# 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 main entry point for the application. It manages the application’s lifecycle, including starting and stopping the **SystemPerformanceManager**.

 **SystemPerformanceManager Module**: This module is responsible for monitoring system performance metrics, such as CPU and memory usage. It collects this data at scheduled intervals and makes it available to the application.

 **BaseSystemUtilTask Class**: This class serves as a foundational structure for utility tasks that retrieve system metrics. It sets up a framework for specific tasks related to CPU and memory utilization.

 **SystemCpuUtilTask Module**: This module specifically focuses on retrieving the system's CPU utilization. It includes a method for accessing CPU metrics.

 **SystemMemUtilTask Module**: This module is dedicated to monitoring the memory utilization of the Java Virtual Machine (JVM). It provides functionality for effectively tracking memory usage.

 **Scheduled Telemetry Handling**: Within the **SystemPerformanceManager**, there is a method called **handleTelemetry()**. This method uses scheduled threads to invoke the telemetry methods from both **SystemCpuUtilTask** and **SystemMemUtilTask**, enabling periodic collection of performance data.

How does your implementation work?

**1. Application Initialization**

* The **GatewayDeviceApp** class is created, which in turn initializes the **SystemPerformanceManager**.
* The application then starts the **SystemPerformanceManager**, which begins monitoring system performance.

**2. Performance Monitoring**

* The **SystemPerformanceManager** schedules a task that periodically calls the **handleTelemetry()** method using a scheduled thread.
* Within **handleTelemetry()**, the application retrieves the current CPU and memory usage metrics by calling the **getTelemetryValue()** methods from both the **SystemCpuUtilTask** and **SystemMemUtilTask**.

**3. Task Implementation**

* The **BaseSystemUtilTask** serves as a foundation for both **SystemCpuUtilTask** and **SystemMemUtilTask**, ensuring they share any common properties or methods.
* Each specific task gathers its respective data (either CPU or memory) using system calls or Java APIs and returns these values to the **SystemPerformanceManager**.

**4. Data Handling**

* The data collected from the performance tasks can be logged, displayed, or processed further based on the needs of the application.

**5. Graceful Shutdown**

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

This structure promotes modularity, making it easier to manage and expand the application as necessary while providing 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: https://github.com/Francistapiwa/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/>).

A screenshot of a computer

Description automatically generated

### 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

EOF.