|  |  |
| --- | --- |
|  | Proprietary  Information |

Current sensor GRAPHING application

Software Requirement Specification

12th June 2023

Version 0.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Rev. | Date | Description of Changes | Author |
| 0.0 | 28th July, 2023 | Created | Dhruv |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Introduction

This Software Requirement Specification (SRS) document outlines the specifications for a Python application interfacing with an Arduino device. The software is designed for data acquisition, visualization, and storage. The data being measured include current, voltage, and power, obtained from sensors connected to the Arduino.

System overview

The system consists of an Arduino microcontroller and a computer running a Python-based GUI. The Arduino collects data from its environment through its sensors and sends it to the Python application over a serial connection. The Python application processes this data, visualizes it in real-time, and provides the user with controls to manage data collection and visualization.

Operating environment

The software is designed to run on a computer with a modern operating system (Windows, macOS, Linux) with Python 3.x installed. The PyQt library is required for the GUI. The computer should be equipped with a USB port to connect the Arduino device. The Arduino device should have the provided sketch loaded and running.

Functional Requirements (Responsibilities)

1. The software shall establish a serial connection with the Arduino device over a specified COM port and baud rate.
2. The software shall provide the user with an option to refresh the list of available COM ports.
3. The software shall provide controls to start and stop data collection.
4. The software shall display raw data from the Arduino device in a dedicated section of the GUI.
5. The software shall visualize the collected current and voltage data in real-time on two separate graphs.
6. The software shall provide an option to save the raw, current, and voltage data to a file on disk.
7. The software shall provide an option to load previously saved data from a file on disk and display it on the graphs.
8. The software shall provide an option to save the current state of the graphs as an image file.
9. The software shall provide an option for the user to set the sample rate of data collection.
10. The software shall automatically adjust the y-axis scale of the plot based on the data if the autoscale option is checked.
11. The software shall provide an option to synchronize the X- and Y-axes of the two plots.
12. The software shall provide an option to clear the data displayed on the plots.
13. The software shall provide an option to zoom in and out on the X-axis of the plots.
14. The software shall provide an option to save and load configuration settings to/from a file.
15. The software shall provide an option to close the application and ensure all resources, such as the serial port, are properly released.

Users (Interface) requirements

User:

* The user should be proficient in using GUI applications.
* The user should have a basic understanding of Arduino, its sensors, and the data they provide.
* The user should have basic skills to connect the Arduino device to the computer.

Input:

* User interactions with the GUI: connect/disconnect to the Arduino, start/stop data collection, set the sample rate, save/load data, synchronize, zoom in/out on the graphs, and so on.
* User can also input data files for loading previously saved data.

Output:

* The primary output is the real-time visualization of the data collected from the Arduino device.
* The software can also output data files when the user chooses to save the data.

Use cases

* **Use Case 1: Connect to Arduino**

User selects the appropriate COM port and baud rate and clicks on the Connect button.  
The software establishes a connection with the Arduino device and updates the connection status.

* **Use Case 2: Start Data Collection**

User clicks on the Start Data Collection button.  
The software starts collecting data from the Arduino and visualizes it in real-time.

* **Use Case 3: Save Collected Data**

User clicks on the Save Data button.  
The software prompts the user for a filename and location and saves the collected data to the specified file.

* **Use Case 4: Load and Visualize Previously Saved Data**

User clicks on the Load Data button and selects a previously saved data file.  
The software loads the data from the file and visualizes it on the graphs.

Database requirements

The application does not interact with a traditional database. All data is temporarily stored in memory while the application is running and can be saved to or loaded from files on disk.

Testability requirements

The Python-PyQt application should include unit tests for its various functions. The Arduino device can be tested independently to ensure it is correctly collecting and transmitting data.

Logging requirements

All interactions with the Arduino device, such as connecting, disconnecting, and any errors, should be logged to a text box within the GUI. This raw data view provides the user with a detailed history of the Arduino data and any important system events.

Manufacturing requirement

No specific manufacturing requirements are needed for this extension.

Budget

Not Applicable.

Human Resource requirement

The project can be done by anyone with knowledge of python and Arduino programming. Additional resources may be required for testing, documentation, and support.

Hardware Set ups requirements

A computer with Python and the necessary libraries installed is required. An Arduino device with the provided sketch loaded and the necessary sensor connected is also required. The sensor should be capable of measuring current, voltage, and power, and it should be connected according to the manufacturer's specifications.

Non-functional requirements

The application must be developed using Python and PyQt.

The application should have a GUI for user interaction.

The application's GUI should be intuitive and easy to use.

The application should be able to operate with minimal latency to ensure real-time data visualization.

The application should handle errors gracefully, providing the user with clear error messages.

The application should be compatible with a variety of Arduino devices and sensors.

What will be the final deliverable?

Executable application.