Header: 각 프레임의 시작을 알리며 패킷의 메타데이터 정보를 포함
 - TLV: 사전에 정의된 데이터 종류에 따라 인코딩된 데이터 패킷

IWR6843 데이터 구조

4.2 데이터 파싱

데이터 파싱

위의 데이터 구조 표를 참고

1. 들어오는 데이터를 펌웨어에 사전 정의 된 Magic Word와 매칭하여 프레임의 시작을 인지
 2. Frame Header의 Total Pack Length로 해당 프레임에 대한 데이터의 끝을 인지
 3. Frame Header가 끝나면 Type이 들어오므로 해당 Type이 어떤 데이터를 의미하는지 확인
 4. Type과 Length에 따라 해당 데이터의 끝을 인지
 5. 4에서 인지한 해당 데이터의 끝까지 디코딩을 했다면 2번에서 인지한 해당 프레임의 끝이 나올 때까지 다음 Frame Header가 들어올지 Type에 대해 4.5를 반복

TLV 디코딩

예를 들어, Type 10200이 한 점에 N 바이트를 가지는 포인트 클라우드 데이터라면, 한 프레임에서 들어오는 포인트 클라우드의 개수는 가변이며 Length를 따르므로 $Length \times N$ byte 만큼의 데이터를 포인트 클라우드로 디코딩 해야함

데이터 파서 함수는 아래 위치의 “common” 폴더의 다음 3가지 파일 참고

"C:\ti\radar_toolbox_3_10_00_05\tools\visualizers\Applications Visualizer\common"

- parseFrame.py

- tlvDefines.py

- parseTLVs.py

parseFrame.py

```
18 parserFunctions = {  
19     MMWDEMO_OUTPUT_MSG_DETECTED_POINTS:           parsePointCloudTLV,  
20     MMWDEMO_OUTPUT_MSG_RANGE_PROFILE:              parseRangeProfileTLV,  
21     MMWDEMO_OUTPUT_EXT_MSG_RANGE_PROFILE_MAJOR:   parseRangeProfileTLV,
```

parseFrame.py 의 parserFunctions

왼쪽 파란색 → 데이터 종류

- ctrl+클릭 하면 tlvDefines.py로 넘어감 type 번호가 매겨져 있음
- type자리에 나오는 번호로 데이터 종류 파악 후 파서 함수와 매칭

```
# Defined TLV's  
MMWDEMO_OUTPUT_MSG_DETECTED_POINTS = 1  
MMWDEMO_OUTPUT_MSG_RANGE_PROFILE = 2  
MMWDEMO_OUTPUT_MSG_NOISE_PROFILE = 3  
MMWDEMO_OUTPUT_MSG_AZIMUT_STATIC_HEAT_MAP = 4
```

tlvDefines.py의 정의된 TLV

오른쪽 노란색 → 데이터 종류에 따른 파서 함수

- ctrl+클릭 하면 각 데이터 종류에 맞는 파서 함수들이 있음

정의되지 않은 TLV도 존재함. 이런 경우 파서 함수를 직접 만들면 됨.

- 해당 type에 대한 데이터 패킷 정보는 해당 예제의 Doc 파일 참고.

```
68 unusedTLVs = [  
69     MMWDEMO_OUTPUT_MSG_NOISE_PROFILE,  
70     MMWDEMO_OUTPUT_MSG_AZIMUT_STATIC_HEAT_MAP,  
71     MMWDEMO_OUTPUT_MSG_RANGE_DOPPLER_HEAT_MAP,  
72     MMWDEMO_OUTPUT_MSG_STATS,  
73     MMWDEMO_OUTPUT_MSG_PRESENCE_INDICATION,  
74     MMWDEMO_OUTPUT_MSG_GESTURE_PRESENCE_x432,  
75     MMWDEMO_OUTPUT_MSG_GESTURE_PRESENCE_THRESH_x432,  
76     MMWDEMO_OUTPUT_EXT_MSG_MICRO_DOPPLER_RAW_DATA,  
77     MMWDEMO_OUTPUT_EXT_MSG_MICRO_DOPPLER_FEATURES,  
78     MMWDEMO_OUTPUT_EXT_MSG_QUICK_EVAL_INFO,  
79     MMWDEMO_OUTPUT_EXT_RANGE_DOPPLER_MAP,  
80     MMWDEMO_OUTPUT_EXT_POINT_CLOUD_CARTESIAN,  
81     MMWDEMO_OUTPUT_EXT_POINT_CLOUD_SPHERICAL,  
82     MMWDEMO_OUTPUT_EXT_CLASSIFIER_VARS  
83 ]
```

parseFrames.py의 사용(정의)되지 않은 TLV

FrameHeader 디코딩

- 아래 코드중 87번째 줄의 headerStruct = 'Q8I'
 - 해당 구조는 3D people tracking의 header 구조임
 - 따라서 다른 예제의 Header 구조는 다를 수 있음
 - 예를 들어 Area Scan 예제의 경우 headerStruct = 'Q9I'로 변경 필요

```

def parseStandardFrame(frameData):
    # Constants for parsing frame header
    headerStruct = 'Q8I'
    frameHeaderLen = struct.calcsize(headerStruct)
    tlvHeaderLength = 8

    # Define the function's output structure and initialize error field to no error
    outputDict = {}
    outputDict['error'] = 0

    # A sum to track the frame packet length for verification for transmission integrity
    totalLenCheck = 0

    # Read in Frame Header
    try:
        magic, version, totalPacketLen, platform, frameNum, timeCPUcycles, numDetectedObj, numTLVs, subFrameNum = struct.unpack(headerStruct, frameData[:frameHeaderLen])
    except:
        log.error('Error: Could not read frame header')
        outputDict['error'] = 1

    # Move frameData ptr to start of 1st TLV |
    frameData = frameData[frameHeaderLen:]
    totalLenCheck += frameHeaderLen

    # Save frame number to output
    outputDict['frameNum'] = frameNum

```

parseFrame.py의 parseStandardFrame()

Q → unsigned int 8byte

I → unsigned int 4byte

더 많은 구조 참고 → [struct — Interpret bytes as packed binary data](#)

Value	Type	Bytes
Magic Word	uint16_t	8
Version	uint32_t	4
Total Packet Length	uint32_t	4
Platform	uint32_t	4
Frame Number	uint32_t	4
Time [in CPU Cycles]	uint32_t	4
Num Detected Obj	uint32_t	4
Num TLVs	uint32_t	4
Subframe Number	uint32_t	4
Num Static Detected Obj	uint32_t	4

Area Scan Head 구조

Value	Type	Bytes
Magic Word	uint64_t	8
Version	uint32_t	4
Total Packet Length	uint32_t	4
Platform	uint32_t	4
Frame Number	uint32_t	4
Time [in CPU Cycles]	uint32_t	4
Num Detected Obj	uint32_t	4
Num TLVs	uint32_t	4
Subframe Number	uint32_t	4

3D People Tracking Header 구조

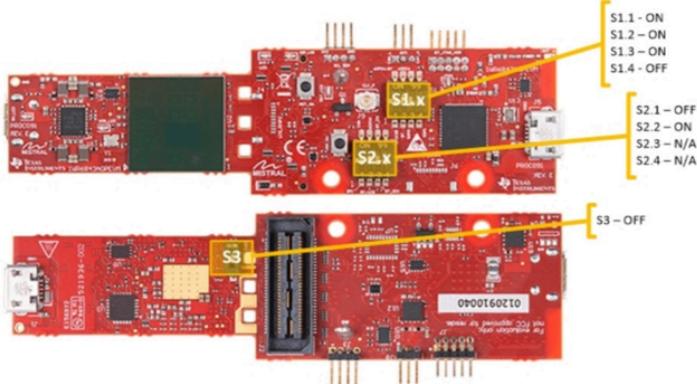
5. DCA1000을 이용한 raw 데이터 수집

참고 문서: [dca1000_mmwave_studio_user_guide.html](#)

5.1 DCA1000 환경 설정

Hardware Setup: AOP EVM's

1. IWR6843AOPEVM, mmWaveIcBoost, DCA1000의 스위치 및 커넥터 설정



IWR6843 AOP Switches Setup

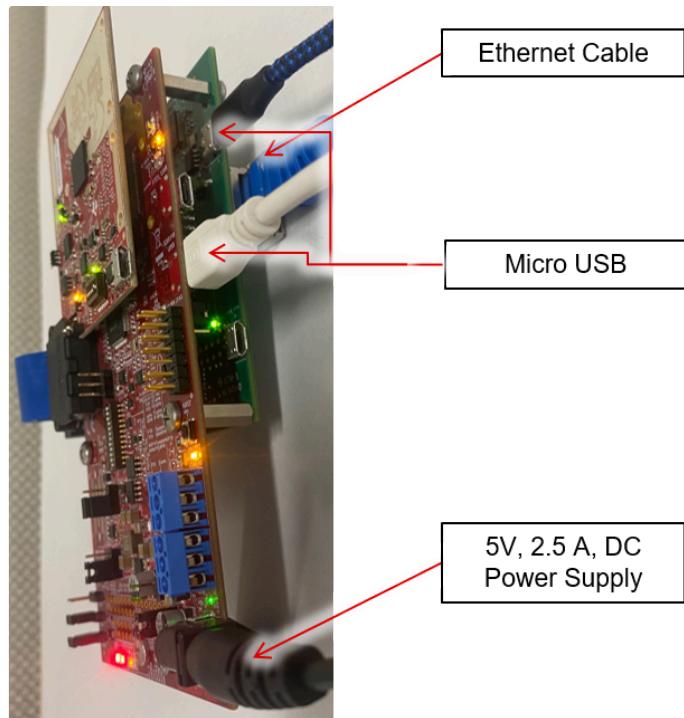
IWR6843AOPEVM과 ICBOOST는 직접 연결, ICBOOST와 DCA1000은 파란 커넥터로 연결

SOP name	Setting
S1.8	OFF
S1.7	OFF
S1.6	OFF
S1.5	ON
S1.4	OFF
S1.3	OFF
S1.2	OFF
S1.1	OFF

Switch name	Setting when connect with DCA
S1.12	ON
S1.11	ON
S1.10	ON
S1.9	OFF
S1.8	OFF
S1.7	OFF
S1.6	OFF
S1.5	ON
S1.4	OFF
S1.3	ON
S1.2	ON
S1.1	OFF

SOP name	Setting
SOP2	OFF
SOP1	ON
SOP0	ON

mmWaveIcBoost Switches Setup



Wiring Setup

2. Software Setup

다음 항목들은 앞의 IWR6843의 개발을 위한 셋업에서 미리 설치되어 있어야 함.

** 개발 환경 셋업 과정을 거쳤을 경우 아래 항목 모두 설치 되어있음.

- 1) FTDI & XDS drivers
- 2) [MATLAB Runtime Engine v8.5.1](#)
- 3) mmWave Studio

3. 위 과정이 완료되면 “장치 관리자”의 Ports(포트)에서 다음과 같이 장치 확인



장치 연결 후 장치관리자 port 목록

⚠️ 드라이버가 설치 되지 않은 경우

드라이버가 설치되지 않은 경우, FTDI 포트에 아래와 같은 노란색 라벨이 표시



이 경우, 드라이버 소프트웨어 업데이트 → 내 컴퓨터에서 드라이버 소프트웨어 찾아보기 → 디렉토리 선택 → “하위 폴더 포함” 체크 → “~\mmwave_studio_xx_xx_xx_xx\ftdi”

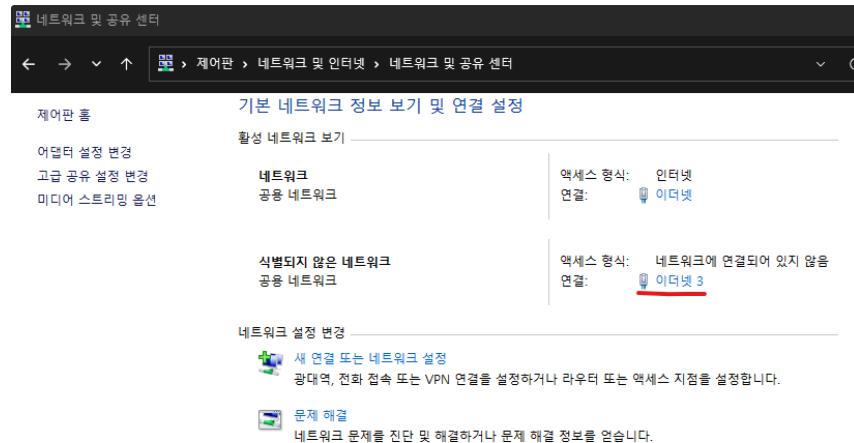
이 작업을 4개 포트 각각에 수행

4. 이더넷 데이터 전송을 위한 고정 IP 설정

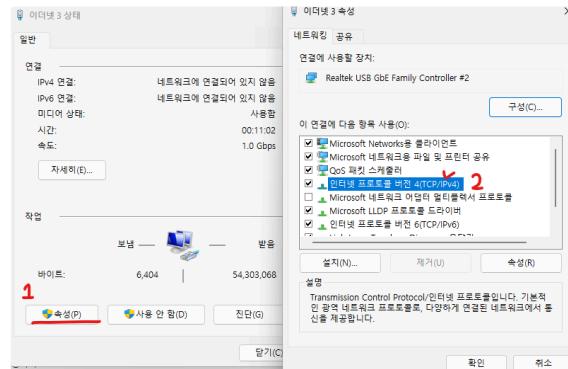
mmWave Studio에 이미 특정 IP에서 데이터를 받아오는 것으로 설정이 되어있으므로 해당 IP를 고정으로 설정해주어야 함.

a. “제어판” → “네트워크 및 인터넷” → “네트워크 및 공유센터”

아래 사진과 같이 장치 연결후 새로운 네트워크 확인 가능

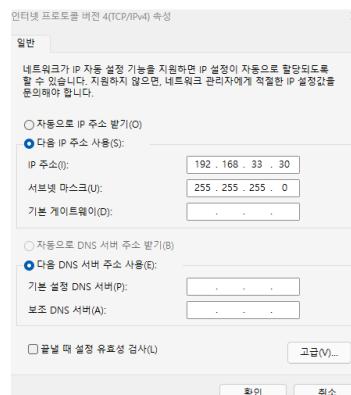


b. 이더넷3(이름이 다를 수 있음) 클릭 → 속성 → 인터넷 프로토콜 버전 4(TCP/IPv4) 더블 클릭



c. 다음과 같이 IP주소/서브넷 마스크 설정 후 저장

- ip : 192.168.33.30
- subnet mask : 255.255.255.0

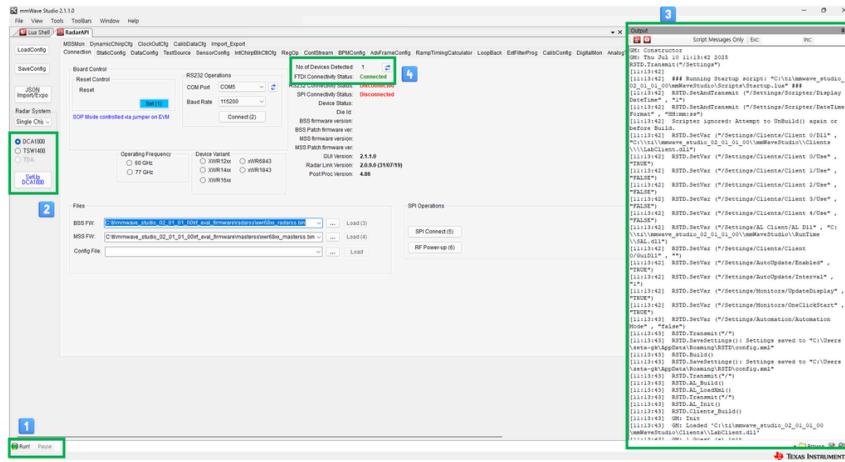


5.2 데이터 수집

mmWave Studio 실행 - Connection 탭



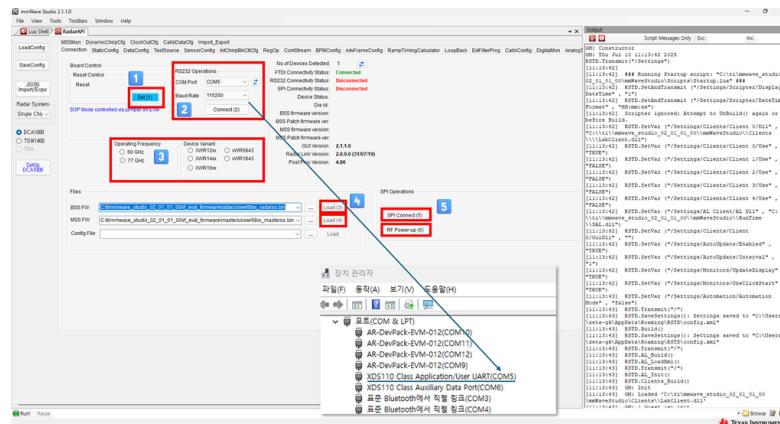
a. 장치의 스위치, 커넥터, 드라이버가 정상적으로 설치되었다면 실행 후 잠시뒤 다음처럼 디스플레이됨



① 위 그림 참조, 체크리스트

- 1 오른쪽 아래 초록색 Run 확인
- 2 왼쪽 Setup DCA1000확인 (화면에서 안보일 경우, 좌우 스크롤)
- 3 오른쪽 output Messege (없으면 상단바 View → output 클릭)
- 4 화면 중앙의 "FTDI Connectivity Status"가 빨간색이면 FTDI 재설치

b. Connection 탭 설정



① Connection 설정 (위 그림 참조)

- 1 Set(1) 클릭
- 2 장치 관리자에서 “XDS110 Class Application/User UART 포트 번호에 br 115200 설정 후 Connect(2) 클릭
- 3 Operating Frequency: 60GHz, Device Varant: xWR6843 (필요시 변경)
- 4 BSS/MSS FW에 mmwave_studio에서 제공하는 펌웨어 로드. (아래 주소 참고)

아래 경로를 통해 파일을 업로드 후 → 로드 버튼 클릭

- C:\ti\mmwave_studio_02_01_01_00\rf_eval_firmware\radarss\xwr68xx_radarss.bin
- C:\ti\mmwave_studio_02_01_01_00\rf_eval_firmware\mastersss\xwr68xx_masterss.bin

- 5 SPI Connect 클릭 후 SPI Connectivity Status가 Connected로 바뀌는지 확인 후 RF Power-up 클릭

위의 Connection 탭 설정 과정이 모두 끝나면 아래와 같이 상태창 확인 가능

No. of Devices Detected: 1

FTDI Connectivity Status: Connected

RS232 Connectivity Status: Connected

SPI Connectivity Status: Connected

Device Status: UnDetDe/QM/SOP:2/ES:2

Die Id: Lot:509421/Wafer:8/DevX:10/DevY:29

BSS firmware version: 6.2.1.5 (09/03/20)

BSS Patch firmware ver: NA

MSS firmware version: 2.0.0.3 (27/08/19)

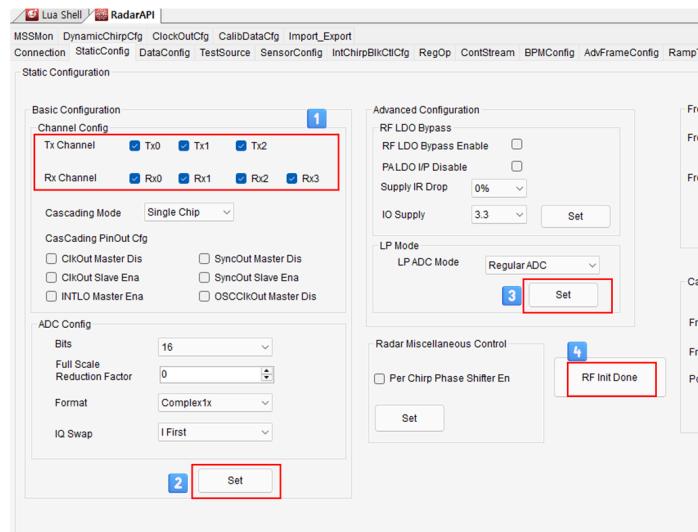
MSS Patch firmware ver: NA

GUI Version: 2.1.1.0

Radar Link Version: 2.0.9.0 (31/07/19)

Post Proc Version: 4.86

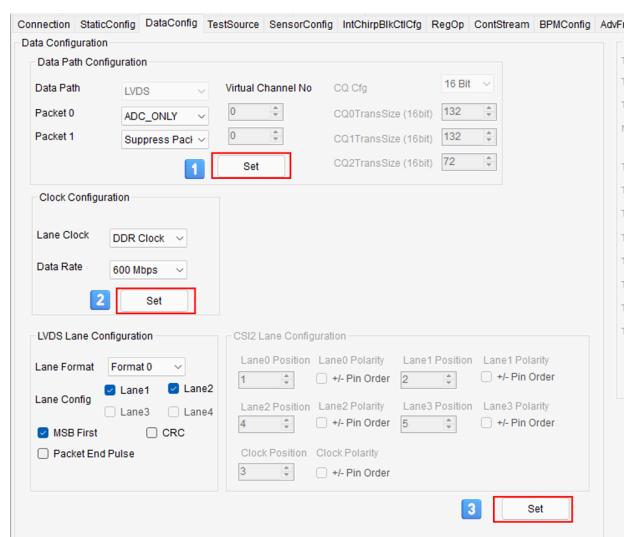
mmWave Studioo 실행 - StaticConfig 탭



StaticConfig 설정 (위 그림 참조)

- 1 사용할 채널 선택
- 2 3 4 순차적으로 클릭

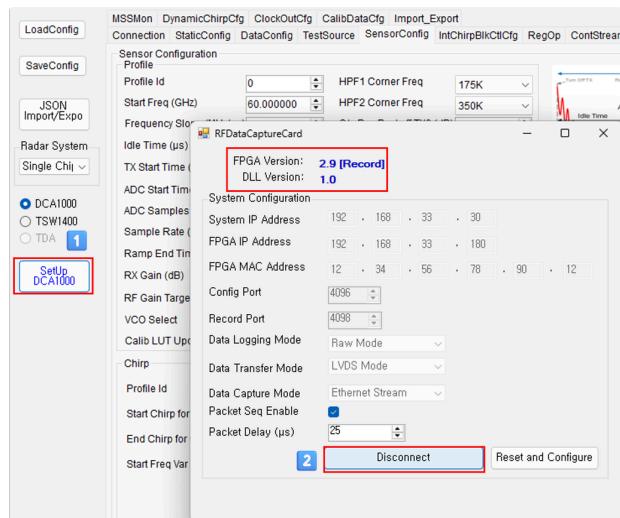
mmWave Studioo 실행 - DataConfig 탭



DataConfig 설정 (위 그림 참조)

- 1 2 3 순차적으로 클릭

mmWave Studio 실행 - Setup DCA1000 연결



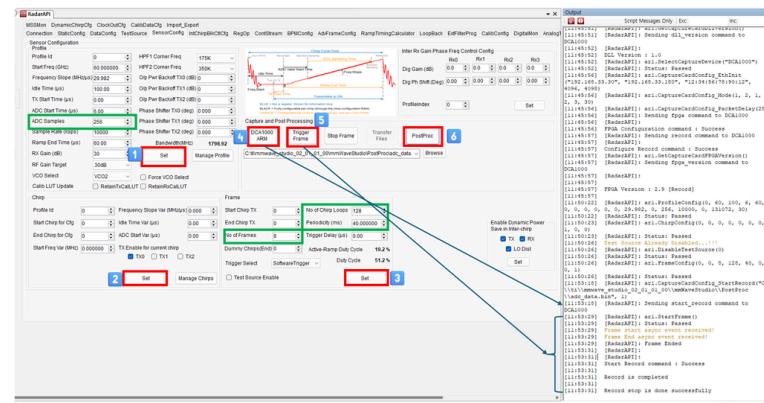
- i** Setup DCA1000 설정 (위 그림 참조)

1 Setup DCA1000 클릭(화면에서 안보일 경우 , 좌우 스크롤)

2 Connect 클릭 후 상단 FPGA Version이 뜨면 연결 성공

⚠ 위 그림처럼 숫자가 아니라 “FPGA Version” 등 다른 문자가 뜨면 모든 연결을 해제, 프로그램 종료 후 다시 시작

mmWave Studio 실행 - SensorConfig 설정



- i** SensorConfig 설정 (위 그림 참조)

1 Set 클릭 (ADC Samples 수에 따라 샘플 개수 변화, 현재 그림의 값은 256 개)

2 Set 클릭

3 Set 클릭 (No of Frames, Periodicity로 수집할 데이터 수와 주기 설정, No of Chirp Loops로 쳐프 수 조정 → 현재 그림의 값은 8 Frame, 40ms 주기, 쳐프 수 128개)

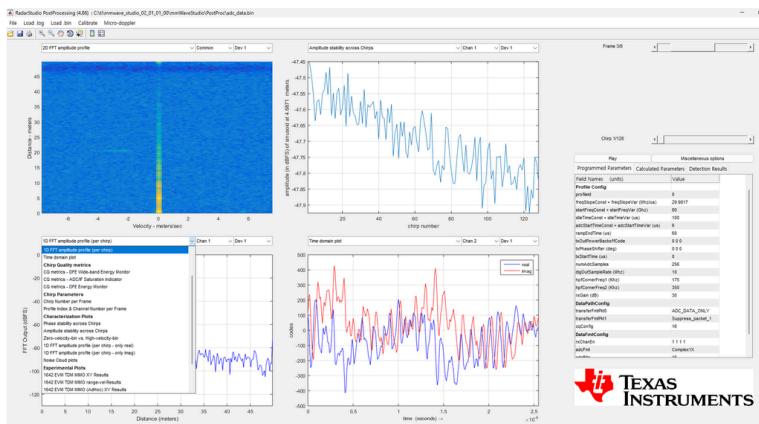
4 DCA1000 ARM 클릭 후 output에 Sending start~ DCA1000 출력 확인

5 Trigger Frame 클릭 후 데이터 수집 시작됨(설정한 데이터 수 및 주기에 따라), 수집이 끝나면 오른쪽 output 창에 Record stop is done successfully 확인 가능 이 때 Trigger Frame 버튼 아래 경로에 .bin 파일로 데이터 저장됨

(에러 시 다시 모든 연결 해제 후 다시 시작)

```
[15:02:56] [RadarAPI]:
[15:02:56] Start Record command : Success
[15:02:56]
[15:02:56] Record is completed
[15:02:56]
[15:02:56] Record stop is done successfully
```

- 6 PostProc 클릭, 아래 저장된 데이터를 일련의 후처리 과정을 거친후 아래 그림처럼 시각화를 위한 새 창이 뜸.



PostProcessed Plot

앞서 SensorConfig에서 설정한 대로 Frame은 총 8개, Chirp는 총 128개인 것을 확인 가능

한번에 4개의 plot을 볼 수 있으며 Channel 별로 확인 가능

오른쪽의 Play 버튼을 누르면 시간 순서대로 확인 가능 - [DCA 1000 mmwave_studio : PostProc 결과 분석](#)

6. 펌웨어 개발 환경 셋업 (별첨 서류 확인)

Debugging using CCS

Edit

Debugging using CCS

By Yoonsang Lee 3 min See views Add a reaction

Powered by Confluence

7. 데이터 수집

7.1 실험 시나리오 설계

타겟 시나리오

- 앞 두 좌석, 뒷 세좌석에 사람이 최대 5명 탄다고 가정
- 성인/유아 또는 사물이 존재할 수 있음
- 탑승자의 호흡/심박수를 측정하여 이상 패턴 감지
- 유아의 존재를 감지 (특히 차에 유아만 있는 경우)

실험 시나리오 고려 사항

cfg 파일의 내용 수정으로 변경 가능한 부분

- 레이더의 위치 (Azimuth 각도/ Elevation 각도)
- 탐지 범위 (차량 크기)와 그 외의 노이즈/반사체 제거된 환경
- SNR threshold 조정하여 민감도 설정
- 정적/동적 물체의 속도 threshold 조정

데이터 수집 규칙

데이터 수집 시 다음과 같이 장소, 탑승 분류, 탑승자 코드를 지정 in-cabin 데이터 수집.xlsx

예시 데이터 폴더 명: <장소코드>_<탑승분류>_<탑승자코드>_<시간>

L1_ANAON_GK,YL_20250704135250

- img
- rdr
- L1_ANAON_GK,YL.json →

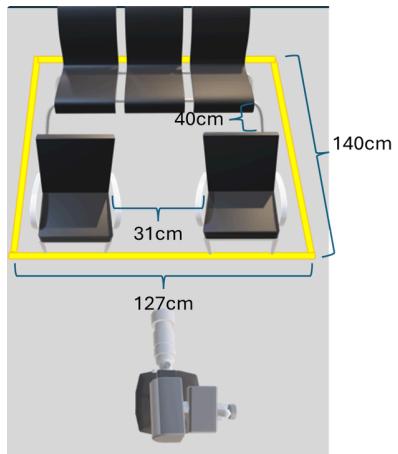
```
1  "BasicInfo": {  
2      "place": "L1",  
3      "date": "20250704135250",  
4      "code_pg": "ANAON",  
5      "idx_pg": "GK,YL"  
6  },  
7  "Occupants": {  
8      "S1": "0",  
9      "S2": "",  
10     "S3": "0",  
11     "S4": "",  
12     "S5": ""  
13  },  
14  "Objects": {  
15      "O1": "0",  
16      "O2": "",  
17      "O3": "",  
18      "O4": "",  
19      "O5": "",  
20      "Num_OBJ": "1",  
21      "OBJ_NAME": "D"  
22  },  
23  "Status": {  
24      "PAUSE": "",  
25      "DYNAMIC": ""  
26  },  
27  "Vitals": {  
28      "HEARTBEAT": "",  
29      "WATCH_ID": ""  
30  },  
31  "Collection": {  
32      "COMPLETE": "",  
33      "NOTE": ""  
34  }  
35 }]
```

json 구조

7.2 실험 환경 셋업

L1 실험 환경 정보

3인용 의자 1개 + 1인용 의자 2개로 차량 내부를 대신함



아래 노란색 마스킹 테이프로 항상 같은 위치
에 세팅

- 가로 전체: 127cm
- 세로 전체: 140cm
- 앞좌석 사이 거리: 31cm
- 앞 뒷좌석 사이 거리: 40cm

*원쪽 문을 닫을 시 외부와 차단됨을 확인