USB Data Logger Software Documentation

*Introduction*

This documentation will serve as insight into the working of the USB Data Logger Software and the procedures, including Library Installation and python file (.py) to executable file (.exe) conversion, required when implement any changes.

*Library Installation*

The required libraries for running the python code are available in Figure 1. The .exe version stores all the required libraries internally thus installation of these libraries will only be required if implementing changes to the code.

A black background with multicolored text

Description automatically generated

*Figure 1 - Required Python Libraries*

To install these libraries the user to navigate to a newly created virtual environment. Instructions on how to do this are easily accessible online and dependent on the user’s IDE of choice.

In the virtual environment terminal enter the following prompts, with each taking approximately a minute to install.

* “ **Pip install digi-xbee** ”
* “ **Pip install watchdog** ”

*Python file to EXE file Conversion Process*

Syntax Convention:

Angle brackets, <>, will be used to indicate that the user should input their specific parameter.

For Example, <filename>.py should be replaced with the actual filename, such as data\_logger.py.

Steps to convert the python file into an Exe File are as follows

1. In command Line enter prompt “ pip install pyinstaller ”
2. Enter prompt “ CD <directory path to file location> ” to navigate to the folder containing the python file of interest
3. Enter prompt “ pyinstaller --onefile <filename>.py ”, this will be the conversion process and will take a few minutes

After completion, the <filename>.exe can be located by opening the same folder as the <filename>.py file and navigating into the 'dist' folder, followed by the 'build' folder.

*Functionality of USB Data Logger Code*

*Configuring of XBee device*

The section of code displayed in Figure 2 outlines the configuring of the XBee device. This initiates the location of the XBee device on the computer, port, and selected baud rate, and additionally the location of the Hub device on the network, via specifying remote ID.

A computer screen shot of text

Description automatically generated

Figure 2 – Confiuration of XBee Device

*Updating Log File of Interest and Cache File*

The next series of functions are focused on the process of identifying the names of the log files of interest and updating the cache text file. Figure 3 displays the 'log\_files' function, with 'folder\_path' as an input parameter. Using the OS library, the code scans the contents of the log folder and identifies any files with the .log extension. These file names are then stored in a list called 'file\_names.

A screen shot of a computer code

Description automatically generated

Figure 3 – Log File Function

The 'Cached\_Files' function serves the basic purpose of reading the cache text file and outputting individual filenames, as shown in Figure 4. These filenames are stored in a list named 'cached\_file\_names' and are used as input in the next function, 'new\_log\_files.

A screen shot of a computer code

Description automatically generated

Figure 4 – Cache File Function

The 'New\_Log\_Files' function, Figure 5, completes the process by analyzing the output of the previous two functions, which are the log file names ('file\_name') and the cached file names ('Cached\_file\_names'). This function has two main objectives: first, it determines which log file was added, and second, it updates the cache file.

The first step is accomplished by analyzing the log file names and checking if they are also present in the cache text file. If a log file is not found in the cache, it is considered the added log file and is stored in a list as the output of this function, labeled as 'log\_files\_to\_analyse.'

The second step performed by this function involves updating the cache file by removing any cache filenames that are no longer present in the log files.

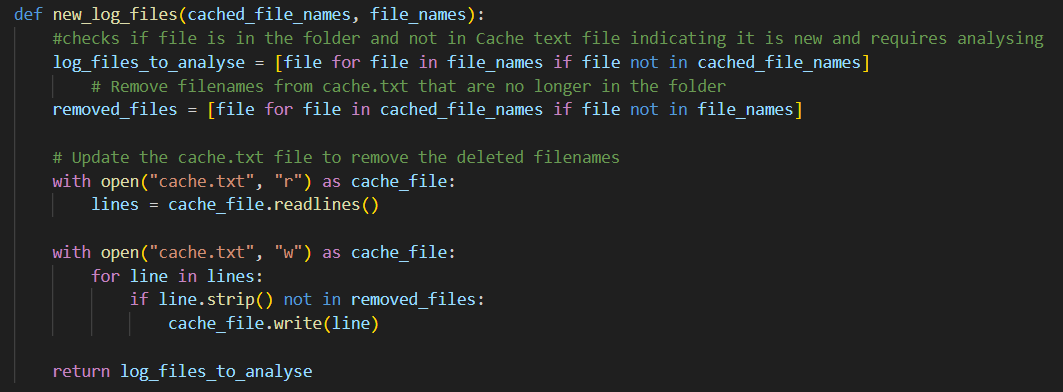


Figure 5 – New Log File Function

*Data and Parameters Extraction*

This next section of functions represents the core purpose of this software and facilities the analysing of the log file along with the transmitting of the required parameters.

The 'extract\_data' function, as shown in Figure 6, is responsible for the data scraping process and outputting the data in an easily accessible format. This function takes as input the list of added log files. It reads the contents of the log files line by line and using tab ('/t') as the delimiter.

In the log header, the point of interest is checking if the line 'Recipe: StandardCleaning.rcp' is present, indicating a cleaning run. The presence of this line is stored as a Boolean variable called 'Cleaning Status.'

The next step in this function is to determine the starting line of the data headers by searching for a known header, 'time (ms),' as the value to locate. With this line identified, the headers and their respective data, located at the same column index, are stored together in the 'data\_columns'.

A screen shot of a computer program

Description automatically generated

Figure 6 – Extract Data Function

With the data of the log file easily accessible the next step is to perform calculations to extract the required parameters. The first such parameter is the time the device is on for which is calculated in the “calculate\_time\_on” function, Figure 7, which takes the data\_columns as the input parameter.

A screen shot of a computer program

Description automatically generated

Figure 7 – Calculate Time on Function

The next parameter of interest is the Average ICP power, which is determined by the 'Calculate\_AVG\_ICP\_Pwr' function, as shown in Figure 8. To calculate this average, the function takes 'data\_columns' as input and extracts the values of data under the 'ICP power (W)' header. These values are then averaged to obtain the average ICP power.

A screen shot of a computer program

Description automatically generated

Figure 8 – Calculates Average ICP Power Function

The primary function responsible for extracting all the parameters and coordinating the calculation-heavy analysis is named 'extract\_parameters.' This function takes 'data\_columns' and 'cleaning status' as inputs. It performs straightforward location finding for parameters, such as the initial and final MSK, using indexing on the 'data\_columns.' Additionally, the 'calculate\_time\_on' and 'Calculate\_AVG\_ICP\_pwr' functions are called within this function to facilitate the completed list of required parameter to output.

A screen shot of a computer

Description automatically generated

Figure 9 – Extract Parameters Function

*Main Function*

The main function serves as the coordinator of the functions mentioned earlier, establishing the order in which they are executed and the inputs they receive. The procedure is straightforward:

1. The process starts with the completion of log file name and cache file processes.
2. Next, the identified log files are sent to the 'extract\_data' function to organize their content into an array named 'data\_columns.'
3. This array is then inputted into the 'extract\_parameter' function, which outputs all the key parameters.
4. These parameters are then converted into a string and combined to form a sentence saved as the variable 'transmit\_data.'
5. Utilizing the configured XBee device, the 'transmit\_data' is transmitted to the Hub via the Xbee library's 'send\_data()' functionality.

If any errors occur during the analysing or transmitting period the error message is captured and transmitted to the Hub to alert the user. Using the try / except method provided robustness to the code as errors will be caught and not result in the code crashing

A computer screen shot of text

Description automatically generated

Figure 10 – Main Function

*Watchdog Functionality*

A crucial requirement for this design is code sleeping to minimize power consumption and CPU usage. This functionality is implemented using the watchdog library, as illustrated in Figure 11. A class for the “LogfileHandler” has been established with the functionality to call the main function 'main(folder\_path)' if a new log file has been detected or to remain idle if not. The logic for detecting new files is located in the section below, where an event handler and observer system have been established. The system is configured to go to sleep for 30 minutes and then check if any new files have been observed during this period.A screen shot of a computer program

Description automatically generated

Figure 11 – Watch Dog Functionality