# Gateway Device Application (Connected Devices)

## Lab Module 02

Be sure to implement all the PIOT-GDA-\* issues (requirements) listed at [PIOT-INF-02-001 - Lab Module 02](https://github.com/orgs/programming-the-iot/projects/1#column-9974938).

### Description

NOTE: Include two full paragraphs describing your implementation approach by answering the questions listed below.

What does your implementation do?

**GatewayDeviceApp Class**: This class acts as the primary entry point for the application, managing the overall lifecycle, including the start and stop operations of the SystemPerformanceManager.

**SystemPerformanceManager Module**: Responsible for tracking system performance metrics such as CPU and memory usage, this module collects data at predefined intervals and makes it available to the rest of the application.

**BaseSystemUtilTask Class**: A base class for utility tasks that gather system metrics, providing a framework for specific tasks related to CPU and memory usage monitoring.

**SystemCpuUtilTask Module**: This module handles the retrieval of CPU utilization metrics by implementing a method specifically designed to gather CPU data.

**SystemMemUtilTask Module**: This module is responsible for tracking JVM memory usage, offering functionality to effectively monitor memory consumption.

**Scheduled Telemetry Handling**: The SystemPerformanceManager includes a method called handleTelemetry(), which leverages scheduled threads to periodically call telemetry retrieval methods from SystemCpuUtilTask and SystemMemUtilTask, enabling the regular collection of system performance data.

How does your implementation work?

**Application Initialization**:

* + The GatewayDeviceApp class is instantiated, which in turn initializes the SystemPerformanceManager.
  + The application starts the SystemPerformanceManager, beginning the process of monitoring system performance.

**Performance Monitoring**:

* + The SystemPerformanceManager schedules a task (using a scheduled thread) that regularly invokes the handleTelemetry() method.
  + Inside handleTelemetry(), the getTelemetryValue() methods from both SystemCpuUtilTask and SystemMemUtilTask are called to retrieve the current CPU and memory usage metrics.

**Task Implementation**:

* + The BaseSystemUtilTask class provides a common foundation for SystemCpuUtilTask and SystemMemUtilTask, ensuring they share common properties or methods (if applicable).
  + Each specific task fetches its respective metrics (CPU or memory) using system calls or Java APIs, then returns the results to the SystemPerformanceManager.

**Data Handling**:

* + The performance data collected by the tasks can be logged, displayed, or further processed based on the application's requirements.

**Graceful Shutdown**:

* + When the application is stopped, the GatewayDeviceApp ensures that the SystemPerformanceManager is properly shut down, stopping any active monitoring tasks.

This implementation approach promotes modularity, simplifying the management and extension of the application while offering real-time insights into system performance metrics.

### Code Repository and Branch

NOTE: Be sure to include the branch (e.g. https://github.com/programming-the-iot/python-components/tree/alpha001).

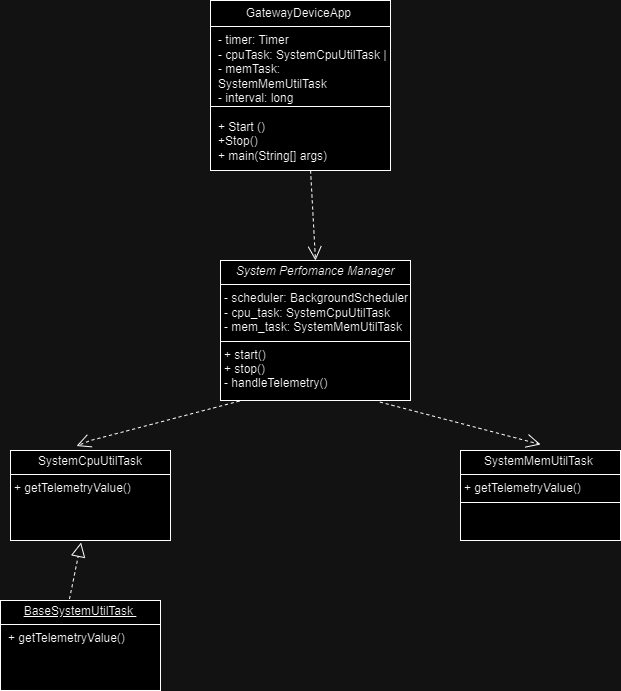
URL: [Elikem161/java-components (github.com)](https://github.com/Elikem161/java-components)

### UML Design Diagram(s)

NOTE: Include one or more UML designs representing your solution. It's expected each

diagram you provide will look similar to, but not the same as, its counterpart in the

book [Programming the IoT](https://learning.oreilly.com/library/view/programming-the-internet/9781492081401/).



### Unit Tests Executed

NOTE: TA's will execute your unit tests. You only need to list each test case below

(e.g. ConfigUtilTest, DataUtilTest, etc). Be sure to include all previous tests, too,

since you need to ensure you haven't introduced regressions.

- ConfigUtilTest  
- SystemCpuUtilTaskTest  
- SystemMemUtilTaskTest

### Integration Tests Executed

NOTE: TA's will execute most of your integration tests using their own environment, with

some exceptions (such as your cloud connectivity tests). In such cases, they'll review

your code to ensure it's correct. As for the tests you execute, you only need to list each

test case below (e.g. SensorSimAdapterManagerTest, DeviceDataManagerTest, etc.)

- GatewayDeviceAppTest  
- SystemPerformanceManagerTest

-