# Mini-Guide: Organizing and Analyzing Sensor Data

## 1 Introduction

This guide explains how to organize your measurement files, how metadata is extracted from filenames and folder structure, and what each analysis function does. It is aimed at users with minimal Python experience.

## 2 File Organization

Files are organized in a hierarchical folder structure as follows:

```
GroupBySample/
    WXXXX/          % Wafer folder
        YY/         % Chiplet folder (e.g., C3,B2,B4)
            YXX/    % Structure folder (e.g., B6, A1)
                +.txt
```

**Explanation of folders:**

- `WXXXX`: Wafer ID.
- `YY`: Chiplet number (individual chip on the wafer).
- `YXX`: Structure or die type within the chiplet (e.g., B6, A1).

## 3 File Naming Convention

Each measurement file encodes metadata in its filename:

- `WXXXX`: Wafer ID.
- `xXXyYY`: Die coordinates on the wafer.
- `WBXX`: Wheatstone bridge number.
- `RbXX`: Reference resistance number.
- `CW`: Sweep direction (clockwise or counterclockwise).

**Rules for writing filenames:**

- Always start with the wafer ID and die coordinates.
- Include measurement type (WB/RB) next.
- Indicate sweep direction at the end.
- Example: `W9122_x01y05_WB2_Rb1_CW.txt`.

## 4 From Filename and Folders to Metadata

The folder structure and filenames are automatically transformed into metadata fields:

- **Wafer ID:** Extracted from the top-level folder (`WXXXX`).
- **Chiplet:** Extracted from the second-level folder (`YY`).
- **Structure/Die type:** Extracted from the third-level folder (`YXX`).
- **Die coordinates:** Extracted from the filename (`xXXyYY`).
- **Wheatstone bridge / reference resistance:** From the filename (`WBXX`, `RBXX`).
- **Sweep direction:** From the filename (`CW` or `CCW`).

## 5 Analysis Functions and Their Purpose

- **Load Data:** Reads the `.txt` files into structured tables containing angle, voltage, current, and resistance.
- **Compute WB Outputs:** Calculates derived Wheatstone bridge outputs:
  - `WB_out_same`: Difference between channels of the same bridge.
  - `WB_out_crossed`: Difference between channels of paired bridges (half bridges of different elements with the same RL magnetic moment).
- **Compute Ratios and Angles:** Uses WB outputs to determine:
  - Sin/cos ratio for a bridge pair.
  - Computed angle of the magnetic field.
  - Difference between computed and applied angles.
- **Filter and Group Samples:** Allows selection by metadata:
  - By wafer, chiplet, structure, die, or measurement type.

  - Useful for comparing multiple dies or bridges under similar conditions.

- **Plotting Functions:** Visualize:
  - WB outputs for a chiplet or die.
  - MR curves for given structures.
  - Computed angles and angular errors.
  - Bridge imbalances for quality control.
- **Debug Reading:** Optional function to inspect raw data lines to verify correct parsing.

## 6 Workflow Summary for Users

1. Organize your files according to the hierarchy: `GroupBySample/Wafer/Chiplet/Structure/`.
2. Name files following the convention: `WXXXX_xXXyYY_WBXX_RBXX_CW.txt`.
3. Load the folder into the analysis tool.
4. Compute WB outputs and derived quantities (ratios, angles, angular differences).
5. Filter or group samples by wafer, chiplet, or structure as needed.
6. Use plotting functions to visualize bridge behavior, MR curves, and angle errors.

This workflow allows users to process multiple samples consistently, compare dies and chiplets, and visualize sensor performance without needing to write or modify Python code.

## 7 Detailed Function Guide

This section describes all available functions in the Python analysis tools, including their arguments and expected behavior. Users can refer to this guide to know when and how to call each function.

### 7.1 SampleMetadata.from_filename(filepath)

**Purpose:** Converts a file path into structured metadata.

**Arguments:**

- `filepath`: Path to a `.txt` measurement file.

**Extracted Metadata Fields:**

- `wafer`: Wafer ID (from folder name).
- `anneal`: Annealing identifier (from filename).
- `die`: Die/folder name (immediate parent folder).
- `x_coord, y_coord`: Die coordinates (from filename).
- `coord_type`: Label mapping of y-coordinate (e.g., 'A1').
- `measurement`: WB or RB number (from filename).
- `direction`: Sweep direction ('CW' or 'CCW').

**When to use:** Automatically called when loading files to generate metadata.

## 7.2 MeasurementData(filepath, skiprows=8)

**Purpose:** Loads a single measurement file into a structured table.
**Arguments:**

- `filepath`: Path to the `.txt` file.
- `skiprows`: Number of initial text lines to skip (default 8).

**What it does:**

- Reads columns: `angle`, `smu_curr`, `smu_volt`, `DMM1`, `DMM2`.
- Converts values to numeric.
- Stores metadata via `SampleMetadata`.

## 7.3 ExperimentDataset.add(sample)

**Purpose:** Adds a `MeasurementData` sample to the dataset.
**Arguments:**

- `sample`: Instance of `MeasurementData`.

## 7.4 ExperimentDataset.import_all(folder, skiprows=8)

**Purpose:** Recursively imports all `.txt` files under a folder into the dataset.
**Arguments:**

- `folder`: Top-level folder containing measurement files.
- `skiprows`: Number of initial lines to skip in each file.

**Side Effects:** Automatically calls `compute_WB_outputs()`.

## 7.5 ExperimentDataset.import_folder(folder, skiprows=8)

**Purpose:** Convenience constructor: creates a dataset and imports all files.
**Arguments:** same as `import_all`.

## 7.6 ExperimentDataset.compute_WB_outputs()

**Purpose:** Calculates bridge outputs for all WB samples.
**What it does:**

- `WB_out_same`: DMM1 - DMM2 of the same bridge.
- `WB_out_crossed`: DMM1 of a bridge minus DMM2 of its paired bridge (WB2 WB6, WB4 WB8).

## 7.7 ExperimentDataset.group_by(attr)

**Purpose:** Groups samples by metadata attribute.
**Arguments:**

- `attr`: Metadata field to group by (e.g., 'wafer', 'anneal', 'die').

**Returns:** Dictionary {`attr_value: [samples]`}.

## 7.8 ExperimentDataset.filter(**criteria)

**Purpose:** Filters samples by metadata fields.
**Arguments:** Keyword arguments where keys are metadata fields and values are either:

- A value to match (e.g., `measurement='WB2'`).
- A function returning True/False for more complex selection.

## 7.9 ExperimentDataset.print_files_by_wafer()

**Purpose:** Prints all sample file paths organized by wafer.

## 7.10 DataPlotter.plot_pair_cw(pair=('WB1','WB2'), output_type='same')

**Purpose:** Overlay two WB traces (CW only) for one wafer+anneal.
**Arguments:**

- `pair`: Tuple of WB labels to compare.
- `output_type`: 'same' or 'crossed'.

**What it plots:** Angle vs WB output for both bridges.

### 7.11 DataPlotter.plot_overlay_by_position(output_type='same', measurements=None)

**Purpose:** Overlay multiple WB arms for each wafer+anneal+coordinate, split by CW/CCW.

**Arguments:**

- `output_type`: 'same' or 'crossed'.
- `measurements`: List of WB labels to plot (default: WB1–WB8).

### 7.12 DataPlotter.plot_d7_balance()

**Purpose:** Visualize bridge imbalances for D7 die.

**What it plots:**

- Overlay of DMM1 - DMM2 vs angle.
- Distribution (violin + boxplot + raw points) of bridge imbalances.

### 7.13 DataPlotter.plot_ratio_and_angle(pair=('WB2','WB6'), output_type='same')

**Purpose:** Compute ratio of two WB legs, calculated angle, and difference to applied angle.

**Arguments:**

- `pair`: Tuple of two WB labels or integers (e.g., (2,6)).
- `output_type`: 'same' or 'crossed'.

**What it plots:**

- Ratio vs angle.
- Computed angle vs applied angle.
- Difference vs angle.

### 7.14 DataPlotter.plot_d7_mr_curves(fluence_map, mode='split_chiplet', rb_label='RB1')

**Purpose:** Plots R vs angle curves for D7 die across fluences.

**Arguments:**

- `fluence_map`: List of fluence values mapped to x_coord.
- `mode`: Plot mode:
  - 'split_chiplet': one plot per chiplet.

  - 'same_chiplet_die': one plot per die within chiplet.
  - 'merged': all RB arms together.
  - 'compare_rbx': compare a single RB arm across fluences.

- `rb_label`: RB arm to filter if mode='compare_rbx'.

### 7.15 debug_reading(ds, max_samples=5, n_raw=10, n_df=5)

**Purpose:** Inspect raw text files and parsed DataFrames for troubleshooting.
**Arguments:**

- `ds`: Instance of `ExperimentDataset`.
- `max_samples`: Number of samples to check.
- `n_raw`: Number of raw lines to print.
- `n_df`: Number of DataFrame rows to print.

**What it prints:**

- First raw lines of each file.
- First rows of parsed DataFrame.
- Quick check of angle monotonicity.