Arrowhead Core Systems Integration and Context

The AI-driven control and optimization system follows the architectural principles of the **Arrowhead Framework**, which provides a standardized method for creating **secure, interoperable, and distributed industrial systems**. This section explains the design abstraction levels (SysD, SysDD, SD, IDD) and how the Arrowhead Core Systems ensure reliable communication, orchestration, and security between all subsystems in the local cloud.

1. System Design Documentation Levels

The Arrowhead design structure consists of four main documentation levels, each defining a specific part of the system and its interaction with services.

- **SysD – System Design:** The System Design specifies how a subsystem behaves, what operations it performs, and which services it provides or consumes. It defines the logic and functional architecture of the system, including interactions with other systems through Arrowhead services. *Example:* The **PowerSensor_SysD** defines how the power sensor measures, calibrates, and performs diagnostics using operations such as `MeasurePower()`, `CalibrateSensor()`, and `DiagnosticSelfTest()`.

- **SysDD – System Design Description:** The System Design Description focuses on the physical implementation and configuration of a subsystem. It describes the hardware model, accuracy, size, environmental limits, and any database connections that the SysD uses. *Example:* The **TP-2 Compact Motor Power Sensor SysDD** defines physical parameters like measurement accuracy, temperature range, and connection to a MySQL database.

- **SD – Service Description:** The Service Description specifies which services are made available by a subsystem and which operations can be requested by other systems. It defines the external service interface and how the system communicates over the network. *Example:* The **PowerMeasurement_SD** service provides the operation `GetPowerMeasurement()` which returns the latest power readings.

- **IDD – Interface Design Description:** The Interface Design Description defines the structure of the data exchanged through a service. It provides a consistent data model, ensuring that both producers and consumers understand the same format. *Example:* The **PowerMeasurement_IDD** defines attributes such as `timestamp`, `voltage`, `current`, `power`, `unit`, and `sensorId`, ensuring consistent interpretation of power data across all subsystems.

These documents together ensure that the system is modular, traceable, and interoperable within the Arrowhead-compliant service-oriented architecture.

2. Arrowhead Core Systems

The **Arrowhead Core Systems** are essential infrastructure components that enable all subsystems (sensors, motors, controllers, and optimizers) to securely interact in a distributed industrial environment. Each core system has a specific role in managing communication, orchestration, and security within the **local cloud** and between external clouds.

2.1 Service Registry System (Blue)

The **ServiceRegistry_Design** system maintains an index of all active services in the local cloud. Each subsystem (for example, **PowerSensor_SysD**) registers its offered service (**PowerMeasurement_SD**) with the registry when it becomes available. Consumers, such as the Motor or Feeder systems, query this registry to find available providers. The registry ensures that only registered and trusted services are accessible for orchestration and consumption.

Interfaces:

- `ServiceManagement` – handles registration and removal of services.
- `ServiceDiscovery` – allows consumers to locate registered services.
- `SysMonitor` – monitors the operational state of registered systems.

Function: Acts as a central directory or phonebook of all available services in the Arrowhead local cloud.

2.2 Service Orchestration System (Green)

The **ServiceOrchestration_Design** system coordinates which service provider should be used when multiple options exist. When a consumer (for instance, the AI optimization system) requests data such as **PowerMeasurement_SD**, the orchestrator retrieves service information from the registry, checks authorization status, and returns the connection details of the best provider. This enables dynamic and optimized service selection during runtime.

Interfaces:

- `ServiceOrchestration` – manages orchestration rules and priorities.
- `OrchestrationStoreManagement` – stores and updates orchestration data.
- `SysMonitor` – supervises system performance and status.

Function: Acts as an intelligent matchmaker that connects consumers to the most suitable and authorized service providers.

2.3 Authentication and Authorization Systems (Red)

The **Authentication_Design** and **ConsumptionAuthorization_Design** systems manage the trust and security model within the Arrowhead ecosystem.

- **Authentication_Design:** Validates the identity of service consumers using X.509 certificates to ensure that only known and trusted systems participate in communication.
- **ConsumptionAuthorization_Design:** Manages access rights and issues **Arrowhead Tokens (JWT)** to systems that have permission to consume certain services. It validates tokens and ensures secure, authorized access.

Interfaces:

- `TokenGeneration` – creates secure authorization tokens for clients.
- `AuthorizationManagement` – manages policies and access control lists.
- `GetPublicKey` – distributes public keys for token validation.

Function: These systems together ensure that all data exchange is authenticated, authorized, and traceable — enforcing security and trust across the entire local cloud.

2.4 Gatekeeper and GateTunnel Systems (Yellow)

The **Gatekeeper_Design** and **GateTunnel_Design** systems handle secure communication between different Arrowhead local clouds. While the Gatekeeper controls access permissions and validates external requests, the GateTunnel establishes the physical and encrypted connection between the clouds.

Interfaces:

- `GlobalServiceDiscovery` – allows external clouds to find services.
- `get_connect_consumer` – initiates secure consumer-side connection setup.
- `get_connect_provider` – connects external providers to local consumers.
- `close_session` – securely terminates communication sessions.

Function: Provides secure, certificate-based, encrypted data transfer between local clouds, enabling scalability beyond a single industrial site.

3. Example of System Communication Flow

1. The **PowerSensor_SysD** registers its service (**PowerMeasurement_SD**) in the **ServiceRegistry**.
2. The **MotorControl_SysD** or **AI Optimization_SysD** requests access to this service through the **ServiceOrchestrator**.

3. The Orchestrator identifies the appropriate provider and creates an orchestration rule.

4. The **Authorization System** issues an **Arrowhead Token** for secure access.

5. The **Authentication System** verifies the identity of the requesting system.

6. If communication occurs across local clouds, the **Gatekeeper** and **Gate-Tunnel** systems handle the secure inter-cloud connection.

7. The consumer system receives the structured data as defined by the **PowerMeasurement_IDD**.

Summary: The Arrowhead Core Systems together establish the foundation for secure, modular, and interoperable communication between distributed systems. They guarantee that each subsystem, such as the PowerSensor, Motor, or Feeder, can reliably discover, authorize, and exchange information, forming the backbone of the AI-driven control and optimization architecture.