RSSI Application

RSSI application

A complete walkthrough - [version 1.0.0]

Contents:

---

1. Purpose of the application

The purpose of this application is to populate the nearby Bluetooth devices and gather their information such as name, MAC ID, their corresponding RSSI values, and manufacturing data.

It also allows the user to scan for a target device for a specified period of time and plot its RSSI values detected against the time stamps at which the RSSI value is detected in a graph.

The user will be able to download this graph as an image (.png, .jpg and .pdf file formats), as .csv files and as a .xlsx file.

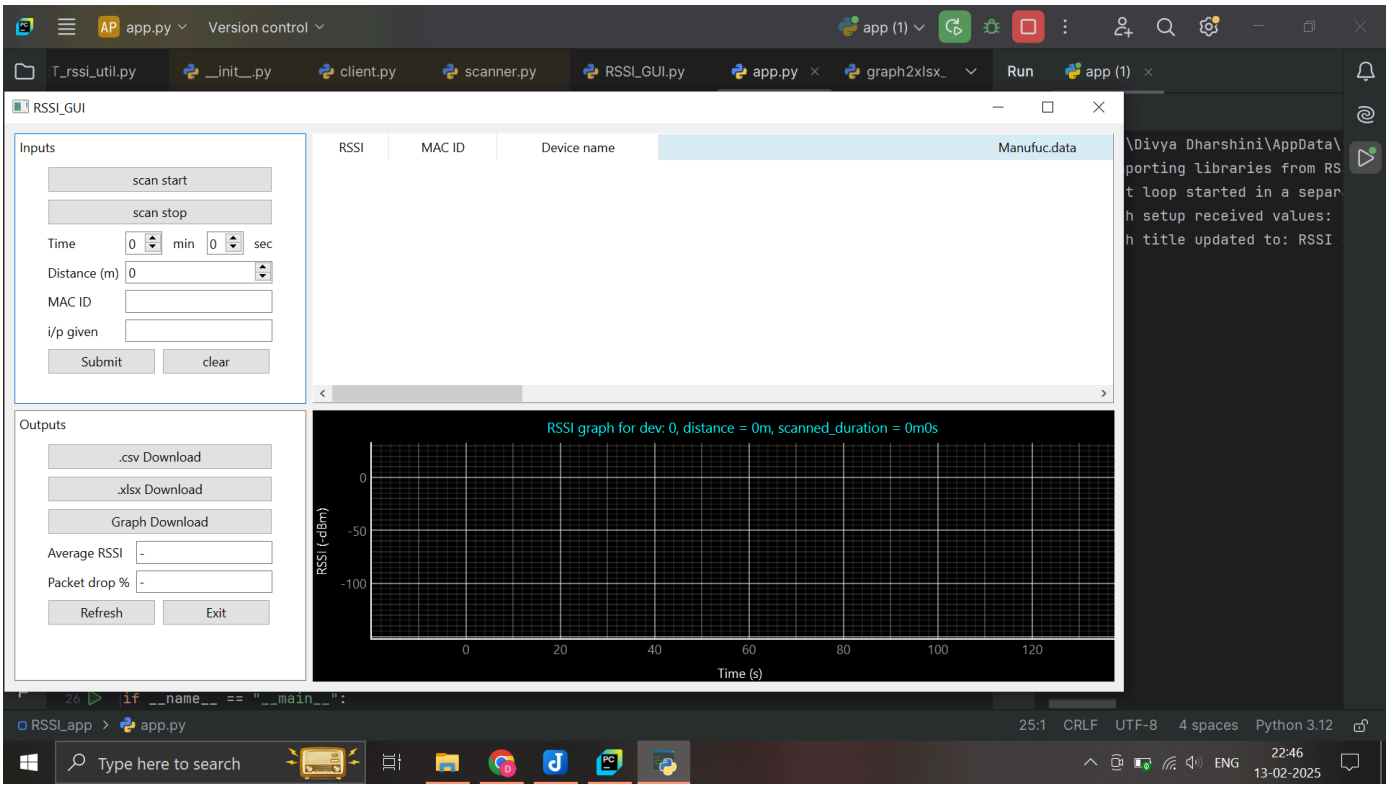
The user can also view the performance parameters of the target device like Average RSSI value and packet drop % once the graph is done plotting for the desired duration of scanning.

---

2. Application layout /UI guide

The application's main window is divided into 4 different grids (or tabs) as follows.

- Inputs
- Outputs
- Table
- Graph



Inputs

1. Scan start and Scan Stop:
   Inputs tab offers "scan start" and "scan stop" options. The former option will take 10 seconds to populate the nearby BLE devices concurrently until the "scan stop" button is pressed.
2. Time:
   The minutes and seconds spin box is where the duration of scanning should be set when scanning for a specific device.
   They both have a maximum value of 60 (60 mins and 60 seconds) increasing in steps of 1 in terms of their face value.
3. Distance:
   Distance is just used as one of the labels for the files that the user will be able to download to mark the experiments details.
4. MAC ID:
   MAC id of the desired device to be scanned for should be entered in this field.
5. I/p given:
   This is just a field to view the inputs (time: minutes, seconds, distance, MAC id) given by the user and serves no purpose other than for its transparency.
6. Submit:
   Once the user enters the inputs, hitting this button will acknowledge that the user wants to start the scanning and graph plotting.
7. Clear:
   Clears the input fields.

Outputs

Once the target device is scanned for the specified duration, the following data will be ready to be downloaded.
All the files has two parameters: RSSI values, Time stamps at which the RSSI is detected.

All the files are saved under the following built-in naming convention:

$$<\text{mac id}><\text{distance}><\text{scanning duration}>.\text{file format}$$

The user can wish to change the name.

1. .csv Download:
   Allows the user to download the csv file.
2. .xlsx Download:
   Allows the user to download the xlsx file.
3. Graph Download:
   Allows the user to download the image is .jpg, .png and .pdf file format.
4. Average RSSI (-dBm):
   It displays the average RSSI value calculated from the data collected over the scanning duration.

$$\text{Average RSSI} = \frac{\sum \text{RSSI values gathered}}{\text{No. of RSSI values gathered}}$$

```
# Average RSSI value
scan_success = len(rssi_list)
if scan_success > 0:
    avg_rssi = sum(rssi_list) / scan_success
```

5. Packet Drop (%):
   It displays the amount of packets dropped in percentage.

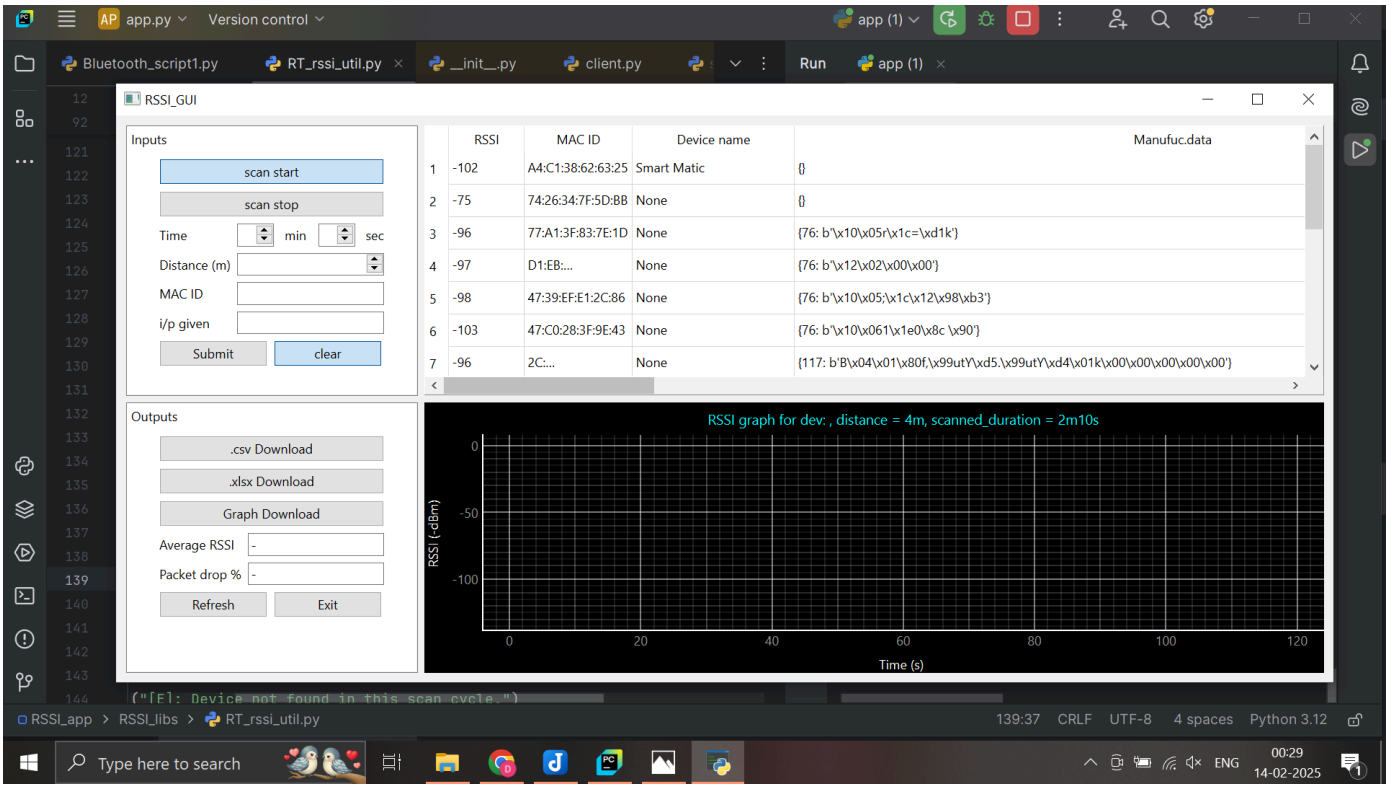
$$\text{Packet drop} = \left[1 - \frac{\text{No. of RSSI values gathered}}{\text{No. of scanning cycles}}\right] * 100$$

```
# Packet drop %
if scan_attempt > 0:
    packet_drop = (1 - (scan_success / scan_attempt)) * 100
```

6. Refresh:
   Refreshes the Application.
7. Exit:
   Exits the Application.

Table

Seen at the top right corner is our table where the populated devices will be listed giving us the details of the BLE devices such as RSSI, MAC id, name and manufacturing data.



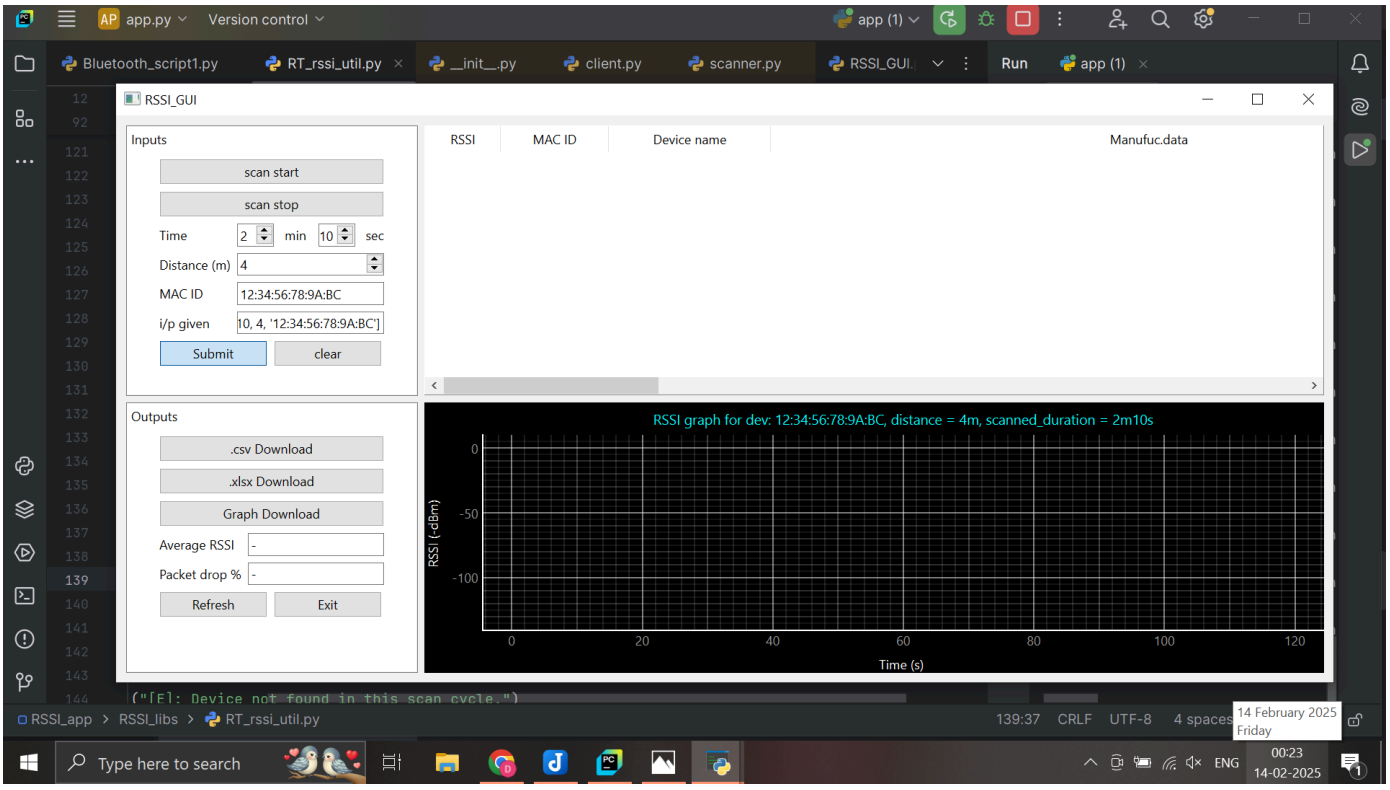
If the device's name is unknown it will be returned as `None` and if the device's Manufacturing data is not available it will be returned as `{}`

This is a consequence of pressing "scan start" button from inputs tab.

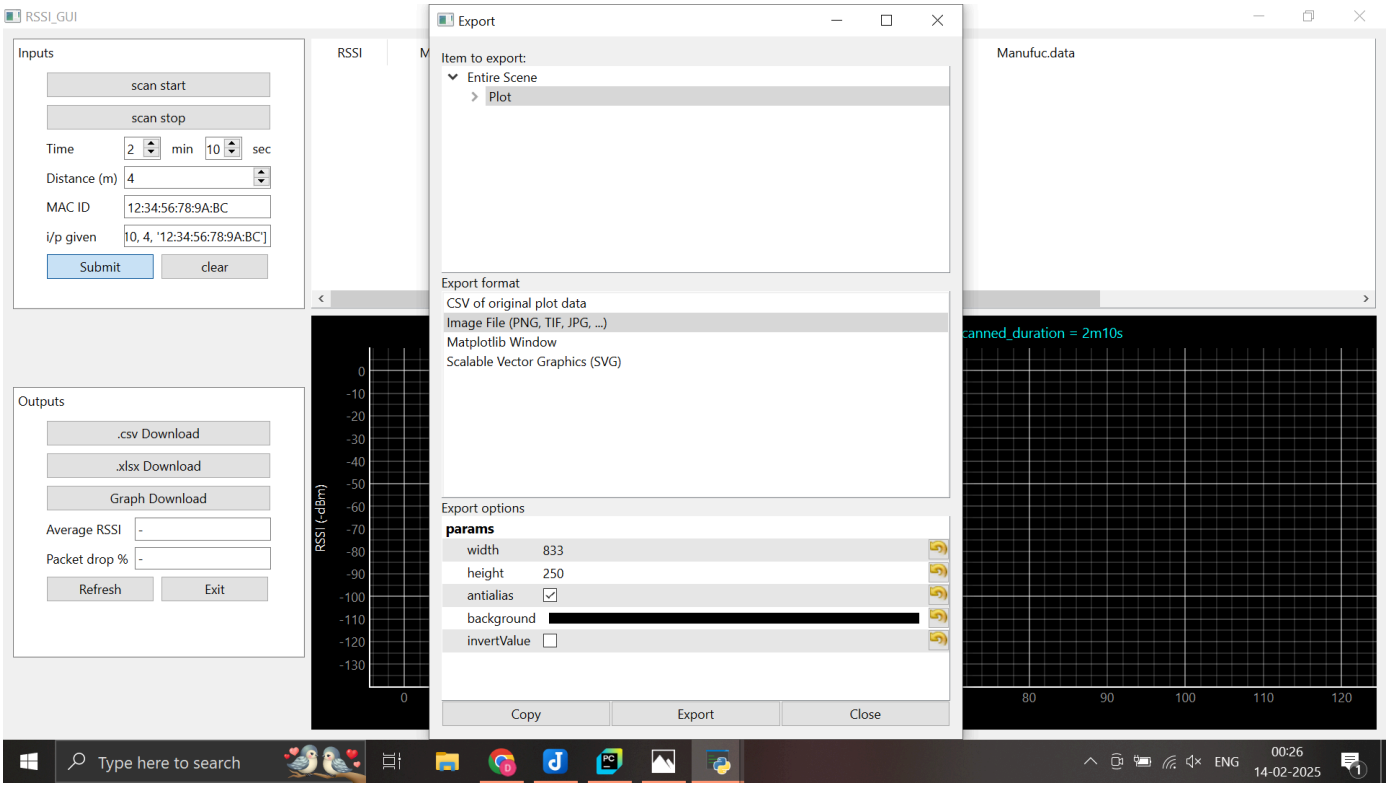
`Graph`

The Graph table is where the target device's RSSI value is plotted against time for the desired scanning duration. The scanning mode is active which supports all the OS. When opting out for Passive mode for battery saving kindly note that this change will throw a "BLE error" in python's console. Check compatibility with respect to Python's Bleak library.

Once the "submit" button is hit, the inputs automatically fill in the graph's title making it easier for identification later.



The graph's visual options are diverse. The user can move the scales, enable or disable the grids, adjust grid's opacity, transform it into power spectrum (FFT), log x, log y, differentials (dy /dx), and y vs y', down sample, clip and much more. Right click on the graph tab to access all the above options.



The graphs can also be downloaded which allows much more customization and control over the data from this in-built "Export" tab.

---

3. Getting started

This section provides step-by-step instructions to install, configure, and run the application.

3.1. Install Python

Ensure Python **3.8 or later** is installed on your system.

- Check Python version:

  ```
  python --version
  ```

  or

  ```
  python3 --version
  ```

- If Python is not installed, download it from:
  https://www.python.org/downloads/
- Ensure `pip` is installed and updated:

  ```
  python -m ensurepip --default-pip
  python -m pip install --upgrade pip
  ```

3.2. Set Up a Virtual Environment

A virtual environment isolates dependencies to prevent conflicts.

```
python -m venv venv
```

- On Windows:

  ```
  venv\Scripts\activate
  ```

- On macOS/Linux:

  ```
  source venv/bin/activate
  ```

3.3. Clone or Download the Project

- Using Git:

  ```
  git clone https://github.com/Divya-dd03/RSSI-application
  <project_directory>
  cd <project_directory>
  ```

- Or manually download and extract the project files.

3.4 Running the Application

Navigate to the project folder and run:

```
python app.py
```

3.5 Configuration (Optional)

If you need to modify settings (e.g., scan timeout, scanning mode, etc), edit the relevant files inside the `RSSI_libs` folder:

- `Bluetooth_script1.py` → BLE scanning logic
- `RT_rssi_util.py` → Real-time RSSI processing
- `graph2xlsx_util.py` → Data export utilities

---

4. Application workflow

< yet to be updated >

---

5. System Requirements

Before installing, ensure your system meets the following requirements:

- [ ] Operating System: Windows 10/11, Linux, or macOS
- [ ] Python Version: 3.8+
- [ ] Bluetooth Adapter: Built-in or external BLE 4.0+
- [ ] Required Python Libraries:
  - [ ] `bleak` (for BLE scanning)
  - [ ] `threading` & `asyncio` (for asynchronous operations)
  - [ ] `PyQt6` (for the GUI)
  - [ ] `pyqtgraph` (for plotting RSSI values)
  - [ ] `openpyxl` (for Excel file handling)
  - [ ] `sys` (for system operations)

Ensure your Bluetooth adapter is enabled and properly configured before running the application. Update system drivers for Bluetooth for the extra step.

---

6. Future Updates

- [ ] Plotting more RSSI values.
- [ ] Continuous scanning.
- [ ] Search bar for device identification in the table.
- [ ] Sorting table in ascending and descending order w.r.t. RSSI values
- [ ] Sorting table according to new data

---

7. Troubleshooting and FAQs

---

8. Version History

`Current: v1.0.0`

Disclaimer

Scan interval and scan window defined in the OS's API will affect the quality of graph, csv and all the output related parameters. Kindly check for system requirements before installation.