BÁO CÁO TỔNG HỢP: CẤU TRÚC CÁC FILE H5 (PYTHON)

Ngày tạo báo cáo: 2026-01-25 15:41:24

MỤC LỤC

==========
==========

PHẦN 1

==========

BÁO CÁO CHI TIẾT: CẤU TRÚC FILE IQETHERNET.H5

1. TỔNG QUAN

File `iqethernet.h5` là file HDF5 chứa dữ liệu IQ (In-phase và Quadrature) từ Ethernet Packets. File này được đọc bởi module Python `reader_iqethernet_h5.py` dựa trên code MATLAB `read_iq_ethernet_h5_verge.m`.

Thông tin cơ bản:

- **Định dạng**: HDF5 (Hierarchical Data Format version 5)
- **Loại dữ liệu**: IQ samples (Ethernet Packets)
- **Cấu trúc chính**:
- - `/attribute`: Chứa metadata (thông tin chung, DDC, request, ...)
- - `/session`: Chứa dữ liệu raw bytes của các Ethernet packets

---

2. CẤU TRÚC FILE H5

2.1. Cấu trúc tổng thể

```
iqethernet.h5
├── /attribute                    # Group chứa metadata
│   ├── Attributes (trực tiếp)    # → global_info
│   ├── /ddc                      # → global_info['ddc']
│   │   └── Attributes
│   ├── /request                  # → global_info['request']
│   │   └── Attributes
│   └── ... (các group con khác)
│
└── /session                      # Group chứa dữ liệu raw packets
    ├── /0000000000000000000      # Session ID
    │   └── /raw                  # Dataset: Raw Ethernet packet bytes
    ├── /0000000000000000001
```

```
│   └── /raw
└── ... (nhiều sessions)
```

Thống kê:
- **Số streams**: 1
- **Tổng số packets**: 12737

2.2. Chi tiết các thành phần

A. `/attribute` Group

Group này chứa tất cả metadata của file:

8. **Attributes trực tiếp tại `/attribute`**:

- - Được đọc vào `data['global_info']`
- - Chứa thông tin chung như: `client_ip`, `frequency`, `bandwidth`, `channel`, `mission`, ...

9. **Sub-groups trong `/attribute`**:

- - Mỗi sub-group được đọc vào `data['global_info'][sub_name]`
- - Ví dụ: `/attribute/ddc` → `data['global_info']['ddc']`
- - Ví dụ: `/attribute/request` → `data['global_info']['request']`

B. `/session` Group

Group này chứa dữ liệu raw bytes của các Ethernet packets:

- Mỗi session có ID dạng: `0000000000000000`, `0000000000000001`, ...
- Mỗi session chứa 1 dataset:
- - `raw`: Raw Ethernet packet bytes (uint8 array)
- Mỗi packet được giải mã theo cấu trúc:

- - Header (40 bytes): header, stream_id, timestamp, frequency, len, bandwidth, switch_id, sample_cnt
- - Payload: IQ samples (Int16, xen kẽ I và Q)

---

3. INPUT VÀ OUTPUT CỦA READER

3.1. Input

Hàm: `read_iqethernet_h5(filename: str)`

Tham số:

- `filename` (str): Đường dẫn đến file H5 cần đọc
- - Ví dụ: ``'../../00_DATA_h5/iqethernet.h5'``

3.2. Output

Kiểu trả về: `Dict[str, Any]`

Cấu trúc output:

```
{
    'global_info': {
        # Dictionary chứa tất cả attributes từ /attribute
        # Ví dụ:
        # 'client_ip': '192.168.1.100',
        # 'frequency': 1000000000.0,  # Hz
        # 'bandwidth': 20000000.0,    # Hz
        # 'ddc': {...},  # Sub-group attributes
        # 'request': {...},  # Sub-group attributes
        # ...
    },

    'streams': {
```

```
        'Stream_0': {
            'packets': [
                {
                    'header': uint32,
                    'stream_id': uint32,
                    'timestamp': uint64,
                    'frequency': uint64,  # Hz
                    'len': uint32,
                    'bandwidth': uint32,  # Hz
                    'switch_id': uint32,
                    'sample_cnt': uint32,
                    'iq_data': np.ndarray,  # Complex IQ samples
                    'h5_session_idx': int  # Index trong H5 file
                },
                # ... (nhiều packets)
            ],
            'all_iq': np.ndarray  # (Optional) Tất cả IQ nối lại
        },
        'Stream_1': {...},
        # ... (nhiều streams)
    }
}
```

---

4. HƯỚNG DẪN SỬ DỤNG READER

4.1. Cài đặt

Yêu cầu:

- Python 3.6+
- Thư viện: `h5py`, `numpy`

Cài đặt dependencies:

```
pip install h5py numpy
```

4.2. Cách sử dụng cơ bản

```
from reader_iqethernet_h5 import read_iqethernet_h5
```

```
# Đọc file
filename = 'path/to/iqethernet.h5'
data = read_iqethernet_h5(filename)

# Truy cập thông tin
print(f"Số streams: {len(data['streams'])}")
if 'global_info' in data and 'frequency' in data['global_info']:
    print(f"Frequency: {data['global_info']['frequency']} Hz")
```

4.3. Cách lấy các trường thông tin output

A. Lấy thông tin chung (Global Info)

```
# Lấy toàn bộ global_info
global_info = data['global_info']

# Lấy từng trường cụ thể
if 'client_ip' in data['global_info']:
    client_ip = data['global_info']['client_ip']
if 'frequency' in data['global_info']:
    frequency = data['global_info']['frequency']  # Hz
if 'bandwidth' in data['global_info']:
    bandwidth = data['global_info']['bandwidth']  # Hz

# Lấy sub-group (ví dụ: ddc)
if 'ddc' in data['global_info']:
    ddc_info = data['global_info']['ddc']
    # Truy cập các trường trong ddc
    # ddc_channel = ddc_info.get('channelIndex', None)

# In ra tất cả các trường
for key, value in data['global_info'].items():
    if isinstance(value, dict):
        print(f"{key}: (sub-group)")
        for sub_key, sub_value in value.items():
            print(f"  {sub_key}: {sub_value}")
    else:
        print(f"{key}: {value}")
```

B. Lấy dữ liệu Streams

1. Lấy danh sách tất cả streams:

```
# Lấy tất cả stream names
stream_names = list(data['streams'].keys())
```

```
print(f"Có {len(stream_names)} streams: {stream_names}")

# Lấy một stream cụ thể
stream_0 = data['streams']['Stream_0']
```

2. Lấy packets từ một stream:

```
# Lấy stream
stream = data['streams']['Stream_0']

# Lấy danh sách packets
packets = stream['packets']
print(f"Số packets: {len(packets)}")

# Lấy packet đầu tiên
packet = packets[0]
print(f"Stream ID: {packet['stream_id']}")
print(f"Frequency: {packet['frequency']} Hz ({packet['frequency']/1e6:.2f}
MHz)")
print(f"Bandwidth: {packet['bandwidth']} Hz ({packet['bandwidth']/1e6:.2f}
MHz)")
print(f"Sample count: {packet['sample_cnt']}")
```

3. Lấy dữ liệu IQ từ packet:

```
import numpy as np

# Lấy packet
packet = data['streams']['Stream_0']['packets'][0]

# Lấy dữ liệu IQ phức
iq_data = packet['iq_data']  # numpy array, dtype: complex128
print(f"IQ data: shape={iq_data.shape}, dtype={iq_data.dtype}")
print(f"IQ samples (5 đầu): {iq_data[:5]}")

# Tính toán từ IQ phức
# Biên độ (Magnitude)
magnitude = np.abs(iq_data)
print(f"Magnitude: min={magnitude.min():.2f}, max={magnitude.max():.2f}")

# Phase (Góc pha)
phase = np.angle(iq_data)
print(f"Phase: min={phase.min():.3f}, max={phase.max():.3f}")

# Power
power = np.abs(iq_data) ** 2
print(f"Power: mean={power.mean():.2f}")
```

4. Lấy all_iq (tất cả IQ nối lại):

```
# Lấy stream
stream = data['streams']['Stream_0']

# Lấy all_iq (nếu có)
if 'all_iq' in stream:
    all_iq = stream['all_iq']  # Tất cả IQ từ tất cả packets nối lại
    print(f"All IQ: {len(all_iq)} samples")
    print(f"Shape: {all_iq.shape}")
else:
    print("Không có all_iq")
```

5. Duyệt qua tất cả streams và packets:

```
import numpy as np

# Duyệt qua tất cả streams
for stream_name, stream_data in data['streams'].items():
    print(f"\nStream: {stream_name}")
    packets = stream_data['packets']
    print(f"  Số packets: {len(packets)}")

    # Duyệt qua từng packet
    for i, packet in enumerate(packets):
        stream_id = packet['stream_id']
        frequency = packet['frequency']
        iq_data = packet['iq_data']

        # Xử lý dữ liệu...
        if len(iq_data) > 0:
            magnitude = np.abs(iq_data)
            phase = np.angle(iq_data)
            print(f"    Packet {i}: Stream ID={stream_id}, "
                  f"Freq={frequency/1e6:.2f} MHz, "
                  f"IQ samples={len(iq_data)}")

        # Ví dụ: chỉ xử lý 10 packets đầu
        if i >= 10:
            break
```

5. VÍ DỤ CODE HOÀN CHỈNH

```
#!/usr/bin/env python3
"""
Ví dụ sử dụng reader_iqethernet_h5.py
"""

import numpy as np
from reader_iqethernet_h5 import read_iqethernet_h5

# 1. Đọc file
filename = '../../00_DATA_h5/iqethernet.h5'
print(f"Đang đọc file: {filename}")
data = read_iqethernet_h5(filename)

# 2. Hiển thị thông tin chung
print("\n=== THÔNG TIN CHUNG ===")
print(f"Số streams: {len(data['streams'])}")

if 'global_info' in data:
    print("\nGlobal Info:")
    for key, value in data['global_info'].items():
        if isinstance(value, dict):
            print(f"  {key}: (sub-group)")
            for sub_key, sub_value in value.items():
                print(f"    {sub_key}: {sub_value}")
        else:
            print(f"  {key}: {value}")

# 3. Xử lý dữ liệu từ stream đầu tiên
print("\n=== XỬ LÝ STREAM ĐẦU TIÊN ===")
if data['streams']:
    stream_name = list(data['streams'].keys())[0]
    stream = data['streams'][stream_name]
    print(f"Stream: {stream_name}")
    print(f"Số packets: {len(stream['packets'])}")

    if stream['packets']:
        packet = stream['packets'][0]
        print(f"\nPacket đầu tiên:")
        print(f"  Stream ID: {packet['stream_id']}")
        print(f"  Frequency: {packet['frequency']} Hz
({packet['frequency']/1e6:.2f} MHz)")
        print(f"  Bandwidth: {packet['bandwidth']} Hz
({packet['bandwidth']/1e6:.2f} MHz)")
        print(f"  Sample count: {packet['sample_cnt']}")

        if 'iq_data' in packet and len(packet['iq_data']) > 0:
            iq_data = packet['iq_data']
            iq_mag = np.abs(iq_data)
            print(f"  IQ data: {len(iq_data)} samples")
            print(f"  Magnitude: Min={iq_mag.min():.2f}, Max={iq_mag.max():.2f},
Mean={iq_mag.mean():.2f}")
            print(f"  Phase: Min={np.angle(iq_data).min():.3f},
Max={np.angle(iq_data).max():.3f}")
```

---

6. BẢNG TÓM TẮT CẤU TRÚC OUTPUT

Trường	Kiểu dữ liệu	Mô tả	Ví dụ truy cập
`global_info`	**`dict`**	**Attributes từ `/attribute`**	**`data['global_info'] ['frequency']`**
`global_info['ddc']`	`dict`	Attributes từ `/attribute/ddc`	`data['global_info'][‘ddc'] ['channelIndex']`
`streams`	`dict`	Dictionary các streams	`data['streams'] ['Stream_0']`
`streams['Stream_X'] ['packets']`	`list`	Danh sách packets	`data['streams'] ['Stream_0'] ['packets'][0]`
`packets[i] ['stream_id']`	`uint32`	Stream ID	`packet['stream_id']`
`packets[i] ['timestamp']`	`uint64`	Timestamp	`packet['timestamp']`
`packets[i] ['frequency']`	`uint64`	Frequency (Hz)	`packet['frequency']`
`packets[i] ['bandwidth']`	`uint32`	Bandwidth (Hz)	`packet['bandwidth']`
`packets[i] ['iq_data']`	`np.ndarray`	Complex IQ samples	`packet['iq_data']`
`streams['Stream_X']['all_iq']`	`np.ndarray`	Tất cả IQ nối lại	`stream['all_iq']`

---

7. LƯU Ý QUAN TRỌNG

10.**Kiểu dữ liệu**:

- - `iq_data` là `complex128` (số phức 128-bit)
- - Các trường header là `uint32` hoặc `uint64`

11.**Cấu trúc packet**:

- - Header: 40 bytes (header, stream_id, timestamp, frequency, len, bandwidth, switch_id, sample_cnt)
- - Payload: IQ samples (Int16, xen kẽ I và Q)

12.**Xử lý dữ liệu**:

- - Luôn kiểm tra `packet['iq_data']` có rỗng không trước khi sử dụng
- - Sử dụng `numpy` để xử lý các phép toán trên mảng

13.**Hiệu năng**:

- - Đọc toàn bộ file có thể mất thời gian nếu có nhiều packets
- - Có thể tối ưu bằng cách chỉ xử lý một số streams/packets cần thiết

---

8. TÀI LIỆU THAM KHẢO

- **Code MATLAB gốc**: `read_iq_ethernet_h5_verge.m`
- **Code Python**: `reader_iqethernet_h5.py`
- **HDF5 Documentation**: https://www.hdfgroup.org/solutions/hdf5/

Ngày tạo báo cáo: 2026-01-25

Phiên bản reader: 1.0

Tương thích với: MATLAB `read_iq_ethernet_h5_verge.m`

==========
==========

PHẦN 2

==========

BÁO CÁO CHI TIẾT: CẤU TRÚC FILE IQTCP.H5

1. TỔNG QUAN

File `iqtcp.h5` là file HDF5 chứa dữ liệu IQ (In-phase và Quadrature) từ Narrowband TCP. File này được đọc bởi module Python `reader_iqtcp_h5.py` dựa trên code MATLAB `read_iqtcp_h5_verge2.m`.

Thông tin cơ bản:

- **Định dạng**: HDF5 (Hierarchical Data Format version 5)
- **Loại dữ liệu**: IQ samples (Narrowband TCP)
- **Số sessions**: 46,072 sessions
- **Cấu trúc chính**:
- - `/attribute`: Chứa metadata (thông tin chung, DDC, request, ...)
- - `/session`: Chứa dữ liệu IQ của các sessions

---

2. CẤU TRÚC FILE H5

2.1. Cấu trúc tổng thể

```
iqtcp.h5
├── /attribute                    # Group chứa metadata
│   ├── Attributes (trực tiếp)    # → global_info
│   ├── /ddc                      # → ddc_info
│   │   └── Attributes
│   ├── /request                  # → request_info
│   │   └── Attributes
│   └── ... (các group con khác)
│
└── /session                      # Group chứa dữ liệu IQ
    ├── /000000000000000000       # Session ID
```

```
│   ├── /i                    # Dataset: In-phase samples
│   └── /q                    # Dataset: Quadrature samples
├── /000000000000000001
│   ├── /i
│   └── /q
└── ... (46,072 sessions)
```

2.2. Chi tiết các thành phần

A. `/attribute` Group

Group này chứa tất cả metadata của file:

14.**Attributes trực tiếp tại `/attribute`**:

- - Được đọc vào `data['global_info']`
- - Chứa thông tin chung như: `client_ip`, `frequency`, `bandwidth`, `channel`, `mission`, ...

15.**Sub-groups trong `/attribute`**:

- - Mỗi sub-group được đọc vào `data['{name}_info']`
- - Ví dụ: `/attribute/ddc` → `data['ddc_info']`
- - Ví dụ: `/attribute/request` → `data['request_info']`
- - Các sub-group có thể chứa:
- - Attributes (metadata)
- - Datasets (dữ liệu bổ sung, ví dụ: labels)

B. `/session` Group

Group này chứa dữ liệu IQ của tất cả sessions:

- Mỗi session có ID dạng: `000000000000000000`, `000000000000000001`, ...
- Mỗi session chứa 2 datasets:
- - `i`: In-phase samples (kiểu int32, shape: (512,))
- - `q`: Quadrature samples (kiểu int32, shape: (512,))

---

3. INPUT VÀ OUTPUT CỦA READER

3.1. Input

Hàm: `read_iqtcp_h5(filename: str)`

Tham số:

- `filename` (str): Đường dẫn đến file H5 cần đọc
- - Ví dụ: ``'../../00_DATA_h5/iqtcp.h5'``

3.2. Output

Kiểu trả về: `Dict[str, Any]`

Cấu trúc output:

```
{
    'global_info': {
        # Dictionary chứa tất cả attributes từ /attribute
        # Ví dụ:
        # 'client_ip': '192.168.1.100',
        # 'frequency': 1000000000.0,  # Hz
        # 'bandwidth': 20000000.0,    # Hz
        # 'channel': 1,
        # 'mission': 'surveillance',
        # ...
    },

    'ddc_info': {
        # Dictionary chứa attributes từ /attribute/ddc
        # Ví dụ:
        # 'channelIndex': 0,
        # 'frequency': 950000000.0,
        # 'deviceId': 'device_001',
```

```
        # ...
    },

    'request_info': {
        # Dictionary chứa attributes từ /attribute/request
        # Ví dụ:
        # 'fileName': 'data_20240113.h5',
        # 'duration': 3600.0,  # seconds
        # 'checkpoint': True,
        # ...
    },

    # Các info khác (nếu có):
    # 'label_info': {...},
    # 'other_info': {...},

    'sessions': [
        {
            'id': '000000000000000000',  # Session ID (string)
            'i': np.array([...]),        # In-phase samples (numpy array, int32)
            'q': np.array([...]),        # Quadrature samples (numpy array,
int32)
            'iq': np.array([...])        # Complex IQ = I + j*Q (numpy array,
complex128)
        },
        {
            'id': '000000000000000001',
            'i': np.array([...]),
            'q': np.array([...]),
            'iq': np.array([...])
        },
        # ... (46,072 sessions)
    ]
}
```

---

4. HƯỚNG DẪN SỬ DỤNG READER

4.1. Cài đặt

Yêu cầu:

- Python 3.6+
- Thư viện: `h5py`, `numpy`

Cài đặt dependencies:

```
pip install h5py numpy
```

4.2. Cách sử dụng cơ bản

```
from reader_iqtcp_h5 import read_iqtcp_h5

# Đọc file
filename = 'path/to/iqtcp.h5'
data = read_iqtcp_h5(filename)

# Truy cập thông tin
print(f"Số sessions: {len(data['sessions'])}")
print(f"Frequency: {data['global_info']['frequency']} Hz")
```

4.3. Cách lấy các trường thông tin output

A. Lấy thông tin chung (Global Info)

```
# Lấy toàn bộ global_info
global_info = data['global_info']

# Lấy từng trường cụ thể
client_ip = data['global_info']['client_ip']
frequency = data['global_info']['frequency']  # Hz
bandwidth = data['global_info']['bandwidth']  # Hz
channel = data['global_info']['channel']
mission = data['global_info']['mission']

# In ra tất cả các trường
for key, value in data['global_info'].items():
    print(f"{key}: {value}")
```

B. Lấy thông tin DDC

```
# Lấy toàn bộ ddc_info
ddc_info = data['ddc_info']

# Lấy từng trường cụ thể
channel_index = data['ddc_info']['channelIndex']
ddc_frequency = data['ddc_info']['frequency']
device_id = data['ddc_info']['deviceId']
```

C. Lấy thông tin Request

```
# Lấy toàn bộ request_info
request_info = data['request_info']

# Lấy từng trường cụ thể
file_name = data['request_info']['fileName']
duration = data['request_info']['duration']  # seconds
checkpoint = data['request_info']['checkpoint']
```

D. Lấy dữ liệu IQ từ Sessions

1. Lấy một session cụ thể:

```
# Lấy session đầu tiên
session_0 = data['sessions'][0]

# Lấy session theo index
session_idx = 100
session_100 = data['sessions'][session_idx]

# Lấy session theo ID
target_id = '0000000000000000100'
session = next((s for s in data['sessions'] if s['id'] == target_id), None)
```

2. Lấy dữ liệu I, Q, và IQ:

```
# Lấy session
session = data['sessions'][0]

# Lấy Session ID
session_id = session['id']
print(f"Session ID: {session_id}")

# Lấy dữ liệu I (In-phase)
i_data = session['i']  # numpy array, shape: (512,), dtype: int32
print(f"I data: shape={i_data.shape}, dtype={i_data.dtype}")
print(f"I samples: {i_data[:5]}")  # 5 giá trị đầu

# Lấy dữ liệu Q (Quadrature)
q_data = session['q']  # numpy array, shape: (512,), dtype: int32
print(f"Q data: shape={q_data.shape}, dtype={q_data.dtype}")
```

```
print(f"Q samples: {q_data[:5]}")  # 5 giá trị đầu

# Lấy dữ liệu IQ phức (I + j*Q)
iq_data = session['iq']  # numpy array, shape: (512,), dtype: complex128
print(f"IQ data: shape={iq_data.shape}, dtype={iq_data.dtype}")
print(f"IQ samples: {iq_data[:5]}")  # 5 giá trị đầu

# Tính toán từ IQ phức
import numpy as np

# Biên độ (Magnitude)
magnitude = np.abs(iq_data)
print(f"Magnitude: min={magnitude.min():.2f}, max={magnitude.max():.2f}")

# Phase (Góc pha)
phase = np.angle(iq_data)
print(f"Phase: min={phase.min():.3f}, max={phase.max():.3f}")

# Power
power = np.abs(iq_data) ** 2
print(f"Power: mean={power.mean():.2f}")
```

3. Duyệt qua tất cả sessions:

```
# Duyệt qua tất cả sessions
for i, session in enumerate(data['sessions']):
    session_id = session['id']
    i_data = session['i']
    q_data = session['q']
    iq_data = session['iq']

    # Xử lý dữ liệu...
    magnitude = np.abs(iq_data)
    phase = np.angle(iq_data)

    # Ví dụ: chỉ xử lý 10 sessions đầu
    if i >= 10:
        break
```

4. Lọc sessions có dữ liệu:

```
# Chỉ lấy các sessions có dữ liệu I và Q
valid_sessions = [
    s for s in data['sessions']
    if s['i'] is not None and s['q'] is not None
    and len(s['i']) > 0 and len(s['q']) > 0
]
```

```
print(f"Số sessions có dữ liệu: {len(valid_sessions)}")
```

5. Lấy thống kê của một session:

```
session = data['sessions'][0]

if session['i'] is not None and len(session['i']) > 0:
    i_stats = {
        'min': session['i'].min(),
        'max': session['i'].max(),
        'mean': session['i'].mean(),
        'std': session['i'].std(),
        'size': len(session['i'])
    }
    print(f"I statistics: {i_stats}")

if session['q'] is not None and len(session['q']) > 0:
    q_stats = {
        'min': session['q'].min(),
        'max': session['q'].max(),
        'mean': session['q'].mean(),
        'std': session['q'].std(),
        'size': len(session['q'])
    }
    print(f"Q statistics: {q_stats}")

if session['iq'] is not None and len(session['iq']) > 0:
    iq_mag = np.abs(session['iq'])
    iq_stats = {
        'magnitude_min': iq_mag.min(),
        'magnitude_max': iq_mag.max(),
        'magnitude_mean': iq_mag.mean(),
        'phase_min': np.angle(session['iq']).min(),
        'phase_max': np.angle(session['iq']).max(),
        'size': len(session['iq'])
    }
    print(f"IQ statistics: {iq_stats}")
```

---

5. VÍ DỤ CODE HOÀN CHỈNH

```
#!/usr/bin/env python3
"""
```

```
Ví dụ sử dụng reader_iqtcp_h5.py
"""

import numpy as np
from reader_iqtcp_h5 import read_iqtcp_h5

# 1. Đọc file
filename = '../../00_DATA_h5/iqtcp.h5'
print(f"Đang đọc file: {filename}")
data = read_iqtcp_h5(filename)

# 2. Hiển thị thông tin chung
print("\n=== THÔNG TIN CHUNG ===")
print(f"Số sessions: {len(data['sessions'])}")

if 'global_info' in data:
    print("\nGlobal Info:")
    for key, value in data['global_info'].items():
        print(f"  {key}: {value}")

if 'ddc_info' in data:
    print("\nDDC Info:")
    for key, value in data['ddc_info'].items():
        print(f"  {key}: {value}")

if 'request_info' in data:
    print("\nRequest Info:")
    for key, value in data['request_info'].items():
        print(f"  {key}: {value}")

# 3. Xử lý dữ liệu từ session đầu tiên
print("\n=== XỬ LÝ SESSION ĐẦU TIÊN ===")
if data['sessions']:
    session = data['sessions'][0]
    print(f"Session ID: {session['id']}")

    if session['i'] is not None:
        print(f"I: shape={session['i'].shape}, dtype={session['i'].dtype}")
        print(f"  Min={session['i'].min()}, Max={session['i'].max()},
Mean={session['i'].mean():.2f}")

    if session['q'] is not None:
        print(f"Q: shape={session['q'].shape}, dtype={session['q'].dtype}")
        print(f"  Min={session['q'].min()}, Max={session['q'].max()},
Mean={session['q'].mean():.2f}")

    if session['iq'] is not None:
        iq_mag = np.abs(session['iq'])
        print(f"IQ: shape={session['iq'].shape}, dtype={session['iq'].dtype}")
        print(f"  Magnitude: Min={iq_mag.min():.2f}, Max={iq_mag.max():.2f},
Mean={iq_mag.mean():.2f}")
        print(f"  Phase: Min={np.angle(session['iq']).min():.3f},
Max={np.angle(session['iq']).max():.3f}")

# 4. Duyệt qua một số sessions
```

```
print("\n=== DUYỆT QUA 5 SESSIONS ĐẦU ===")
for i in range(min(5, len(data['sessions']))):
    session = data['sessions'][i]
    if session['iq'] is not None and len(session['iq']) > 0:
        iq_mag = np.abs(session['iq'])
        print(f"Session {i} (ID: {session['id']}): "
              f"Magnitude mean={iq_mag.mean():.2f}, "
              f"Size={len(session['iq'])}")
```

---

6. BẢNG TÓM TẮT CẤU TRÚC OUTPUT

Trường	Kiểu dữ liệu	Mô tả	Ví dụ truy cập
`global_info`	**`dict`**	**Attributes từ `/attribute`**	**`data['global_info']['frequency']`**
`ddc_info`	`dict`	Attributes từ `/attribute/ddc`	`data['ddc_info']['channelIndex']`
`request_info`	`dict`	Attributes từ `/attribute/request`	`data['request_info']['fileName']`
`sessions`	`list`	Danh sách các sessions	`data['sessions'][0]`
`sessions[i]['id']`	`str`	ID của session thứ i	`data['sessions'][0]['id']`
`sessions[i]['i']`	`np.ndarray`	In-phase samples (int32)	`data['sessions'][0]['i']`
`sessions[i]['q']`	`np.ndarray`	Quadrature samples (int32)	`data['sessions'][0]['q']`
`sessions[i]['iq']`	`np.ndarray`	Complex IQ = I + j*Q (complex128)	`data['sessions'][0]['iq']`

---

7. LƯU Ý QUAN TRỌNG

16.**Kiểu dữ liệu**:

- - `i` và `q` là `int32` (số nguyên 32-bit)
- - `iq` là `complex128` (số phức 128-bit, được tính từ I và Q)

17.**Kích thước**:

- - Mỗi session có 512 samples (I và Q)
- - File có 46,072 sessions

18.**Xử lý dữ liệu**:

- - Luôn kiểm tra `session['i']` và `session['q']` có `None` hoặc rỗng không trước khi sử dụng
- - Sử dụng `numpy` để xử lý các phép toán trên mảng

19.**Hiệu năng**:

- - Đọc toàn bộ file có thể mất thời gian (46,072 sessions)
- - Có thể tối ưu bằng cách chỉ đọc một số sessions cần thiết

---

8. TÀI LIỆU THAM KHẢO

- **Code MATLAB gốc**: `read_iqtcp_h5_verge2.m`
- **Code Python**: `reader_iqtcp_h5.py`
- **File test**: `test_reader_iqtcp.py`
- **HDF5 Documentation**: https://www.hdfgroup.org/solutions/hdf5/

Ngày tạo báo cáo: 2025-01-XX

Phiên bản reader: 1.0

Tương thích với: MATLAB `read_iqtcp_h5_verge2.m`

==========

PHẦN 3

==========

BÁO CÁO CHI TIẾT: CẤU TRÚC FILE SPECTRUM.H5

1. TỔNG QUAN

File `spectrum.h5` là file HDF5 chứa dữ liệu Spectrum. File này được đọc bởi module Python `reader_spectrum_h5.py` dựa trên code MATLAB `read_spectrum_data.m`.

Thông tin cơ bản:

- **Định dạng**: HDF5 (Hierarchical Data Format version 5)
- **Loại dữ liệu**: Spectrum samples (decoded)
- **Số sessions**: 1598 sessions
- **Cấu trúc chính**:
- - `/attribute`: Chứa metadata (thông tin chung)
- - `/session`: Chứa dữ liệu spectrum của các sessions

---

2. CẤU TRÚC FILE H5

2.1. Cấu trúc tổng thể

```
spectrum.h5
├── /attribute                    # Group chứa metadata
│   └── Attributes (trực tiếp)    # → global_info
│
└── /session                      # Group chứa dữ liệu spectrum
    ├── /000xx                    # Session ID
    │   ├── Attributes            # → sessions[i].attributes
    │   ├── /source               # → sessions[i].source_info
    │   │   └── Attributes
    │   └── /sample_decoded        # Dataset: Decoded spectrum samples
    ├── /000yy
    │   ├── Attributes
    │   ├── /source
    │   │   └── Attributes
    │   └── /sample_decoded
    └── ... (nhiều sessions)
```

2.2. Chi tiết các thành phần

A. `/attribute` Group

Group này chứa tất cả metadata của file:

20.**Attributes trực tiếp tại `/attribute`**:

- - Được đọc vào `data['global_info']`
- - Chứa thông tin chung như: `client_ip`, `frequency`, `bandwidth`, `channel`, `mission`, ...

B. `/session` Group

Group này chứa dữ liệu spectrum của các sessions:

- Mỗi session có ID dạng: `000xx`, `000yy`, ...
- Mỗi session chứa:
- - **Attributes**: Thông tin chung của session (timestamp, freq, bw, ...)
- - **`/source` sub-group**: Thông tin thiết bị (attributes)
- - **`/sample_decoded` dataset**: Vector dữ liệu spectrum đã giải mã (float64 array)

---

3. INPUT VÀ OUTPUT CỦA READER

3.1. Input

Hàm: `read_spectrum_h5(filename: str)`

Tham số:

- `filename` (str): Đường dẫn đến file H5 cần đọc
- - Ví dụ: ``'../../00_DATA_h5/spectrum.h5'``

3.2. Output

Kiểu trả về: `Dict[str, Any]`

Cấu trúc output:

```
{
    'global_info': {
        # Dictionary chứa tất cả attributes từ /attribute
        # Ví dụ:
        # 'client_ip': '192.168.1.100',
        # 'frequency': 1000000000.0,  # Hz
        # 'bandwidth': 20000000.0,    # Hz
        # 'mission': '...',
        # ...
    },

    'sessions': [
        {
            'id': '000xx',                    # Session ID
            'attributes': {                  # Attributes của session
                'timestamp': ...,
                'frequency': ...,              # Hz
                'bandwidth': ...,              # Hz
```

```
                # ... các attributes khác
            },
            'source_info': {                # Thông tin thiết bị
                # Attributes từ /source
                # ...
            },
            'samples': np.ndarray            # Vector dữ liệu spectrum
(float64)
        },
        # ... (nhiều sessions)
    ]
}
```

---

4. HƯỚNG DẪN SỬ DỤNG READER

4.1. Cài đặt

Yêu cầu:

- Python 3.6+
- Thư viện: `h5py`, `numpy`

Cài đặt dependencies:

```
pip install h5py numpy
```

4.2. Cách sử dụng cơ bản

```
from reader_spectrum_h5 import read_spectrum_h5

# Đọc file
filename = 'path/to/spectrum.h5'
data = read_spectrum_h5(filename)

# Truy cập thông tin
print(f"Số sessions: {len(data['sessions'])}")
if 'global_info' in data and 'frequency' in data['global_info']:
    print(f"Frequency: {data['global_info']['frequency']} Hz")
```

4.3. Cách lấy các trường thông tin output

A. Lấy thông tin chung (Global Info)

```
# Lấy toàn bộ global_info
global_info = data['global_info']

# Lấy từng trường cụ thể
if 'client_ip' in data['global_info']:
    client_ip = data['global_info']['client_ip']
if 'frequency' in data['global_info']:
    frequency = data['global_info']['frequency']  # Hz
if 'bandwidth' in data['global_info']:
    bandwidth = data['global_info']['bandwidth']  # Hz

# In ra tất cả các trường
for key, value in data['global_info'].items():
    print(f"{key}: {value}")
```

B. Lấy dữ liệu Sessions

1. Lấy danh sách tất cả sessions:

```
# Lấy số lượng sessions
num_sessions = len(data['sessions'])
print(f"Có {num_sessions} sessions")

# Lấy session đầu tiên
first_session = data['sessions'][0]
```

2. Lấy thông tin từ một session:

```
# Lấy session theo index
session = data['sessions'][0]

# Lấy Session ID
session_id = session['id']
print(f"Session ID: {session_id}")

# Lấy attributes của session
attributes = session['attributes']
if 'frequency' in attributes:
    freq = attributes['frequency']  # Hz
```

```
if 'bandwidth' in attributes:
    bw = attributes['bandwidth']  # Hz
if 'timestamp' in attributes:
    timestamp = attributes['timestamp']
```

3. Lấy source info:

```
# Lấy session
session = data['sessions'][0]

# Lấy source info
source_info = session['source_info']
# Truy cập các trường trong source_info
# Ví dụ: device_name = source_info.get('device_name', None)
```

4. Lấy dữ liệu samples (spectrum):

```
import numpy as np

# Lấy session
session = data['sessions'][0]

# Lấy dữ liệu spectrum
samples = session['samples']  # numpy array, dtype: float64
print(f"Samples: shape={samples.shape}, dtype={samples.dtype}")
print(f"Samples (5 đầu): {samples[:5]}")

# Tính toán từ samples
# Min, Max, Mean
print(f"Min: {samples.min():.2f}, Max: {samples.max():.2f}, Mean:
{samples.mean():.2f}")

# Power (nếu cần)
power = samples ** 2
print(f"Power: mean={power.mean():.2f}")
```

5. Duyệt qua tất cả sessions:

```
import numpy as np

# Duyệt qua tất cả sessions
for i, session in enumerate(data['sessions']):
```

```
    session_id = session['id']
    attributes = session['attributes']
    source_info = session['source_info']
    samples = session['samples']

    # Xử lý dữ liệu...
    if len(samples) > 0:
        print(f"Session {i} ({session_id}): {len(samples)} samples")
        print(f"  Frequency: {attributes.get('frequency', 'N/A')} Hz")
        print(f"  Min: {samples.min():.2f}, Max: {samples.max():.2f}")

    # Ví dụ: chỉ xử lý 10 sessions đầu
    if i >= 10:
        break
```

5. VÍ DỤ CODE HOÀN CHỈNH

```
#!/usr/bin/env python3
"""
Ví dụ sử dụng reader_spectrum_h5.py
"""

import numpy as np
from reader_spectrum_h5 import read_spectrum_h5

# 1. Đọc file
filename = '../../00_DATA_h5/spectrum.h5'
print(f"Đang đọc file: {filename}")
data = read_spectrum_h5(filename)

# 2. Hiển thị thông tin chung
print("\n=== THÔNG TIN CHUNG ===")
print(f"Số sessions: {len(data['sessions'])}")

if 'global_info' in data:
    print("\nGlobal Info:")
    for key, value in data['global_info'].items():
        print(f"  {key}: {value}")

# 3. Xử lý dữ liệu từ session đầu tiên
print("\n=== XỬ LÝ SESSION ĐẦU TIÊN ===")
if data['sessions']:
    session = data['sessions'][0]
    print(f"Session ID: {session['id']}")

    # Attributes
    if session['attributes']:
        print("\nAttributes:")
        for key, value in session['attributes'].items():
            print(f"  {key}: {value}")
```

```
    # Source info
    if session['source_info']:
        print("\nSource Info:")
        for key, value in session['source_info'].items():
            print(f"  {key}: {value}")

    # Samples
    if len(session['samples']) > 0:
        samples = session['samples']
        print(f"\nSamples: {len(samples)} điểm")
        print(f"  Min: {samples.min():.2f}, Max: {samples.max():.2f}, Mean:
{samples.mean():.2f}")
```

---

6. BẢNG TÓM TẮT CẤU TRÚC OUTPUT

Trường	Kiểu dữ liệu	Mô tả	Ví dụ truy cập
`global_info`	**`dict`**	**Attributes từ `/attribute`**	**`data['global_info'] ['frequency']`**
`sessions`	`list`	Danh sách các sessions	`data['sessions'] [0]`
`sessions[i] ['id']`	`str`	Session ID	`session['id']`
`sessions[i] ['attributes']`	`dict`	Attributes của session	`session['attributes']['frequency']`
`sessions[i] ['source_info']`	`dict`	Thông tin thiết bị	`session['source_info'] ['device_name']`
`sessions[i] ['samples']`	`np.ndarray`	Vector spectrum samples (float64)	`session['samples']`

---

7. LƯU Ý QUAN TRỌNG

21.**Kiểu dữ liệu**:

- - `samples` là `float64` (số thực double precision)
- - Các attributes có thể là string, int, float tùy theo file

22.**Cấu trúc session**:

- - Mỗi session có thể có hoặc không có `/source` sub-group

- - Mỗi session có thể có hoặc không có `/sample_decoded` dataset

23.**Xử lý dữ liệu**:

- - Luôn kiểm tra `len(session['samples']) > 0` trước khi sử dụng
- - Sử dụng `numpy` để xử lý các phép toán trên mảng
- - Kiểm tra `session['attributes']` và `session['source_info']` có rỗng không

24.**Hiệu năng**:

- - Đọc toàn bộ file có thể mất thời gian nếu có nhiều sessions
- - Có thể tối ưu bằng cách chỉ xử lý một số sessions cần thiết

---

8. TÀI LIỆU THAM KHẢO

- **Code MATLAB gốc**: `read_spectrum_data.m`
- **Code Python**: `reader_spectrum_h5.py`
- **HDF5 Documentation**: https://www.hdfgroup.org/solutions/hdf5/

Ngày tạo báo cáo: 2026-01-25

Phiên bản reader: 1.0

Tương thích với: MATLAB `read_spectrum_data.m`

==========
==========

PHẦN 4

==========
==========

BÁO CÁO CHI TIẾT: CẤU TRÚC FILE HISTOGRAM.H5

1. TỔNG QUAN

File `histogram.h5` là file HDF5 chứa dữ liệu Histogram. File này được đọc bởi module Python `reader_histogram_h5.py` dựa trên code MATLAB `read_histogram_h5_multitype.m`.

Thông tin cơ bản:

- **Định dạng**: HDF5 (Hierarchical Data Format version 5)
- **Loại dữ liệu**: Histogram samples (decoded)
- **Số sessions**: 6003 sessions
- - AccumulatedPower: 3002 sessions
- - CrossingThresholdPower: 3001 sessions
- **Cấu trúc chính**:
- - `/attribute`: Chứa metadata (thông tin chung)
- - `/session`: Chứa dữ liệu histogram của các sessions (hỗ trợ nhiều loại message)

---

2. CẤU TRÚC FILE H5

2.1. Cấu trúc tổng thể

```
histogram.h5
├── /attribute                    # Group chứa metadata
│   └── Attributes (trực tiếp)    # → global_info
│
└── /session                      # Group chứa dữ liệu histogram
    ├── /000xx                    # Session ID
    │   ├── Attributes            # → sessions[i].attributes (bao gồm
message_type)
    │   ├── /context              # → sessions[i].context_info
    │   │   └── Attributes
    │   ├── /source               # → sessions[i].source_info
    │   │   └── Attributes
    │   └── Dataset (phụ thuộc message_type):
    │       ├── sample_decoded        # Cho AccumulatedPower
    │       ├── acc_sample_decoded    # Cho CrossingThresholdPower
    │       └── crx_sample_decoded    # Cho CrossingThresholdPower
    ├── /000yy
    │   ├── Attributes
    │   ├── /context
    │   │   └── Attributes
    │   ├── /source
    │   │   └── Attributes
    │   └── Dataset (phụ thuộc message_type)
    └── ... (nhiều sessions)
```

2.2. Chi tiết các thành phần

A. `/attribute` Group

Group này chứa tất cả metadata của file:

25.**Attributes trực tiếp tại `/attribute`**:

- - Được đọc vào `data['global_info']`
- - Chứa thông tin chung như: `client_ip`, `frequency`, `bandwidth`, `channel`, `mission`, ...

B. `/session` Group

Group này chứa dữ liệu histogram của các sessions:

- Mỗi session có ID dạng: `000xx`, `000yy`, ...
- Mỗi session chứa:
- - **Attributes**: Thông tin chung của session (timestamp, freq, bw, **message_type**, ...)
- - **`/context` sub-group**: Thông tin ngữ cảnh (attributes)
- - **`/source` sub-group**: Thông tin thiết bị (attributes)
- - **Dataset phụ thuộc vào `message_type`**:
- - **AccumulatedPower**: `sample_decoded` (vector histogram)
- - **CrossingThresholdPower**: `acc_sample_decoded` (accumulated) VÀ `crx_sample_decoded` (crossing)

---

3. INPUT VÀ OUTPUT CỦA READER

3.1. Input

Hàm: `read_histogram_h5(filename: str)`

Tham số:

- `filename` (str): Đường dẫn đến file H5 cần đọc
- - Ví dụ: `'../../00_DATA_h5/histogram.h5'`

3.2. Output

Kiểu trả về: `Dict[str, Any]`

Cấu trúc output:

```
{
    'global_info': {
        # Dictionary chứa tất cả attributes từ /attribute
        # Ví dụ:
        # 'client_ip': '192.168.1.100',
```

```
        # 'frequency': 1000000000.0,  # Hz
        # 'bandwidth': 20000000.0,    # Hz
        # 'mission': '...',
        # ...
    },

    'sessions': [
        {
            'id': '000xx',                    # Session ID
            'type': 'AccumulatedPower',       # Message type (hoặc
'CrossingThresholdPower')
            'attributes': {                  # Attributes của session
                'message_type': '...',
                'timestamp': ...,
                'frequency': ...,              # Hz
                'bandwidth': ...,             # Hz
                # ... các attributes khác
            },
            'context_info': {                # Thông tin ngữ cảnh
                # Attributes từ /context
                # ...
            },
            'source_info': {                  # Thông tin thiết bị
                # Attributes từ /source
                # ...
            },
            # Dữ liệu phụ thuộc vào message_type:
            'sample_decoded': np.ndarray,     # Cho AccumulatedPower (float64)
            'acc_sample_decoded': np.ndarray, # Cho CrossingThresholdPower
(float64)
            'crx_sample_decoded': np.ndarray  # Cho CrossingThresholdPower
(float64)
        },
        # ... (nhiều sessions)
    ]
}
```

---

4. HƯỚNG DẪN SỬ DỤNG READER

4.1. Cài đặt

Yêu cầu:

- Python 3.6+
- Thư viện: `h5py`, `numpy`

Cài đặt dependencies:

```
pip install h5py numpy
```

4.2. Cách sử dụng cơ bản

```
from reader_histogram_h5 import read_histogram_h5

# Đọc file
filename = 'path/to/histogram.h5'
data = read_histogram_h5(filename)

# Truy cập thông tin
print(f"Số sessions: {len(data['sessions'])}")
if 'global_info' in data and 'frequency' in data['global_info']:
    print(f"Frequency: {data['global_info']['frequency']} Hz")
```

4.3. Cách lấy các trường thông tin output

A. Lấy thông tin chung (Global Info)

```
# Lấy toàn bộ global_info
global_info = data['global_info']

# Lấy từng trường cụ thể
if 'client_ip' in data['global_info']:
    client_ip = data['global_info']['client_ip']
if 'frequency' in data['global_info']:
    frequency = data['global_info']['frequency']  # Hz
if 'bandwidth' in data['global_info']:
    bandwidth = data['global_info']['bandwidth']  # Hz

# In ra tất cả các trường
for key, value in data['global_info'].items():
    print(f"{key}: {value}")
```

B. Lấy dữ liệu Sessions

1. Lấy danh sách tất cả sessions:

```
# Lấy số lượng sessions
num_sessions = len(data['sessions'])
print(f"Có {num_sessions} sessions")

# Lấy session đầu tiên
first_session = data['sessions'][0]
```

2. Lấy thông tin từ một session:

```
# Lấy session theo index
session = data['sessions'][0]

# Lấy Session ID
session_id = session['id']
print(f"Session ID: {session_id}")

# Lấy Message Type
msg_type = session['type']
print(f"Message Type: {msg_type}")

# Lấy attributes của session
attributes = session['attributes']
if 'frequency' in attributes:
    freq = attributes['frequency']  # Hz
if 'bandwidth' in attributes:
    bw = attributes['bandwidth']  # Hz
if 'timestamp' in attributes:
    timestamp = attributes['timestamp']
```

3. Lấy context info:

```
# Lấy session
session = data['sessions'][0]

# Lấy context info
context_info = session['context_info']
# Truy cập các trường trong context_info
# Ví dụ: context_value = context_info.get('some_field', None)
```

4. Lấy source info:

```
# Lấy session
session = data['sessions'][0]

# Lấy source info
source_info = session['source_info']
# Truy cập các trường trong source_info
# Ví dụ: device_name = source_info.get('device', None)
```

5. Lấy dữ liệu histogram (phụ thuộc vào message_type):

```
import numpy as np

# Lấy session
session = data['sessions'][0]
msg_type = session['type']

# Xử lý theo loại message
if 'CrossingThresholdPower' in msg_type:
    # TRƯỜNG HỢP: CrossingThresholdPower
    # Đọc acc_sample_decoded
    acc_data = session['acc_sample_decoded']  # numpy array, dtype: float64
    print(f"acc_sample_decoded: shape={acc_data.shape}, dtype={acc_data.dtype}")
    print(f"  Min: {acc_data.min():.2e}, Max: {acc_data.max():.2e}, Sum:
{acc_data.sum():.2e}")

    # Đọc crx_sample_decoded
    crx_data = session['crx_sample_decoded']  # numpy array, dtype: float64
    print(f"crx_sample_decoded: shape={crx_data.shape}, dtype={crx_data.dtype}")
    print(f"  Min: {crx_data.min():.2e}, Max: {crx_data.max():.2e}, Sum:
{crx_data.sum():.2e}")
else:
    # TRƯỜNG HỢP: AccumulatedPower (hoặc mặc định)
    hist_data = session['sample_decoded']  # numpy array, dtype: float64
    print(f"sample_decoded: shape={hist_data.shape}, dtype={hist_data.dtype}")
    print(f"  Min: {hist_data.min():.2e}, Max: {hist_data.max():.2e}, Sum:
{hist_data.sum():.2e}")
```

6. Duyệt qua tất cả sessions:

```
import numpy as np

# Duyệt qua tất cả sessions
for i, session in enumerate(data['sessions']):
    session_id = session['id']
    msg_type = session['type']
    attributes = session['attributes']
    context_info = session['context_info']
```

```
    source_info = session['source_info']

    # Xử lý dữ liệu theo loại message
    if 'CrossingThresholdPower' in msg_type:
        acc_data = session['acc_sample_decoded']
        crx_data = session['crx_sample_decoded']

        if len(acc_data) > 0:
            print(f"Session {i} ({session_id}): CrossingThresholdPower")
            print(f"  acc: {len(acc_data)} bins, Sum={acc_data.sum():.2e}")
            print(f"  crx: {len(crx_data)} bins, Sum={crx_data.sum():.2e}")
    else:
        hist_data = session['sample_decoded']

        if len(hist_data) > 0:
            print(f"Session {i} ({session_id}): AccumulatedPower")
            print(f"  samples: {len(hist_data)} bins,
Sum={hist_data.sum():.2e}")

    # Ví dụ: chỉ xử lý 10 sessions đầu
    if i >= 10:
        break
```

5. VÍ DỤ CODE HOÀN CHỈNH

```
#!/usr/bin/env python3
"""
Ví dụ sử dụng reader_histogram_h5.py
"""

import numpy as np
from reader_histogram_h5 import read_histogram_h5

# 1. Đọc file
filename = '../../00_DATA_h5/histogram.h5'
print(f"Đang đọc file: {filename}")
data = read_histogram_h5(filename)

# 2. Hiển thị thông tin chung
print("\n=== THÔNG TIN CHUNG ===")
print(f"Số sessions: {len(data['sessions'])}")

if 'global_info' in data:
    print("\nGlobal Info:")
    for key, value in data['global_info'].items():
        print(f"  {key}: {value}")

# 3. Xử lý dữ liệu từ session đầu tiên
print("\n=== XỬ LÝ SESSION ĐẦU TIÊN ===")
if data['sessions']:
    session = data['sessions'][0]
```

```
    print(f"Session ID: {session['id']}")
    print(f"Message Type: {session['type']}")

    # Attributes
    if session['attributes']:
        print("\nAttributes:")
        for key, value in session['attributes'].items():
            print(f"  {key}: {value}")

    # Context info
    if session['context_info']:
        print("\nContext Info:")
        for key, value in session['context_info'].items():
            print(f"  {key}: {value}")

    # Source info
    if session['source_info']:
        print("\nSource Info:")
        for key, value in session['source_info'].items():
            print(f"  {key}: {value}")

    # Samples (phụ thuộc vào message_type)
    msg_type = session['type']
    if 'CrossingThresholdPower' in msg_type:
        if len(session['acc_sample_decoded']) > 0:
            acc_data = session['acc_sample_decoded']
            print(f"\nacc_sample_decoded: {len(acc_data)} bins")
            print(f"  Min: {acc_data.min():.2e}, Max: {acc_data.max():.2e}, Sum:
{acc_data.sum():.2e}")

        if len(session['crx_sample_decoded']) > 0:
            crx_data = session['crx_sample_decoded']
            print(f"\ncrx_sample_decoded: {len(crx_data)} bins")
            print(f"  Min: {crx_data.min():.2e}, Max: {crx_data.max():.2e}, Sum:
{crx_data.sum():.2e}")
    else:
        if len(session['sample_decoded']) > 0:
            hist_data = session['sample_decoded']
            print(f"\nsample_decoded: {len(hist_data)} bins")
            print(f"  Min: {hist_data.min():.2e}, Max: {hist_data.max():.2e},
Sum: {hist_data.sum():.2e}")
```

---

6. BẢNG TÓM TẮT CẤU TRÚC OUTPUT

Trường	Kiểu dữ liệu	Mô tả	Ví dụ truy cập
`global_info`	`dict`	Attributes từ	`data['global_in

		`/attribute`	fo'] ['frequency']`
`sessions`	`list`	Danh sách các sessions	`data['sessions'][0]`
`sessions[i]['id']`	`str`	Session ID	`session['id']`
`sessions[i]['type']`	`str`	Message type	`session['type']`
`sessions[i]['attributes']`	`dict`	Attributes của session	`session['attributes']['frequency']`
`sessions[i]['context_info']`	`dict`	Thông tin ngữ cảnh	`session['context_info']['field']`
`sessions[i]['source_info']`	`dict`	Thông tin thiết bị	`session['source_info']['device']`
`sessions[i]['sample_decoded']`	`np.ndarray`	Histogram (AccumulatedPower)	`session['sample_decoded']`
`sessions[i]['acc_sample_decoded']`	`np.ndarray`	Accumulated histogram (CrossingThresholdPower)	`session['acc_sample_decoded']`
`sessions[i]['crx_sample_decoded']`	`np.ndarray`	Crossing histogram (CrossingThresholdPower)	`session['crx_sample_decoded']`

---

7. LƯU Ý QUAN TRỌNG

26.**Kiểu dữ liệu**:

- - Tất cả samples là `float64` (số thực double precision)
- - Các attributes có thể là string, int, float tùy theo file

27.**Message Type**:

- - **AccumulatedPower**: Sử dụng `sample_decoded`
- - **CrossingThresholdPower**: Sử dụng `acc_sample_decoded` VÀ `crx_sample_decoded`
- - Luôn kiểm tra `session['type']` trước khi truy cập dữ liệu

28.**Cấu trúc session**:

- - Mỗi session có thể có hoặc không có `/context` sub-group
- - Mỗi session có thể có hoặc không có `/source` sub-group

- - Dataset phụ thuộc vào `message_type` trong attributes

29.**Xử lý dữ liệu**:

- - Luôn kiểm tra `len(session['sample_decoded']) > 0` (hoặc tương ứng) trước khi sử dụng
- - Sử dụng `numpy` để xử lý các phép toán trên mảng
- - Kiểm tra `session['attributes']`, `session['context_info']`, `session['source_info']` có rỗng không

30.**Hiệu năng**:

- - Đọc toàn bộ file có thể mất thời gian nếu có nhiều sessions
- - Có thể tối ưu bằng cách chỉ xử lý một số sessions cần thiết

---

8. TÀI LIỆU THAM KHẢO

- **Code MATLAB gốc**: `read_histogram_h5_multitype.m`
- **Code Python**: `reader_histogram_h5.py`
- **HDF5 Documentation**: https://www.hdfgroup.org/solutions/hdf5/

Ngày tạo báo cáo: 2026-01-25

Phiên bản reader: 1.0

Tương thích với: MATLAB `read_histogram_h5_multitype.m`

==========
==========

PHẦN 5

==========
==========

BÁO CÁO CHI TIẾT: CẤU TRÚC FILE DF.H5

1. TỔNG QUAN

File `df.h5` là file HDF5 chứa dữ liệu DF/DOA (Direction Finding / Direction of Arrival). File này được đọc bởi module Python `reader_df_h5.py` dựa trên code MATLAB `reader_df.m`.

Thông tin cơ bản:

- **Định dạng**: HDF5 (Hierarchical Data Format version 5)
- **Loại dữ liệu**: DF/DOA data (Pulses, DOA vectors, Calibration)
- **Số sessions**: 53 sessions
- **Số bảng calibration**: 3 tables
- **Số nhóm configuration**: 5 groups
- **Cấu trúc chính**:
- - `/attribute/configuration`: Chứa cấu hình (attributes)
- - `/attribute/calibration/calibs`: Chứa dữ liệu hiệu chuẩn (datasets)
- - `/session`: Chứa dữ liệu pulses và DOA của các sessions

---

2. CẤU TRÚC FILE H5

2.1. Cấu trúc tổng thể

```
df.h5
├── /attribute
│   ├── /configuration              # → configuration
│   │   ├── /antParams               # → configuration['antParams'] (attributes)
│   │   ├── /filterParams            # → configuration['filterParams']
(attributes)
│   │   └── ... (các sub-groups khác)
│   │
│   └── /calibration
│       └── /calibs                  # → calibration
│           ├── /0                   # → calibration['Table_0'] (datasets: pow1,
dps...)
│           ├── /1                   # → calibration['Table_1'] (datasets)
│           └── ... (các bảng khác)
│
└── /session                         # → sessions
    ├── /000xx                       # Session ID
    │   ├── Datasets (pulses)        # → sessions[i].pulses (amp, fc, bw...)
    │   └── /doa
    │       └── /doa
    │           ├── /0               # → sessions[i].doa['Target_0']
    │           │   ├── /position    # → position (datasets: vecDoas...)
    │           │   ├── /velocity     # → velocity (datasets: velocDoas...)
    │           │   └── /identity
    │           │       └── /features # → identity_features (datasets: meanBws,
meanFcs...)
    │           ├── /1               # → sessions[i].doa['Target_1']
    │           │   └── ...
    │           └── ... (các targets khác)
    ├── /000yy
    │   └── ...
    └── ... (nhiều sessions)
```

2.2. Chi tiết các thành phần

A. `/attribute/configuration` Group

Group này chứa các cấu hình dưới dạng attributes:

- Mỗi sub-group (antParams, filterParams...) chứa attributes
- Được đọc vào `data['configuration'][sub_name]`
- Ví dụ: `data['configuration']['antParams']` chứa các attributes của antParams

B. `/attribute/calibration/calibs` Group

Group này chứa các bảng hiệu chuẩn dưới dạng datasets:

- Mỗi sub-group (0, 1, 2...) chứa datasets (pow1, dps...)
- Được đọc vào `data['calibration']['Table_0']`, `data['calibration']['Table_1']`, ...
- Mỗi bảng chứa các datasets như: pow1, dps, ...

C. `/session` Group

Group này chứa dữ liệu sessions:

- Mỗi session có ID dạng: `000xx`, `000yy`, ...
- Mỗi session chứa:
- - **Pulses datasets**: Trực tiếp tại session (amp, fc, bw...)
- - **DOA data**: Cấu trúc lồng nhau `/doa/doa/0,1,2...`
- - Mỗi target (0, 1, 2...) có:
- - `position`: Datasets (vecDoas...)
- - `velocity`: Datasets (velocDoas...)
- - `identity_features`: Datasets (meanBws, meanFcs...)

---

3. INPUT VÀ OUTPUT CỦA READER

3.1. Input

Hàm: `read_df_h5(filename: str)`

Tham số:

- `filename` (str): Đường dẫn đến file H5 cần đọc
- - Ví dụ: ``'../../00_DATA_h5/df.h5'``

3.2. Output

Kiểu trả về: `Dict[str, Any]`

Cấu trúc output:

```
{
    'configuration': {
        'antParams': {                    # Attributes từ
/attribute/configuration/antParams
            'attr1': value1,
            'attr2': value2,
            # ...
        },
        'filterParams': {                  # Attributes từ
/attribute/configuration/filterParams
            # ...
        },
        # ... (các sub-groups khác)
    },

    'calibration': {
        'Table_0': {                       # Datasets từ
/attribute/calibration/calibs/0
            'pow1': np.ndarray,
            'dps': np.ndarray,
            # ... (các datasets khác)
        },
        'Table_1': {                       # Datasets từ
/attribute/calibration/calibs/1
            # ...
        },
        # ... (các bảng khác)
    },

    'sessions': [
        {
            'id': '000xx',                  # Session ID
            'pulses': {                    # Datasets từ session (pulses)
                'amp': np.ndarray,
                'fc': np.ndarray,           # Frequency center
                'bw': np.ndarray,           # Bandwidth
                # ... (các datasets khác)
            },
```

```
            'doa': {
                'Target_0': {
                    'position': {           # Datasets từ /doa/doa/0/position
                        'vecDoas': np.ndarray,  # DOA vectors
                        # ... (các datasets khác)
                    },
                    'velocity': {           # Datasets từ /doa/doa/0/velocity
                        'velocDoas': np.ndarray,
                        # ... (các datasets khác)
                    },
                    'identity_features': {  # Datasets từ
/doa/doa/0/identity/features
                        'meanBws': np.ndarray,
                        'meanFcs': np.ndarray,
                        # ... (các datasets khác)
                    }
                },
                'Target_1': {...},
                # ... (các targets khác)
            }
        },
        # ... (nhiều sessions)
    ]
}
```

---

4. HƯỚNG DẪN SỬ DỤNG READER

4.1. Cài đặt

Yêu cầu:

- Python 3.6+
- Thư viện: `h5py`, `numpy`

Cài đặt dependencies:

```
pip install h5py numpy
```

4.2. Cách sử dụng cơ bản

```
from reader_df_h5 import read_df_h5

# Đọc file
filename = 'path/to/df.h5'
data = read_df_h5(filename)

# Truy cập thông tin
print(f"Số sessions: {len(data['sessions'])}")
print(f"Số bảng calibration: {len(data.get('calibration', {}))}")
```

4.3. Cách lấy các trường thông tin output

A. Lấy thông tin Configuration

```
# Lấy toàn bộ configuration
configuration = data['configuration']

# Lấy antParams
if 'antParams' in data['configuration']:
    ant_params = data['configuration']['antParams']
    # Truy cập các attributes
    # attr_value = ant_params.get('attr_name', None)

# Lấy filterParams
if 'filterParams' in data['configuration']:
    filter_params = data['configuration']['filterParams']
    # ...

# Duyệt qua tất cả configuration groups
for group_name, group_attrs in data['configuration'].items():
    print(f"{group_name}: {len(group_attrs)} attributes")
```

B. Lấy dữ liệu Calibration

```
# Lấy toàn bộ calibration
calibration = data['calibration']

# Lấy Table_0
if 'Table_0' in data['calibration']:
    table_0 = data['calibration']['Table_0']

    # Lấy pow1
    if 'pow1' in table_0:
        pow1 = table_0['pow1']  # numpy array
        print(f"pow1: shape={pow1.shape}, dtype={pow1.dtype}")

    # Lấy dps
```

```
    if 'dps' in table_0:
        dps = table_0['dps']  # numpy array
        print(f"dps: shape={dps.shape}, dtype={dps.dtype}")

# Duyệt qua tất cả calibration tables
for table_name, table_data in data['calibration'].items():
    print(f"{table_name}: {len(table_data)} datasets")
    for ds_name, ds_value in table_data.items():
        if isinstance(ds_value, np.ndarray):
            print(f"  {ds_name}: shape={ds_value.shape},
dtype={ds_value.dtype}")
```

C. Lấy dữ liệu Sessions

1. Lấy danh sách tất cả sessions:

```
# Lấy số lượng sessions
num_sessions = len(data['sessions'])
print(f"Có {num_sessions} sessions")

# Lấy session đầu tiên
first_session = data['sessions'][0]
```

2. Lấy thông tin pulses từ một session:

```
# Lấy session theo index
session = data['sessions'][0]

# Lấy Session ID
session_id = session['id']
print(f"Session ID: {session_id}")

# Lấy pulses
pulses = session['pulses']

# Lấy các trường cụ thể
if 'fc' in pulses:
    fc = pulses['fc']  # Frequency center
    print(f"Frequency center: {len(fc)} pulses")
    print(f"  Min: {fc.min():.2f}, Max: {fc.max():.2f}")

if 'bw' in pulses:
    bw = pulses['bw']  # Bandwidth
    print(f"Bandwidth: {len(bw)} pulses")
```

```
if 'amp' in pulses:
    amp = pulses['amp']  # Amplitude
    print(f"Amplitude: {len(amp)} pulses")
```

3. Lấy dữ liệu DOA từ một session:

```
# Lấy session
session = data['sessions'][0]

# Lấy DOA
doa = session['doa']

# Lấy Target_0
if 'Target_0' in doa:
    target_0 = doa['Target_0']

    # Lấy position (vecDoas)
    if 'position' in target_0:
        position = target_0['position']
        if 'vecDoas' in position:
            vec_doas = position['vecDoas']  # numpy array
            print(f"DOA vectors: shape={vec_doas.shape},
dtype={vec_doas.dtype}")
            # vec_doas có thể là 2D array: [n_samples, n_dimensions]

    # Lấy velocity (velocDoas)
    if 'velocity' in target_0:
        velocity = target_0['velocity']
        if 'velocDoas' in velocity:
            veloc_doas = velocity['velocDoas']  # numpy array
            print(f"Velocity DOA: shape={veloc_doas.shape},
dtype={veloc_doas.dtype}")

    # Lấy identity_features
    if 'identity_features' in target_0:
        identity = target_0['identity_features']
        if 'meanBws' in identity:
            mean_bws = identity['meanBws']  # numpy array
            print(f"Mean BWs: shape={mean_bws.shape}, dtype={mean_bws.dtype}")

        if 'meanFcs' in identity:
            mean_fcs = identity['meanFcs']  # numpy array
            print(f"Mean FCs: shape={mean_fcs.shape}, dtype={mean_fcs.dtype}")
```

4. Duyệt qua tất cả sessions và targets:

```
import numpy as np

# Duyệt qua tất cả sessions
for i, session in enumerate(data['sessions']):
    session_id = session['id']
    pulses = session['pulses']
    doa = session['doa']

    print(f"\nSession {i} ({session_id}):")

    # Pulses
    if 'fc' in pulses:
        print(f"  Pulses: {len(pulses['fc'])} xung")

    # DOA targets
    print(f"  DOA Targets: {len(doa)} targets")
    for target_name, target_data in doa.items():
        print(f"    {target_name}:")
        if 'position' in target_data and 'vecDoas' in target_data['position']:
            vec = target_data['position']['vecDoas']
            print(f"      Position vectors: {vec.shape}")
        if 'identity_features' in target_data:
            print(f"      Identity features:
{len(target_data['identity_features'])} datasets")

    # Ví dụ: chỉ xử lý 5 sessions đầu
    if i >= 5:
        break
```

5. VÍ DỤ CODE HOÀN CHỈNH

```
#!/usr/bin/env python3
"""
Ví dụ sử dụng reader_df_h5.py
"""

import numpy as np
from reader_df_h5 import read_df_h5

# 1. Đọc file
filename = '../../00_DATA_h5/df.h5'
print(f"Đang đọc file: {filename}")
data = read_df_h5(filename)

# 2. Hiển thị thông tin chung
print("\n=== THÔNG TIN CHUNG ===")
print(f"Số sessions: {len(data['sessions'])}")
print(f"Số bảng calibration: {len(data.get('calibration', {}))}")
print(f"Số nhóm configuration: {len(data.get('configuration', {}))}")

# 3. Configuration
```

```
if 'configuration' in data:
    print("\n=== CONFIGURATION ===")
    for group_name, group_attrs in data['configuration'].items():
        print(f"{group_name}: {len(group_attrs)} attributes")
        for attr_name in list(group_attrs.keys())[:3]:
            print(f"  {attr_name}: ...")

# 4. Calibration
if 'calibration' in data:
    print("\n=== CALIBRATION ===")
    for table_name, table_data in data['calibration'].items():
        print(f"{table_name}: {len(table_data)} datasets")
        for ds_name in list(table_data.keys())[:3]:
            ds_val = table_data[ds_name]
            if isinstance(ds_val, np.ndarray):
                print(f"  {ds_name}: shape={ds_val.shape},
dtype={ds_val.dtype}")

# 5. Session đầu tiên
print("\n=== SESSION ĐẦU TIÊN ===")
if data['sessions']:
    session = data['sessions'][0]
    print(f"Session ID: {session['id']}")

    # Pulses
    if session['pulses']:
        print("\nPulses:")
        for pulse_name, pulse_val in session['pulses'].items():
            if isinstance(pulse_val, np.ndarray):
                print(f"  {pulse_name}: shape={pulse_val.shape},
dtype={pulse_val.dtype}")

    # DOA
    if session['doa']:
        print("\nDOA:")
        for target_name, target_data in session['doa'].items():
            print(f"  {target_name}:")
            if 'position' in target_data:
                print(f"    position: {len(target_data['position'])} datasets")
            if 'velocity' in target_data:
                print(f"    velocity: {len(target_data['velocity'])} datasets")
            if 'identity_features' in target_data:
                print(f"    identity_features:
{len(target_data['identity_features'])} datasets")
```

---

6. BẢNG TÓM TẮT CẤU TRÚC OUTPUT

Trường	Kiểu dữ liệu	Mô tả	Ví dụ truy cập
`configuration`	**`dict`**	**Configuration groups với attributes**	**`data['configuration']['antParams']`**
`calibration`	`dict`	Calibration tables với datasets	`data['calibration']['Table_0']`
`calibration['Table_X']`	`dict`	Datasets trong bảng calibration	`data['calibration']['Table_0']['pow1']`
`sessions`	`list`	Danh sách các sessions	`data['sessions'][0]`
`sessions[i]['id']`	`str`	Session ID	`session['id']`
`sessions[i]['pulses']`	`dict`	Pulse datasets	`session['pulses']['fc']`
`sessions[i]['doa']`	`dict`	DOA targets	`session['doa']['Target_0']`
`sessions[i]['doa']['Target_X']['position']`	`dict`	Position datasets	`target['position']['vecDoas']`
`sessions[i]['doa']['Target_X']['velocity']`	`dict`	Velocity datasets	`target['velocity']['velocDoas']`
`sessions[i]['doa']['Target_X']['identity_features']`	`dict`	Identity feature datasets	`target['identity_features']['meanBws']`

---

7. LƯU Ý QUAN TRỌNG

31.**Kiểu dữ liệu**:

- - Tất cả datasets là `numpy.ndarray`
- - Configuration attributes có thể là string, int, float tùy theo file

32.**Cấu trúc lồng nhau**:

- - DOA có cấu trúc lồng nhau sâu: `/doa/doa/0/position`
- - Mỗi session có thể có nhiều targets (Target_0, Target_1, ...)
- - Mỗi target có position, velocity, và identity_features

33.**Xử lý dữ liệu**:

- - Luôn kiểm tra key có tồn tại trước khi truy cập (sử dụng `in` hoặc `.get()`)
- - Sử dụng `numpy` để xử lý các phép toán trên mảng
- - Kiểm tra shape của arrays trước khi xử lý

34.**Hiệu năng**:

- - Đọc toàn bộ file có thể mất thời gian nếu có nhiều sessions
- - Có thể tối ưu bằng cách chỉ xử lý một số sessions cần thiết

35.**Cấu trúc DOA**:

- - Cấu trúc DOA rất lồng nhau, cần chú ý khi truy cập
- - Mỗi target có thể có hoặc không có position, velocity, identity_features

---

8. TÀI LIỆU THAM KHẢO

- **Code MATLAB gốc**: `reader_df.m`
- **Code Python**: `reader_df_h5.py`
- **HDF5 Documentation**: https://www.hdfgroup.org/solutions/hdf5/

Ngày tạo báo cáo: 2026-01-25

Phiên bản reader: 1.0

Tương thích với: MATLAB `reader_df.m`

==========

PHẦN 6

==========

BÁO CÁO CHI TIẾT: CẤU TRÚC FILE DEMODULATION.H5

1. TỔNG QUAN

File `demodulation.h5` là file HDF5 chứa dữ liệu Demodulation (IQ). File này được đọc bởi module Python `reader_demodulation_h5.py` dựa trên code MATLAB `reader_demodulation_no_recursive.m`.

Thông tin cơ bản:

- **Định dạng**: HDF5 (Hierarchical Data Format version 5)
- **Loại dữ liệu**: Demodulation IQ data (In-phase và Quadrature)
- **Số sessions**: 4240 sessions
- **Số nhóm request**: 6 groups
- **Cấu trúc chính**:
- - `/attribute/request`: Chứa cấu hình (attributes từ các sub-groups)
- - `/session`: Chứa dữ liệu IQ của các sessions (datasets 'i' và 'q')

---

2. CẤU TRÚC FILE H5

2.1. Cấu trúc tổng thể

```
demodulation.h5
├── /attribute
│   └── /request                        # → request
│       ├── /hwConfiguration            # → request['hwConfiguration']
(attributes)
│       ├── /libConfiguration           # → request['libConfiguration']
(attributes)
│       ├── /recordingOptions           # → request['recordingOptions']
(attributes)
│       ├── /source                     # → request['source'] (attributes)
│       ├── /spectrumOptions            # → request['spectrumOptions']
(attributes)
│       ├── /transaction                # → request['transaction'] (attributes)
│       └── ... (các sub-groups khác)
│
└── /session                            # → sessions
    ├── /000xx                          # Session ID
    │   ├── /i                          # Dataset: In-phase samples
    │   └── /q                          # Dataset: Quadrature samples
    ├── /000yy
    │   ├── /i
    │   └── /q
    └── ... (nhiều sessions)
```

2.2. Chi tiết các thành phần

A. `/attribute/request` Group

Group này chứa các cấu hình dưới dạng attributes:

- Mỗi sub-group (hwConfiguration, libConfiguration, recordingOptions, source, spectrumOptions, transaction...) chứa attributes
- Được đọc vào `data['request'][sub_name]`
- Ví dụ: `data['request']['hwConfiguration']` chứa các attributes của hwConfiguration

B. `/session` Group

Group này chứa dữ liệu IQ của các sessions:

- Mỗi session có ID dạng: `000xx`, `000yy`, ...

- Mỗi session chứa 2 datasets:
- - **`i`**: In-phase samples (real part)
- - **`q`**: Quadrature samples (imaginary part)
- Dữ liệu được kết hợp thành complex IQ: `iq = i + j*q`

---

3. INPUT VÀ OUTPUT CỦA READER

3.1. Input

Hàm: `read_demodulation_h5(filename: str)`

Tham số:

- `filename` (str): Đường dẫn đến file H5 cần đọc
- - Ví dụ: `'../../00_DATA_h5/demodulation.h5'`

3.2. Output

Kiểu trả về: `Dict[str, Any]`

Cấu trúc output:

```
{
    'request': {
        'hwConfiguration': {                    # Attributes từ
/attribute/request/hwConfiguration
            'attr1': value1,
            'attr2': value2,
            # ...
        },
        'libConfiguration': {                   # Attributes từ
/attribute/request/libConfiguration
            # ...
        },
```

```
        'recordingOptions': {                   # Attributes từ
/attribute/request/recordingOptions
            # ...
        },
        'source': {                             # Attributes từ
/attribute/request/source
            # ...
        },
        'spectrumOptions': {                    # Attributes từ
/attribute/request/spectrumOptions
            # ...
        },
        'transaction': {                        # Attributes từ
/attribute/request/transaction
            # ...
        },
        # ... (các sub-groups khác)
    },

    'sessions': [
        {
            'id': '000xx',                      # Session ID
            'iq': np.ndarray                    # Complex IQ data (I + j*Q),
dtype: complex128
        },
        # ... (nhiều sessions)
    ]
}
```

---

4. HƯỚNG DẪN SỬ DỤNG READER

4.1. Cài đặt

Yêu cầu:

- Python 3.6+
- Thư viện: `h5py`, `numpy`

Cài đặt dependencies:

```
pip install h5py numpy
```

4.2. Cách sử dụng cơ bản

```
from reader_demodulation_h5 import read_demodulation_h5

# Đọc file
filename = 'path/to/demodulation.h5'
data = read_demodulation_h5(filename)

# Truy cập thông tin
print(f"Số sessions: {len(data['sessions'])}")
print(f"Số nhóm request: {len(data.get('request', {}))}")
```

4.3. Cách lấy các trường thông tin output

A. Lấy thông tin Request Configuration

```
# Lấy toàn bộ request
request = data['request']

# Lấy hwConfiguration
if 'hwConfiguration' in data['request']:
    hw_config = data['request']['hwConfiguration']
    # Truy cập các attributes
    # attr_value = hw_config.get('attr_name', None)

# Lấy libConfiguration
if 'libConfiguration' in data['request']:
    lib_config = data['request']['libConfiguration']
    # ...

# Lấy recordingOptions
if 'recordingOptions' in data['request']:
    rec_options = data['request']['recordingOptions']
    # ...

# Lấy source
if 'source' in data['request']:
    source = data['request']['source']
    # ...

# Lấy spectrumOptions
if 'spectrumOptions' in data['request']:
    spec_options = data['request']['spectrumOptions']
    # ...

# Lấy transaction
if 'transaction' in data['request']:
    transaction = data['request']['transaction']
    # ...
```

```
# Duyệt qua tất cả request groups
for group_name, group_attrs in data['request'].items():
    print(f"{group_name}: {len(group_attrs)} attributes")
```

B. Lấy dữ liệu Sessions

1. Lấy danh sách tất cả sessions:

```
# Lấy số lượng sessions
num_sessions = len(data['sessions'])
print(f"Có {num_sessions} sessions")

# Lấy session đầu tiên
first_session = data['sessions'][0]
```

2. Lấy thông tin từ một session:

```
# Lấy session theo index
session = data['sessions'][0]

# Lấy Session ID
session_id = session['id']
print(f"Session ID: {session_id}")

# Lấy IQ data
iq_data = session['iq']  # numpy array, dtype: complex128
print(f"IQ data: {len(iq_data)} samples, dtype={iq_data.dtype}")
```

3. Xử lý dữ liệu IQ:

```
import numpy as np

# Lấy session
session = data['sessions'][0]
iq_data = session['iq']

# Lấy I và Q riêng biệt
i_data = iq_data.real  # In-phase (real part)
q_data = iq_data.imag  # Quadrature (imaginary part)
```

```
print(f"I (real): Min={i_data.min():.2f}, Max={i_data.max():.2f},
Mean={i_data.mean():.2f}")
print(f"Q (imag): Min={q_data.min():.2f}, Max={q_data.max():.2f},
Mean={q_data.mean():.2f}")

# Tính toán từ IQ phức
# Biên độ (Magnitude)
magnitude = np.abs(iq_data)
print(f"Magnitude: Min={magnitude.min():.2f}, Max={magnitude.max():.2f},
Mean={magnitude.mean():.2f}")

# Phase (Góc pha)
phase = np.angle(iq_data)
print(f"Phase: Min={phase.min():.3f}, Max={phase.max():.3f}")

# Power
power = np.abs(iq_data) ** 2
print(f"Power: Mean={power.mean():.2f}")
```

4. Duyệt qua tất cả sessions:

```
import numpy as np

# Duyệt qua tất cả sessions
for i, session in enumerate(data['sessions']):
    session_id = session['id']
    iq_data = session['iq']

    # Xử lý dữ liệu...
    if len(iq_data) > 0:
        magnitude = np.abs(iq_data)
        print(f"Session {i} ({session_id}): {len(iq_data)} samples")
        print(f"  Magnitude: Min={magnitude.min():.2f},
Max={magnitude.max():.2f}")

    # Ví dụ: chỉ xử lý 10 sessions đầu
    if i >= 10:
        break
```

5. VÍ DỤ CODE HOÀN CHỈNH

```
#!/usr/bin/env python3
"""
Ví dụ sử dụng reader_demodulation_h5.py
"""

import numpy as np
```

```
import matplotlib.pyplot as plt
from reader_demodulation_h5 import read_demodulation_h5

# 1. Đọc file
filename = '../../00_DATA_h5/demodulation.h5'
print(f"Đang đọc file: {filename}")
data = read_demodulation_h5(filename)

# 2. Hiển thị thông tin chung
print("\n=== THÔNG TIN CHUNG ===")
print(f"Số sessions: {len(data['sessions'])}")
print(f"Số nhóm request: {len(data.get('request', {}))}")

# 3. Request Configuration
if 'request' in data:
    print("\n=== REQUEST CONFIGURATION ===")
    for group_name, group_attrs in data['request'].items():
        print(f"{group_name}: {len(group_attrs)} attributes")
        for attr_name in list(group_attrs.keys())[:3]:
            print(f"  {attr_name}: ...")

# 4. Session đầu tiên
print("\n=== SESSION ĐẦU TIÊN ===")
if data['sessions']:
    session = data['sessions'][0]
    print(f"Session ID: {session['id']}")

    if len(session['iq']) > 0:
        iq_data = session['iq']
        print(f"IQ data: {len(iq_data)} samples")
        print(f"  I (real): Min={iq_data.real.min():.2f},
Max={iq_data.real.max():.2f}")
        print(f"  Q (imag): Min={iq_data.imag.min():.2f},
Max={iq_data.imag.max():.2f}")
        print(f"  Magnitude: Min={np.abs(iq_data).min():.2f},
Max={np.abs(iq_data).max():.2f}")

        # Vẽ biểu đồ (nếu có matplotlib)
        # plt.figure(figsize=(12, 6))
        #
        # plt.subplot(2, 1, 1)
        # plt.plot(iq_data.real, 'b', label='I')
        # plt.plot(iq_data.imag, 'r', label='Q')
        # plt.title('Time Domain (I & Q)')
        # plt.legend()
        # plt.grid(True)
        #
        # plt.subplot(2, 1, 2)
        # plt.plot(iq_data.real, iq_data.imag, '.')
        # plt.title('Constellation Diagram')
        # plt.axis('equal')
        # plt.grid(True)
        #
        # plt.tight_layout()
        # plt.show()
```

---

6. BẢNG TÓM TẮT CẤU TRÚC OUTPUT

Trường	Kiểu dữ liệu	Mô tả	Ví dụ truy cập
`request`	**`dict`**	**Request configuration groups với attributes**	**`data['request']['hwConfiguration']`**
`request['hwConfiguration']`	`dict`	Hardware configuration attributes	`data['request']['hwConfiguration']['attr']`
`request['libConfiguration']`	`dict`	Library configuration attributes	`data['request']['libConfiguration']['attr']`
`request['recordingOptions']`	`dict`	Recording options attributes	`data['request']['recordingOptions']['attr']`
`request['source']`	`dict`	Source attributes	`data['request']['source']['attr']`
`request['spectrumOptions']`	`dict`	Spectrum options attributes	`data['request']['spectrumOptions']['attr']`
`request['transaction']`	`dict`	Transaction attributes	`data['request']['transaction']['attr']`
`sessions`	`list`	Danh sách các sessions	`data['sessions'][0]`
`sessions[i]['id']`	`str`	Session ID	`session['id']`
`sessions[i]['iq']`	`np.ndarray`	Complex IQ data (I + j*Q)	`session['iq']`

---

7. LƯU Ý QUAN TRỌNG

36.**Kiểu dữ liệu**:

- - `iq` là `complex128` (số phức 128-bit)
- - I và Q được đọc từ datasets riêng biệt và kết hợp thành complex
- - Request attributes có thể là string, int, float tùy theo file

37.**Cấu trúc session**:

- - Mỗi session có 2 datasets: `i` và `q`
- - Dữ liệu được kết hợp: `iq = i + j*q`
- - Nếu `i` và `q` có độ dài khác nhau, chỉ lấy phần chung (min length)

38.**Xử lý dữ liệu**:

- - Luôn kiểm tra `len(session['iq']) > 0` trước khi sử dụng
- - Sử dụng `numpy` để xử lý các phép toán trên mảng
- - Sử dụng `.real` và `.imag` để lấy I và Q riêng biệt
- - Sử dụng `np.abs()` và `np.angle()` để tính magnitude và phase

39.**Hiệu năng**:

- - Đọc toàn bộ file có thể mất thời gian nếu có nhiều sessions
- - Có thể tối ưu bằng cách chỉ xử lý một số sessions cần thiết

40.**Visualization**:

- - Có thể vẽ time domain (I và Q theo thời gian)
- - Có thể vẽ constellation diagram (Q vs I)

---

8. TÀI LIỆU THAM KHẢO

- **Code MATLAB gốc**: `reader_demodulation_no_recursive.m`
- **Code Python**: `reader_demodulation_h5.py`
- **HDF5 Documentation**: https://www.hdfgroup.org/solutions/hdf5/

Ngày tạo báo cáo: 2026-01-25

Phiên bản reader: 1.0

Tương thích với: MATLAB `reader_demodulation_no_recursive.m`

==========

PHẦN 7

==========

BÁO CÁO CHI TIẾT: CẤU TRÚC FILE IDENTIFIER.H5

1. TỔNG QUAN

File `identifier.h5` là file HDF5 chứa dữ liệu Identifier. File này được đọc bởi module Python `reader_identifier_h5.py` dựa trên code MATLAB `read_identifier.m`.

Thông tin cơ bản:

- **Định dạng**: HDF5 (Hierarchical Data Format version 5)
- **Loại dữ liệu**: Identifier data (Hop parameters, DOA, IQ)
- **Số sessions**: 750 sessions
- **Cấu trúc chính**:
- - `/attribute/estm_bdw`: Chứa tham số Hop (datasets: fc, ...)
- - `/attribute/request/label`: Chứa label text (dataset)
- - `/attribute/doa/position`: Chứa DOA position datasets (vecDoas, ...)
- - `/attribute/doa/identity/features`: Chứa identity feature datasets (meanBws, meanFcs, ...)
- - `/session`: Chứa dữ liệu IQ của các sessions (dataset 'iq' xen kẽ I, Q, I, Q...)

---

2. CẤU TRÚC FILE H5

2.1. Cấu trúc tổng thể

```
identifier.h5
├── /attribute
│   ├── /estm_bdw                    # → estm_bdw (datasets: fc, ...)
│   │   └── Datasets                 # Tham số Hop
│   │
│   ├── /request                     # → request
│   │   └── /label                   # Dataset: Label text
│   │
│   └── /doa                         # → doa
│       ├── /position                # → doa['position'] (datasets:
vecDoas, ...)
│       └── /identity
│           └── /features            # → doa['identity']['features']
(datasets: meanBws, meanFcs, ...)
│
└── /session                         # → sessions
    ├── /000xx                       # Session ID
    │   └── /iq                      # Dataset: IQ data (xen kẽ I, Q, I, Q...)
    ├── /000yy
    │   └── /iq
    └── ... (nhiều sessions)
```

2.2. Chi tiết các thành phần

A. `/attribute/estm_bdw` Group

Group này chứa tham số Hop dưới dạng datasets:

- Các datasets như: `fc` (frequency center), ...
- Được đọc vào `data['estm_bdw']`
- Ví dụ: `data['estm_bdw']['fc']` chứa frequency centers của các hops

B. `/attribute/request/label` Dataset

Dataset này chứa label text:

- Được đọc và parse thành dictionary
- Được đọc vào `data['request']['label']`
- Format: text với các dòng dạng `key=value` hoặc plain text

C. `/attribute/doa` Group

Group này chứa dữ liệu DOA:

- **`/position`**: Datasets như `vecDoas` (DOA vectors)
- **`/identity/features`**: Datasets như `meanBws`, `meanFcs` (identity features)

D. `/session` Group

Group này chứa dữ liệu IQ của các sessions:

- Mỗi session có ID dạng: `000xx`, `000yy`, ...
- Mỗi session chứa 1 dataset:
- - **`iq`**: IQ data xen kẽ (I, Q, I, Q, ...)
- - Dữ liệu được xử lý thành complex IQ: `iq = I + j*Q`

---

3. INPUT VÀ OUTPUT CỦA READER

3.1. Input

Hàm: `read_identifier_h5(filename: str)`

Tham số:

- `filename` (str): Đường dẫn đến file H5 cần đọc
- - Ví dụ: `'../../00_DATA_h5/identifier.h5'`

3.2. Output

Kiểu trả về: `Dict[str, Any]`

Cấu trúc output:

```
{
    'estm_bdw': {
        'fc': np.ndarray,              # Frequency centers (Hop parameters)
        # ... (các datasets khác)
    },

    'request': {
        'label': {                     # Parsed label dictionary
            'key1': 'value1',
            'key2': 'value2',
            # ... (các key-value pairs từ label text)
        }
    },

    'doa': {
        'position': {
            'vecDoas': np.ndarray,     # DOA vectors
            # ... (các datasets khác)
        },
        'identity': {
            'features': {
                'meanBws': np.ndarray, # Mean bandwidths
                'meanFcs': np.ndarray,  # Mean frequency centers
                # ... (các datasets khác)
            }
        }
    },

    'sessions': [
        {
            'id': '000xx',              # Session ID
            'iq': np.ndarray            # Complex IQ data (I + j*Q), dtype:
complex128
        },
        # ... (nhiều sessions)
```

```
    ]
}
```

---

4. HƯỚNG DẪN SỬ DỤNG READER

4.1. Cài đặt

Yêu cầu:

- Python 3.6+
- Thư viện: `h5py`, `numpy`

Cài đặt dependencies:

```
pip install h5py numpy
```

4.2. Cách sử dụng cơ bản

```
from reader_identifier_h5 import read_identifier_h5

# Đọc file
filename = 'path/to/identifier.h5'
data = read_identifier_h5(filename)

# Truy cập thông tin
print(f"Số sessions: {len(data['sessions'])}")
print(f"Có estm_bdw: True")
print(f"Có doa: True")
```

4.3. Cách lấy các trường thông tin output

A. Lấy thông tin estm_bdw (Hop Parameters)

```
# Lấy estm_bdw
if 'estm_bdw' in data:
```

```
    estm_bdw = data['estm_bdw']

    # Lấy fc (frequency centers)
    if 'fc' in estm_bdw:
        fc = estm_bdw['fc']  # numpy array
        print(f"Frequency centers: {len(fc)} hops")
        print(f"  Min: {fc.min():.2f}, Max: {fc.max():.2f}")

    # Duyệt qua tất cả datasets
    for ds_name, ds_value in estm_bdw.items():
        if isinstance(ds_value, np.ndarray):
            print(f"{ds_name}: shape={ds_value.shape}, dtype={ds_value.dtype}")
```

B. Lấy thông tin Request Label

```
# Lấy request label
if 'request' in data and 'label' in data['request']:
    label = data['request']['label']

    # Truy cập các key-value pairs
    for key, value in label.items():
        print(f"{key}: {value}")

    # Lấy giá trị cụ thể nếu biết key
    # value = label.get('key_name', None)
```

C. Lấy dữ liệu DOA

```
# Lấy DOA
if 'doa' in data:
    doa = data['doa']

    # Lấy position (vecDoas)
    if 'position' in doa:
        position = doa['position']
        if 'vecDoas' in position:
            vec_doas = position['vecDoas']  # numpy array
            print(f"DOA vectors: shape={vec_doas.shape},
dtype={vec_doas.dtype}")

    # Lấy identity features
    if 'identity' in doa and 'features' in doa['identity']:
        features = doa['identity']['features']

        if 'meanBws' in features:
            mean_bws = features['meanBws']  # numpy array
            print(f"Mean BWs: shape={mean_bws.shape}, dtype={mean_bws.dtype}")
```

```
        if 'meanFcs' in features:
            mean_fcs = features['meanFcs']  # numpy array
            print(f"Mean FCs: shape={mean_fcs.shape}, dtype={mean_fcs.dtype}")
```

D. Lấy dữ liệu Sessions

1. Lấy danh sách tất cả sessions:

```
# Lấy số lượng sessions
num_sessions = len(data['sessions'])
print(f"Có {num_sessions} sessions")

# Lấy session đầu tiên
first_session = data['sessions'][0]
```

2. Lấy thông tin từ một session:

```
# Lấy session theo index
session = data['sessions'][0]

# Lấy Session ID
session_id = session['id']
print(f"Session ID: {session_id}")

# Lấy IQ data
iq_data = session['iq']  # numpy array, dtype: complex128
print(f"IQ data: {len(iq_data)} samples, dtype={iq_data.dtype}")
```

3. Xử lý dữ liệu IQ:

```
import numpy as np

# Lấy session
session = data['sessions'][0]
iq_data = session['iq']

# Lấy I và Q riêng biệt
i_data = iq_data.real  # In-phase (real part)
q_data = iq_data.imag  # Quadrature (imaginary part)

print(f"I (real): Min={i_data.min():.2f}, Max={i_data.max():.2f},
```

```
Mean={i_data.mean():.2f}")
print(f"Q (imag): Min={q_data.min():.2f}, Max={q_data.max():.2f},
Mean={q_data.mean():.2f}")

# Tính toán từ IQ phức
# Biên độ (Magnitude)
magnitude = np.abs(iq_data)
print(f"Magnitude: Min={magnitude.min():.2f}, Max={magnitude.max():.2f},
Mean={magnitude.mean():.2f}")

# Phase (Góc pha)
phase = np.angle(iq_data)
print(f"Phase: Min={phase.min():.3f}, Max={phase.max():.3f}")

# Power
power = np.abs(iq_data) ** 2
print(f"Power: Mean={power.mean():.2f}")
```

4. Duyệt qua tất cả sessions:

```
import numpy as np

# Duyệt qua tất cả sessions
for i, session in enumerate(data['sessions']):
    session_id = session['id']
    iq_data = session['iq']

    # Xử lý dữ liệu...
    if len(iq_data) > 0:
        magnitude = np.abs(iq_data)
        print(f"Session {i} ({session_id}): {len(iq_data)} samples")
        print(f"  Magnitude: Min={magnitude.min():.2f},
Max={magnitude.max():.2f}")

    # Ví dụ: chỉ xử lý 10 sessions đầu
    if i >= 10:
        break
```

5. VÍ DỤ CODE HOÀN CHỈNH

```
#!/usr/bin/env python3
"""
Ví dụ sử dụng reader_identifier_h5.py
"""

import numpy as np
from reader_identifier_h5 import read_identifier_h5
```

```
# 1. Đọc file
filename = '../../00_DATA_h5/identifier.h5'
print(f"Đang đọc file: {filename}")
data = read_identifier_h5(filename)

# 2. Hiển thị thông tin chung
print("\n=== THÔNG TIN CHUNG ===")
print(f"Số sessions: {len(data['sessions'])}")
print(f"Có estm_bdw: True")
print(f"Có request: True")
print(f"Có doa: True")

# 3. estm_bdw (Hop Parameters)
if 'estm_bdw' in data:
    print("\n=== ESTM_BDW (HOP PARAMETERS) ===")
    for ds_name, ds_value in data['estm_bdw'].items():
        if isinstance(ds_value, np.ndarray):
            print(f"{ds_name}: shape={ds_value.shape}, dtype={ds_value.dtype}")
            if ds_name == 'fc' and len(ds_value) > 0:
                print(f"  Min: {ds_value.min():.2f}, Max: {ds_value.max():.2f}")

# 4. Request Label
if 'request' in data and 'label' in data['request']:
    print("\n=== REQUEST LABEL ===")
    label = data['request']['label']
    for key, value in list(label.items())[:10]:  # Hiển thị 10 dòng đầu
        print(f"{key}: {value}")

# 5. DOA
if 'doa' in data:
    print("\n=== DOA ===")
    if 'position' in data['doa']:
        print("Position:")
        for ds_name, ds_value in data['doa']['position'].items():
            if isinstance(ds_value, np.ndarray):
                print(f"  {ds_name}: shape={ds_value.shape},
dtype={ds_value.dtype}")

    if 'identity' in data['doa'] and 'features' in data['doa']['identity']:
        print("Identity Features:")
        for ds_name, ds_value in data['doa']['identity']['features'].items():
            if isinstance(ds_value, np.ndarray):
                print(f"  {ds_name}: shape={ds_value.shape},
dtype={ds_value.dtype}")

# 6. Session đầu tiên
print("\n=== SESSION ĐẦU TIÊN ===")
if data['sessions']:
    session = data['sessions'][0]
    print(f"Session ID: {session['id']}")

    if len(session['iq']) > 0:
        iq_data = session['iq']
        print(f"IQ data: {len(iq_data)} samples")
```

```
        print(f"  I (real): Min={iq_data.real.min():.2f},
Max={iq_data.real.max():.2f}")
        print(f"  Q (imag): Min={iq_data.imag.min():.2f},
Max={iq_data.imag.max():.2f}")
        print(f"  Magnitude: Min={np.abs(iq_data).min():.2f},
Max={np.abs(iq_data).max():.2f}")
```

---

6. BẢNG TÓM TẮT CẤU TRÚC OUTPUT

Trường	Kiểu dữ liệu	Mô tả	Ví dụ truy cập
`estm_bdw`	**`dict`**	**Hop parameters datasets**	**`data['estm_bdw']['fc']`**
`request`	`dict`	Request info	`data['request']['label']`
`request['label']`	`dict`	Parsed label dictionary	`data['request']['label']['key']`
`doa`	`dict`	DOA data	`data['doa']['position']`
`doa['position']`	`dict`	Position datasets	`data['doa']['position']['vecDoas']`
`doa['identity']['features']`	`dict`	Identity feature datasets	`data['doa']['identity']['features']['meanBws']`
`sessions`	`list`	Danh sách các sessions	`data['sessions'][0]`
`sessions[i]['id']`	`str`	Session ID	`session['id']`
`sessions[i]['iq']`	`np.ndarray`	Complex IQ data (I + j*Q)	`session['iq']`

---

7. LƯU Ý QUAN TRỌNG

41.**Kiểu dữ liệu**:

- - `iq` là `complex128` (số phức 128-bit)
- - IQ được đọc từ dataset xen kẽ (I, Q, I, Q...) và xử lý thành complex
- - Các datasets khác có thể là float, int tùy theo file

42.**Cấu trúc IQ**:

- - Dataset `iq` chứa dữ liệu xen kẽ: [I, Q, I, Q, ...]
- - Được xử lý thành complex: `iq = I + j*Q`
- - Nếu số lượng phần tử lẻ, phần tử cuối sẽ bị bỏ qua

43.**Label parsing**:

- - Label được parse từ text thành dictionary
- - Format: các dòng dạng `key=value` hoặc plain text
- - Plain text được lưu với key dạng `line_1`, `line_2`, ...

44.**Xử lý dữ liệu**:

- - Luôn kiểm tra `len(session['iq']) > 0` trước khi sử dụng
- - Sử dụng `numpy` để xử lý các phép toán trên mảng
- - Sử dụng `.real` và `.imag` để lấy I và Q riêng biệt
- - Sử dụng `np.abs()` và `np.angle()` để tính magnitude và phase

45.**Hiệu năng**:

- - Đọc toàn bộ file có thể mất thời gian nếu có nhiều sessions
- - Có thể tối ưu bằng cách chỉ xử lý một số sessions cần thiết

---

8. TÀI LIỆU THAM KHẢO

- **Code MATLAB gốc**: `read_identifier.m`
- **Code Python**: `reader_identifier_h5.py`
- **HDF5 Documentation**: https://www.hdfgroup.org/solutions/hdf5/

Ngày tạo báo cáo: 2026-01-25

Phiên bản reader: 1.0

Tương thích với: MATLAB `read_identifier.m`
