Lab Exercise 1 – Single Responsibility Principle (SRP) in C++

In an embedded system scenario, let's consider a simple example to illustrate the Single Responsibility Principle (SRP) in C++. The Single Responsibility Principle states that a class should have only one reason to change. In embedded systems, this could be relevant when dealing with different aspects of device functionality.

Without SRP:

```
#include <iostream>

class TemperatureSensor {
public:
    void operate() const {
        // Operating temperature sensor
        std::cout << "Operating temperature sensor." << std::endl;
    }

    void logData() const {
        // Logging temperature data
        std::cout << "Logging temperature data." << std::endl;
    }
};

int main() {
    TemperatureSensor temperatureSensor;

    temperatureSensor.operate(); // Okay, expected behavior
    temperatureSensor.logData(); // Okay, expected behavior
    // Adding a new responsibility (e.g., error handling) requires modifying existing code.
    // temperatureSensor.handleError(); // Not following SRP

    return 0;
}
```

With SRP:

```
#include <iostream>

class TemperatureSensor {
public:
    void operate() const {
        // Operating temperature sensor
        std::cout << "Operating temperature sensor." << std::endl;
    }
};

class Logger {
public:
    void logData() const {
        // Logging data
        std::cout << "Logging data." << std::endl;
    }
};

int main() {
    TemperatureSensor temperatureSensor;
    Logger logger;

    temperatureSensor.operate(); // Okay, expected behavior
    logger.logData();            // Okay, expected behavior

    // Adding a new responsibility (e.g., error handling) doesn't require modifying existing code.
    // ErrorHandler errorHandler;
    // errorHandler.handle(); // Following SRP

    return 0;
}
```

Observations:

- In the version without SRP, the TemperatureSensor class handles both operating the sensor and logging data, violating the Single Responsibility Principle.
- In the version with SRP, the responsibilities of operating the sensor and logging data are separated into two different classes (TemperatureSensor and Logger). This makes the code more modular and adheres to the Single Responsibility Principle, allowing for easier maintenance and extension in an embedded system.