Realtime BIN File Parser

This program is designed to automatically detect and decode `.bin` files in the specified directory and generate real-time CSV outputs into a designated output directory. The configuration settings for this program are customizable through a `config.json` file.

Features

- **Automatic File Detection**: Automatically detects `.bin` files in the current directory and decodes them in real-time.
- **Manual File Decoding**: If automatic detection is disabled, the program will decode the file provided in the `file_path` configuration.
- **CSV Output**: The decoded data is saved as a CSV file in the specified output directory.
- **Clock Rate and Frame Rate Synchronization**: The `clock` and `desired_frame_rate` settings should match those used in the smartwatch GUI.
- **Logging**: The program logs operations with an adjustable logging level and stores logs in a specified directory.

Configuration

The program behavior is controlled by a `config.json` file with the following fields:

`config.json` Structure

```
{
  "clock": 32768,
  "desired_frame_rate": 250,
  "auto_detect": true,
  "file_path": "",
  "search_pattern": "*.bin",
  "output_directory": "./",
  "logging": "info",
  "logging_directory": "./logging"
}
```

Field Descriptions:

- **clock**: (Integer) The clock rate, which should match the settings used in the smartwatch GUI (e.g., `32768`).
- **desired_frame_rate**: (Integer) The frame rate to be used, which should also align with the settings in the smartwatch GUI (e.g., `250`).
- **auto_detect**: (Boolean)
  - `true`: Automatically detects `.bin` files in the current directory and decodes them in real-time.
  - `false`: Decodes only the file specified in the `file_path`.
- **file_path**: (String) The path of the `.bin` file to decode when `auto_detect` is set to `false`.

- **search_pattern**: (String) The file search pattern. Default is `*.bin`, meaning it will look for all `.bin` files.
- **output_directory**: (String) The directory where the decoded CSV files will be saved. Default is `./`.
- **logging**: (String) The logging level for the program. Options include `info`, `debug`, `error`, etc.
- **logging_directory**: (String) The directory where log files will be stored. Default is `./logging`.

Usage

1. **Set up the smartwatch GUI**:
   - Run the smartwatch GUI software.
   - Disable the **ECG** feature.
   - Disable **PPG2**.
   - Set the **PPG mode** to **raw mode**.
   - Set **logging** to **file**.

   Prepare the program:
   - Place the `realtime_parser_auto` program in the same directory as the log files generated by the smartwatch GUI (specified in the GUI's logging directory).

   Start data collection:
   - In the smartwatch GUI, click **Start** to begin measuring PPG data.
   - Once the GUI starts collecting data, run the `realtime_parser_auto` program.

   Automatic or manual detection:
   - If `auto_detect` is enabled in the `config.json`, the program will automatically scan the directory for `.bin` files and begin decoding.
   - If `auto_detect` is disabled, the program will decode the file specified in the `file_path` field of `config.json`.

   Output:
   - The decoded data will be saved as CSV files in the specified `output_directory`.
   - Logs will be generated in the `logging_directory`.

Example

For a configuration where the clock rate is `32768`, the desired frame rate is `250`, and automatic detection is enabled, the `config.json` would look like this:

```
{
  "clock": 32768,
  "desired_frame_rate": 250,
  "auto_detect": true,
  "file_path": "",
  "search_pattern": "*.bin",
  "output_directory": "./output",
  "logging": "info",
  "logging_directory": "./logs"
}
```

In this case, the program will automatically look for `.bin` files in the current directory, decode them, and output the CSV files to the `./output` directory. Log files will be saved in the `./logs` directory.

Logging

The program uses a flexible logging system, allowing you to monitor its behavior at different verbosity levels. You can adjust the logging level in the `config.json` file by setting the `logging` field. Options include:

- `info`: General information about the program's operation.
- `debug`: More detailed information useful for debugging.
- `error`: Logs only errors encountered during execution.