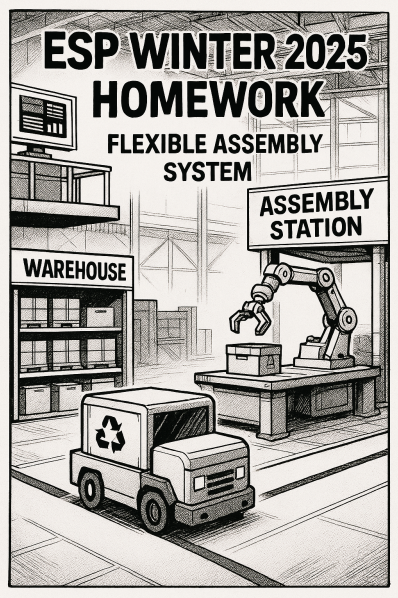
Embedded Systems Programming

Winter Homework

2025-12-05



1 Introduction

Modern industrial plants increasingly rely on flexible assembly systems (FAS) to cope with fluctuating production volumes, product variability, and evolving market needs. Material-handling technologies such as Automated Guided Vehicles (AGVs) and Autonomous Mobile Robots (AMRs) play a central role in these environments, enabling dynamic routing of components and adaptive scheduling. In this assignment, you will design and simulate a small assembly cell. The goal is to model the behaviour of a simplified FAS with the interaction between order management, component handling, warehouse logistics, AGV motion, everything with the concurrent execution of multiple threads that collectively drive the simulation.

2 Industrial Context

The scenario focuses on a flexible assembly cell operating within an industrial plant. Production orders arrive over time and specify which product must be assembled. Each product type has an associated BOM and assembly operations requiring a certain time. The assembly station processes one order at a time, while the warehouse provides all required components and interacts with an AGV fleet that transports materials on demand.

The plant's behaviour emerges from the coordination among these subsystems. Orders must be scheduled, components must be retrieved from storage, AGVs must navigate between locations, and the assembly station must record its activity, finally the control center must compute performance indicators for the entire system. Your simulation must orchestrate these activities under realistic time progression and concurrency constraints.

3 System Description

3.1 Control Center

The Control Center is responsible for releasing production orders into the system according to the timestamps provided in `orders.txt`. At minimum, a FIFO (First-In First-Out) scheduling policy must be supported, though you may later implement alternative strategies such as priority-based dispatching, Shortest Processing Time (SPT), or Earliest Due Date (EDD).

Throughout the simulation, the Control Center supervises execution and, at the end of the execution, computes the Key Performance Indicators (KPIs) before saving them to a summary report. The end of simulation and the reference time horizon are left for you to define and justify.

3.2 Assembly Station

The assembly station processes one order at a time. Before starting any operation, it must request all necessary components from the warehouse based on the product's BOM. The collection of materials incurs a setup time (T_{setup}), which must be incorporated into the operation time, together with the T_{base} time required for the assembly:

$$T_{\text{op}} = T_{\text{base}} + T_{\text{setup}}$$

Assembly begins only once every component is available at the station. Once assembly is complete, the finished product must be transported back to the warehouse by an AGV and recorded in the inventory. The station must record the end time of each order, contributing to the system logs and to later KPI computation.

3.3 AGV Fleet

The logistics system is served by a configurable number of AGVs, each acting as an independent thread. The number of AGVs must be at least 10 ($N_{AGV} \geq 10$), though it is recommended to start with 2 AGVs during the initial design and debugging phase, then scale up to the required fleet size. Every AGV can transport a **single unit** of a given component at a time. When the assembly station requests components, one or more AGVs must collect the required items from the warehouse and deliver them to the station.

The AGV operates according to a discrete set of states:

$$state \in \{IDLE, TO\_WAREHOUSE, PICKING, TO\_STATION, DROPPING\}$$

You must model the AGV's movements along simulated paths, including the travel time between the parking location, the warehouse, and the assembly station. Picking and dropping activities must also be represented with meaningful delays. The method by which this is logged and computed is intentionally left open for your design. You can then extend your work with the return of the AGV back to the warehouse to store the final product.

3.4 Warehouse

The warehouse stores all component types and finished products, serving multiple concurrent requests. Since AGVs may interact with the warehouse simultaneously, careful management of

shared data is required. The initial inventory of components is provided in `warehouse.txt`. When a finished product is delivered by an AGV, it must be added to the warehouse inventory. Your simulation must ensure safe and correct updates to both component and finished product quantities.

4 Scheduling and Performance Indicators

At the end of the simulation, the Control Center must produce a report containing the main Key Performance Indicators:

- **Average Lead Time:**
$$LT = t_{\text{completion}} - t_{\text{release}}$$
- **Assembly Station Utilization:**
$$U = \frac{T_{\text{busy}}}{T_{\text{sim}}}$$
- **Throughput:**
$$TH = \frac{\text{orders}}{\text{hour}}$$
- **AGV Utilization (average):**
$$U_{\text{AGV}} = \frac{\sum_{i=1}^{N} T_{\text{busy},i}}{N_{\text{AGV}} \cdot T_{\text{sim}}}$$

These values must be saved into a file named `kpi_report.txt`.

5 Input Files

Three input files define the structure of the simulation.

5.0.1 orders.txt

Each line contains the release time, product identifier, and (optionally) priority, or due dates. Format:

```
HH MM product_id priority
```

Example:

```
08 10   P1   1
08 15   P2   3
08 45   P1   2
```

5.0.2 bom.txt

Associates each product with its component requirements:

```
product_id assembly_base_time_in_minutes
product_id component_id quantity
product_id component_id quantity
...
```

Example:

```
P1   30
P1   C1   12
P1   C3   4
P2   40
P2   C2   9
P2   C1   7
```

5.0.3 warehouse.txt

Initial warehouse inventory:

```
component_id initial_quantity
```

Example:

```
C1   50
C2   30
C3   20
```

6 Output Requirements

The system must generate a simulation log (`sim_log.txt`) that records significant events. A typical entry should include a timestamp and a short description of the event, for example:

```
08:15  AGV1 arrived at warehouse to pick component C3
```

The Control Center must also produce the KPI report described earlier. Together, these files should document the behaviour and performance of the simulated plant.

7 Assignment Phases

7.1 Phase 1 — Object-Oriented Modelling

Begin by identifying the key abstractions: AGV, Assembly Station, Control Center, Warehouse, Order, Product, Bill of Materials, and any data structures needed for inter-thread communication. Define clear responsibilities and interactions among classes.

7.2 Phase 2 — Multithreading and Concurrency

Determine which elements of the system must run as separate threads. AGVs and the assembly station naturally operate independently; the Control Center may also run as its own thread. Consider how you will manage logging, how threads will coordinate their actions, and which condition variables and critical sections are required for safe and correct behaviour.

7.3 Phase 3 — Scheduling and KPIs

Develop a notion of simulated time, design the logic for order release and assembly sequencing, and compute the KPIs once all work is completed. The simulation must end cleanly, after which the final report is generated.

7.4 Phase 4 — Implementation

Only once the architecture is fully understood you should begin writing code. The assignment rewards thoughtful design more than ad-hoc implementation.

8 Optional ROS extension

An optional ROS extension is available but will be released only upon request by the student and after the completion of the proposed concurrent part.

9 Assessment Criteria

Criterion	Weight
Object-oriented design	40%
Concurrency and synchronization	25%
Simulation correctness and behaviour	20%
KPI computation and reporting	5%
Documentation and clarity	10%
Optional ROS extension	+5% bonus