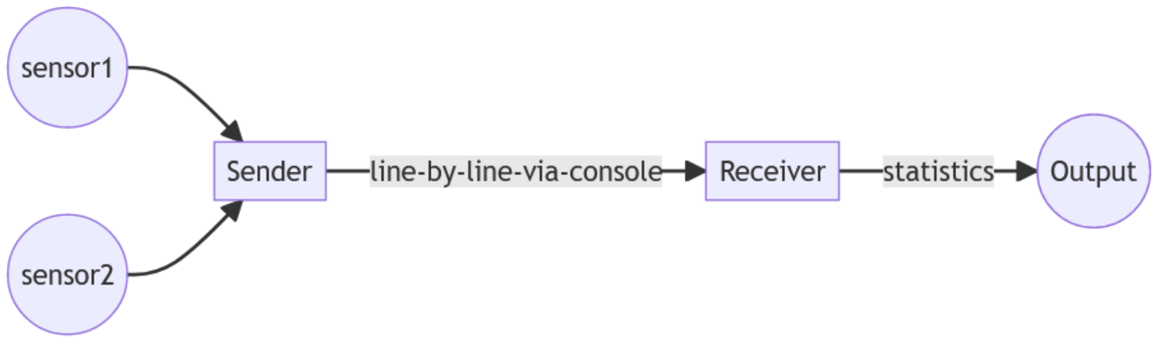
Project – Streaming BMS Data

Project Description:

At a top level, the program runs in two processes - the sender and the receiver.



The Sender simulates and processes data from sensors. It sends data line-by-line to the console, in a format that it defines. The Receiver inputs this data and computes statistics.

The Sender sends data to the Receiver using console redirection and pipes. They need to run on the command-line as follows:

```
sender-executable | receiver-executable
```

Phase 1: Sender

Minimum Functionality of Sender:

- simulates and sends at least two Battery / Charging parameters
- sends fifty readings in a stream
- can either generate values for the parameters, or read from a file
- uses console output to communicate the parameters.

Parameters selected for the project:

1) State of charge SOC:

The BSOC is defined as the fraction of the total energy or battery capacity that has been used over the total available from the battery. State of charge (SOC) gives the ratio of the amount of energy presently stored in the battery to the nominal rated capacity.

For example, for a battery at 80% SOC and with a 500 Ah capacity, the energy stored in the battery is 400 Ah.

Project – Streaming BMS Data

BSOC Values range from 0 to 100 in integers. These integers actually represent the percentage of energy stored.

2) Operating Temperature(T):

Battery operating temperature(T) is the temperature at which battery operates. T ranges from - 20 degrees to 45 degrees.

Output Format:

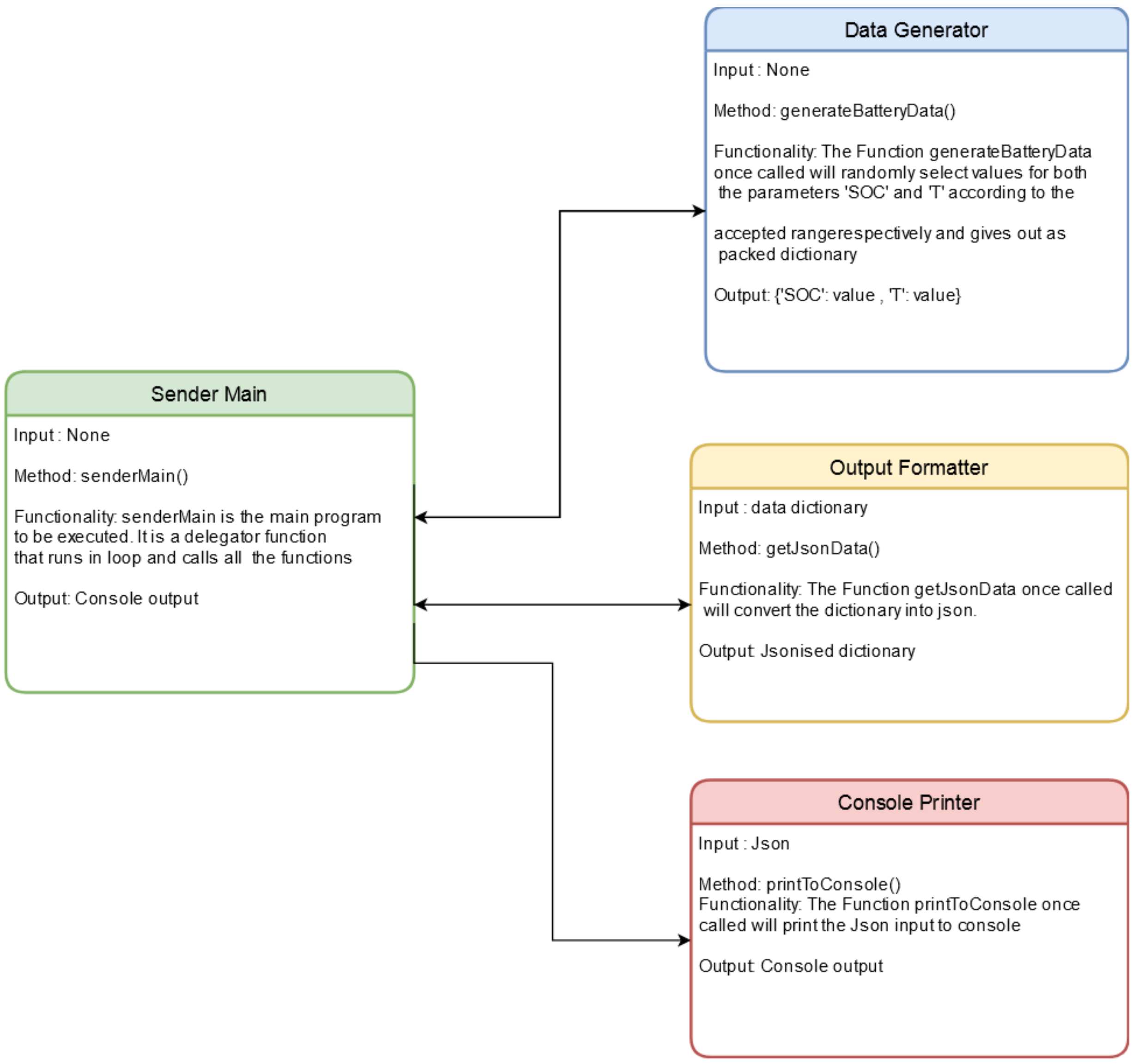
The standard decided for output in console from sender is Json in encoded format. Each reading of the data stream will be a separate json sent (printed) line by line in console.

Sample Sender Output: (In Json)

```
{
"SOC":23,
"T": -2
}
```

Project – Streaming BMS Data

Flow Diagram:


Data Generator
Input : None
Method: generateBatteryData()
Functionality: The Function generateBatteryData once called will randomly select values for both the parameters 'SOC' and 'T' according to the
accepted rangerespectively and gives out as packed dictionary
Output: {'SOC': value , 'T': value}
Sender Main
Input : None
Method: senderMain()
Functionality: senderMain is the main program to be executed. It is a delegator function that runs in loop and calls all the functions
Output: Console output
Output Formatter
Input : data dictionary
Method: getJsonData()
Functionality: The Function getJsonData once called will convert the dictionary into json.
Output: Jsonised dictionary
Console Printer
Input : Json
Method: printToConsole()
Functionality: The Function printToConsole once called will print the Json input to console
Output: Console output

Test Specifications:

1. Given None when the function generateBatteryData() is called then a dictionary output with keys SOC , T with random values is expected.
2. Given dictionary when the function getJsonData() is called then output of type json is expected.

Project – Streaming BMS Data

3. Given None when the function sendData() is called then the count of each variables indicating the number of time the stub code is called should be equal to length of data stream.

Note:

The main code to run is SenderMain.py

Command to run: `python SenderMain.py`