Group 7

1CV50

Álvaro Arrieta Puente – s1811347
Vasileios Karvouniaris - s1531042
Milly Landmeter – s1893971
Ioanna Panagiotopoulou – s1785729
Kars Polderman – s1887696
Paulo Vieira - s1798618

6-24-2024

Table of Contents

Introduction

In the current landscape of global manufacturing, companies are facing unprecedented challenges driven by globalization and evolving customer demands. These demands include dynamic shifts in market requirements, shorter product life cycles, a higher variety of products, and increased levels of customization. To stay competitive and meet these challenges, manufacturers are compelled to rapidly and radically transform their production processes. This transformation involves moving away from traditional manufacturing systems towards the adoption of next-generation manufacturing technologies, such as cyber-physical systems and automated manufacturing systems. This assignment explores the implementation and evaluation of an advanced automated manufacturing system designed for high-precision machining of diverse materials and component sizes. The system integrates several key elements: a work centre equipped with parallel machines and an automated robot, a manufacturing preparation centre (MPC) with multiple workstations, and a fleet of automated guided vehicles (AGVs) to facilitate the seamless movement of materials.

The structure of this document is as follows: Section 1 provides a detailed description of the automated manufacturing system, including the roles of the MPC, the work centre, and the AGVs. Section 2 outlines the objectives of the assignment, focusing on developing and validating simulation models using Python to analyse system performance. Subsequent sections delve into specific exercises designed to model various components of the system, such as the operators, AGVs, and internal buffers, using SimPy. The final section integrates these models to evaluate the overall system performance, providing insights into optimal configurations and operational efficiencies. By developing and analysing these simulation models, we aim to gain a deeper understanding of the automated manufacturing system's throughput and flow time, enabling manufacturers to make informed decisions about design and operational strategies. This approach not only addresses current manufacturing challenges but also paves the way for future advancements in automated and cyber-physical manufacturing technologies.

Section 6.1. Manufacturing Preparation Centre (MPC)

Executed by 1531042 & 1785729

6.1.1. Operators

To start with, we need to model an appropriate source for the arrival of items and the operators constructing new fixtures. Regarding the source, the inter arrival time of items follow an exponential distribution with a mean arrival time, m, of 8 minutes. For the operators, there are three operators modelled as three standard servers with infinite queue capacity. The building time, B_{time}, is uniformly distributed between B_{min}=5 minutes and B_{max}=15 minutes. The time horizon for the model is 50 days with a warm-up period of 10 days.

To determine the throughput and flow time analytically, we can follow these steps:

Step 1: Calculate the arrival rate (λ) for the source.

The inter-arrival time for items follows an exponential distribution with mean m = 8 minutes. The formula for the arrival rate (λ) for an exponential distribution is:

$\lambda = \frac{1}{m} = \frac{1}{8} = 0.125$ items per minute

Step 2: Calculate the service rate (μ) for the operators.

The average service time for a uniform distribution is the average of the minimum and maximum values, so $\mu = \frac{1}{t_e}$, where $t_e = \frac{Bmin+Bmax}{2} = \frac{5+15}{2} = 10$ minutes. Thus, mean $\mu = \frac{1}{10} = 0.1$ items per minutes and standard deviation $\sigma = \frac{Bmax-Bmin}{\sqrt{12}} = \frac{15-5}{\sqrt{12}} = 2.886$

Step 3: Use Little's Law to calculate the throughput (T) and flow time (W).

Little's Law states that for a stable system, the average number of items in the system (L) is equal to the product of the average throughput (T) and the average flow time (W), so we have: $L = T \cdot W$, where the throughput is equal to the arrival rate when the system is in a steady state, so $T = \lambda = 0.125$, so by Little's Law we end up with:$W = \frac{L}{\lambda}$. But we need to find the average number of items in the system (L). For an M/M/3 queue, the average number of items in the system is given by: $L = \lambda \cdot Wq$, where Wq is the average time, a customer spends waiting in the queue.

The average flow time can be calculated using the following formula:

$$W_q = average.build.time(t_e) + average.time.in.queue(ATQ)$$

So, the remaining value that we need to calculate now is ATQ, in order to be able to find the value of the flowtime.

$$ATQ = \frac{t_e}{M} \cdot \frac{\rho^{\sqrt{(2(M+1))}+1}}{1-\rho} \cdot \frac{CV \cdot \alpha^2 + CV \cdot \rho^2}{2}$$

, where utilization is calculated by $\rho = \frac{t_e}{8} \cdot M = \frac{10}{24}$

$CV \cdot \alpha^2 = \left(\frac{std(interarrival)}{avg(interarrival)}\right)^2 = \left(\frac{10}{10}\right)^2 = 1$ and

$$CV \cdot \rho^2 = \left(\frac{std(t_e)}{t_e}\right)^2 = \left(\frac{2.886}{10}\right)^2 = 0.0833$$

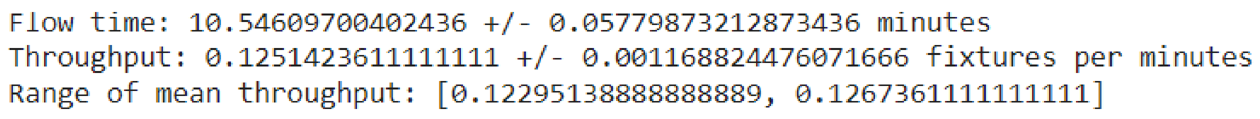
, so now we can combine everything, which results in:

$$ATQ = \frac{10}{3} \cdot \frac{\left(\frac{10}{24}\right)^{\sqrt{8}-1}}{\frac{14}{24}} \cdot \frac{1 + 0.08333}{2} = 0.6245$$

So, by substituting back in the first formula for the flow time, we end up with:

$$W_q = t_e + ATQ = 10 + 0.6245 = 10.6245$$

However, in case one wants to find the solution in other ways, it is also possible to reach to the same output by building the model in python. Below it appears the output of the simPy code corresponding to this section.



```
Flow time: 10.54609700402436 +/- 0.05779873212873436 minutes
Throughput: 0.1251423611111111 +/- 0.001168824476071666 fixtures per minutes
Range of mean throughput: [0.12295138888888889, 0.1267361111111111]
```

Figure 1: SimPy output for 6.1.1

To conclude, the output of the SimPy code corresponds correctly to what was computed in the first part of the exercise where we used the hand-in procedure to calculate the flow time of the process. So, the total time a product/item spends in the production process from start to finish is approximately 10.5 minutes.

6.1.2. Operator bio breaks

This time, the model will include a bio break of the operator while building a new fixture. It is possible for the operator to interrupt the current work and go to bio break before continuing working on it. The operator's break lasts 3 minutes and can be taken every 20 minutes. We assume the operator is ready at time 0. We'll represent the time spent by each item in the source and in the operator workstation. Given the deterministic arrival times and the operator's bio break schedule, the duration each item spends in each station will be calculated.

1. Item 1:
    - Item 1 arrives at 1.0 minute and processing starts immediately
    - Processing time = 10 minutes, completes at 11.0 minutes
2. Item 2:
    - Item 2 arrives at 5.0 minutes, while Item 1 is being processed

- Item 2 starts processing at 11.0 minutes, bio break at 20.0 minutes
- Completes at 24.0 minutes

3. Item 3:
    - Item 3 arrives at 10.0 minutes, while Item 2 is being processed
    - Item 3 starts processing at 24.0 minutes, no bio break interruption
    - Completes at 34.0 minutes
4. Item 4:
    - Arrives at 30.0 minutes, starts processing after Item 3
    - Starts processing at 34.0 minutes, bio break at 40.0 minutes
    - Completes at 47.0 minutes
5. Item 5:
    - Arrives at 45.0 min, starts processing after Item 4
    - Starts processing at 47.0 min
    - Processing interrupted at 57.0 minutes (next bio break)
    - Completes at 57.0 minutes

Item	Arrival time	Start processing	End processing	Bio breaks	Total time in systems
1	1.0 min	1.0 min	11.0 min	0	10.0 min
2	5.0 min	11.0 min	24.0 min	20.0 - 23.0	19.0 min
3	10.0 min	24.0 min	34.0 min	0	24.0 min
4	30.0 min	34.0 min	47.0 min	40.0 - 43.0	17.0 min
5	45.0 min	47.0 min	57.0 min	0	12.0 min

Table 1: Duration of item in each station with bio breaks

So, the respective lot-time diagram will look like:

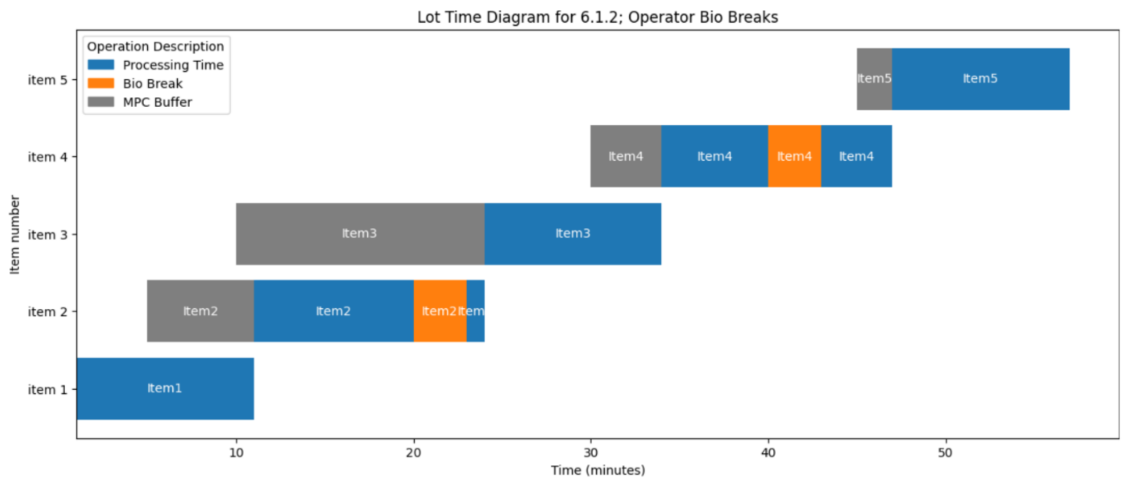
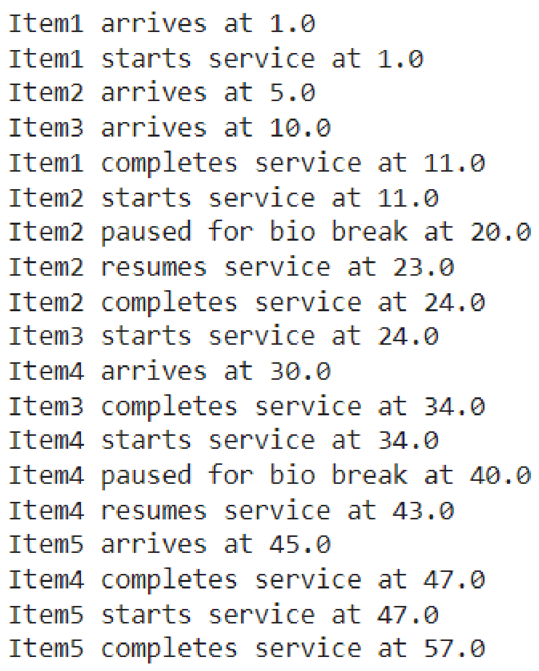


Figure 2: Lot Time Diagram 6.1.2

The following output of the simPy, confirms what was found in the computational part of the exercise and what was seen in the lot-time diagram.



```
Item1 arrives at 1.0
Item1 starts service at 1.0
Item2 arrives at 5.0
Item3 arrives at 10.0
Item1 completes service at 11.0
Item2 starts service at 11.0
Item2 paused for bio break at 20.0
Item2 resumes service at 23.0
Item2 completes service at 24.0
Item3 starts service at 24.0
Item4 arrives at 30.0
Item3 completes service at 34.0
Item4 starts service at 34.0
Item4 paused for bio break at 40.0
Item4 resumes service at 43.0
Item5 arrives at 45.0
Item4 completes service at 47.0
Item5 starts service at 47.0
Item5 completes service at 57.0
```

Figure 3: Output SimPy Simulation 6.1.2

6.1.3. Operator shift breaks

We'll represent the time spent by each item in the source and in the operator workstation. Given the deterministic arrival times and the operator's shift break schedule, we'll calculate how long each item spends in each station.

i) Time = 1.0 min:
- Item 1 arrives and starts processing immediately
- Processing time = 15 minutes (as in every case)
- Completion time without a break = 1.0 + 15 = 16.0 minutes

ii) Time = 5.0 min:
- Item 2 arrives while Item 1 is being processed and then it waits in the queue

iii) Time = 10.0 min:
- Item 3 arrives while Item 1 is being processed and then it waits in the queue

iv) Time = 16.0 min:
- Item 1 completes processing and item 2 starts processing
- Completion time without a break = 16.0 + 15 = 31.0 minutes

vi) Time = 31.0 min:
- Operator resumes work, with item 3 starts processing
- Completion time = 31.0 + 15 = 46.0 minutes

vii) Time = 46.0 min:

- Item 3 completes processing and item 4 starts processing
- Completion time = 46.0 + 15 = 61.0 minutes

viii) Time = 61.0 min:
- Item 4 completes processing and item 5 starts processing
- Completion time = 61.0 + 15 = 76.0 minutes

So, we end up with no shift breaks, because an item is being processed continuously and it is stated in the exercise that the operator does not interrupt its current work to go on a shift break. The first shift break will be done at 80 minutes then, where the processing of the items will be finished.

Item	Arrival time	Start processing	End processing	Shift breaks	Total time in system
1	1.0 min	1.0 min	16.0 min	0	15.0 min
2	5.0 min	16.0 min	31.0 min	0	15.0 min
3	10.0 min	31.0 min	46.0 min	0	15.0 min
4	30.0 min	46.0 min	61.0 min	0	15.0 min
5	45.0 min	61.0 min	76.0 min	0	15.0 min

Table 2: *Duration of item in each station with shift break schedule*

Respectively with the previous question it will also work for this part of the exercise and the lot-time diagram representing the outcome of the process will look like:

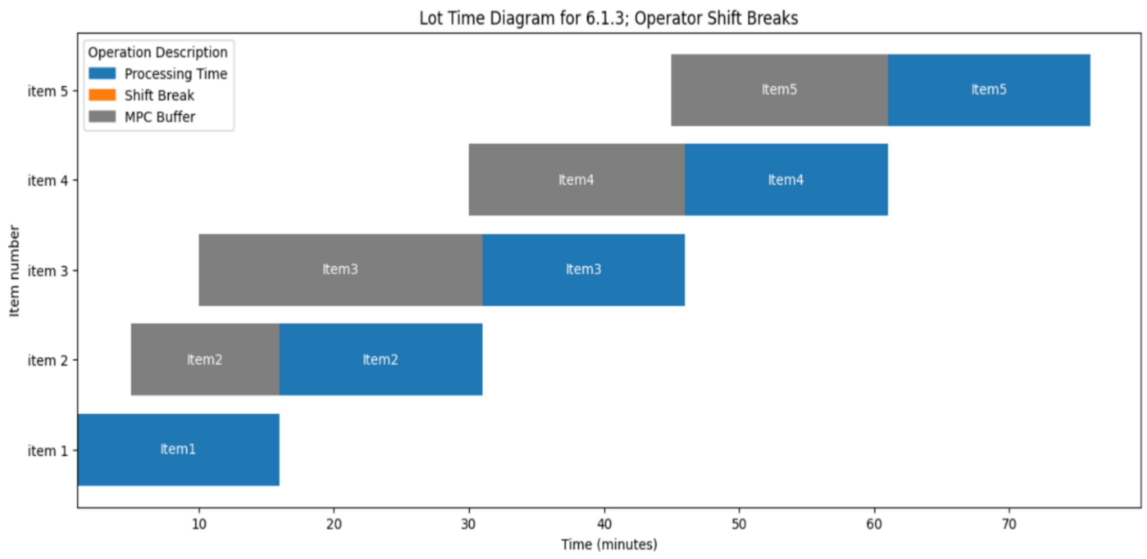
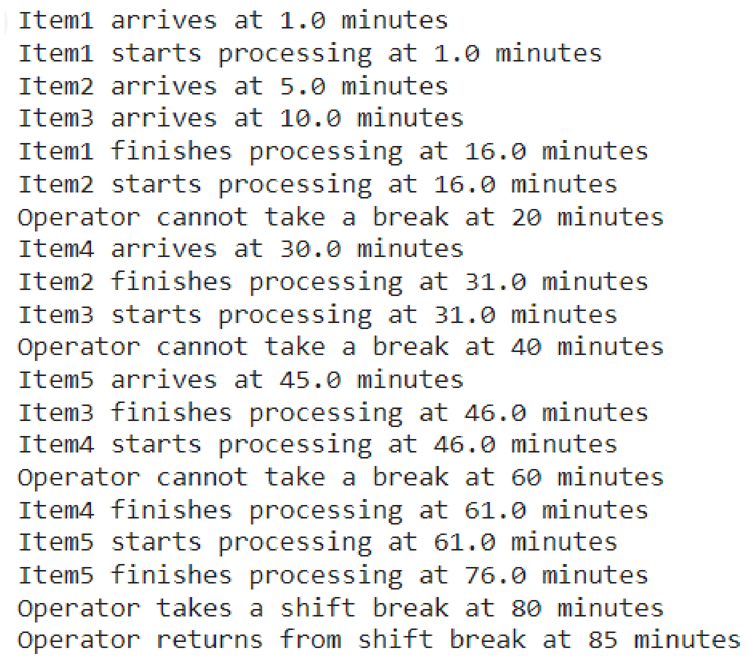


Figure 4: Lot Time Diagram 6.1.3

Below it is clear that the output of the code, leads to a verification of what was computed and depicted in the lot time diagram. There is no shift break in the first 80 minutes of the process and the reason for not including the break from minute 80 to minute 85 in the lot time diagram, is that it is after the end of the process.



```
Item1 arrives at 1.0 minutes
Item1 starts processing at 1.0 minutes
Item2 arrives at 5.0 minutes
Item3 arrives at 10.0 minutes
Item1 finishes processing at 16.0 minutes
Item2 starts processing at 16.0 minutes
Operator cannot take a break at 20 minutes
Item4 arrives at 30.0 minutes
Item2 finishes processing at 31.0 minutes
Item3 starts processing at 31.0 minutes
Operator cannot take a break at 40 minutes
Item5 arrives at 45.0 minutes
Item3 finishes processing at 46.0 minutes
Item4 starts processing at 46.0 minutes
Operator cannot take a break at 60 minutes
Item4 finishes processing at 61.0 minutes
Item5 starts processing at 61.0 minutes
Item5 finishes processing at 76.0 minutes
Operator takes a shift break at 80 minutes
Operator returns from shift break at 85 minutes
```

Figure 5: Output Simpy Simulation 6.1.3

Section 6.2. AGV fleet

Executed by 1893971 & 1887696

6.2.1. AGVs

a. Analytical calculations

In order to determine the throughput and flow time analytically, queueing theory will be applied for which some important parameters need to be determine first. These are determined based on the information provided in the assignment description;

$$a = average\ interarrival\ time = 8\ minutes$$

$$\lambda = arrival\ rate = \frac{1}{8}\ fixtures\ per\ minute$$

$$\sigma a = standard\ deviation\ of\ the\ interarrival\ time = \sqrt{\frac{1}{\lambda^2}} = \frac{1}{\lambda} = 8$$

$$mean\ service\ time = \ E(Vtime) = p = \frac{V_{\min}+V_{\max}}{2} = \frac{5+10}{2} = 7.5\ minutes$$

$$\sigma p = standard\ derviation\ of\ the\ service\ time \sqrt{\frac{1}{12} * (10-5)^2} = \sqrt{\frac{25}{12}}$$

$$c = number\ of\ servers = \ 2$$

$$\mu = service\ rate\ per\ server = \frac{1}{7{,}5}$$

Subsequently, it needs to be checked whether the system is stable, as this is an important assumption when using queueing theory. For a system to be stable, the arrival rate should be less than the total service rate. As there are two servers, the total service rate equals $\mu total = 2 * \frac{1}{7{,}5} = \frac{2}{7{,}5} \approx 0{,}267$, which is bigger than the arrival rate $\lambda = \frac{1}{8}$.

Once it is confirmed that the system is stable, the throughput (T) in the steady state is equal to the arrival rate. So, the $throughput = T = \lambda = \frac{1}{8}$ fixtures per minute.

The total flow time is calculated using the following two formulas:

$$Total\ flow\ time = average\ time\ in\ queue + p$$

$$Average\ time\ in\ queue = \ \mathrm{ATQ} = \left(\frac{p}{c}\right) \cdot \left(\frac{U^{\sqrt{2(c+1)}-1}}{1-U}\right) \cdot \left(\frac{CV_a^2 + CV_p^2}{2}\right)$$

Where:

$$Utilization = U = \frac{P}{c \cdot a} = \frac{7{,}5}{2 * 8} = \frac{15}{32}$$

$$Coefficient\ of\ variation\ for\ arrivals\ time = \ CV_{\mathrm{a}} = \frac{\sigma_{\mathrm{arrival}}}{\mu_{\mathrm{arrival}}} = \frac{8}{8} = 1$$

$$Coefficient\ of\ variation\ for\ processing\ time = CV_{\mathrm{P}} = \frac{\sigma}{t_e} = \frac{1.4434}{7.5} \approx 0.1925$$

Substituting these values gives the following outcome for the average time in queue:

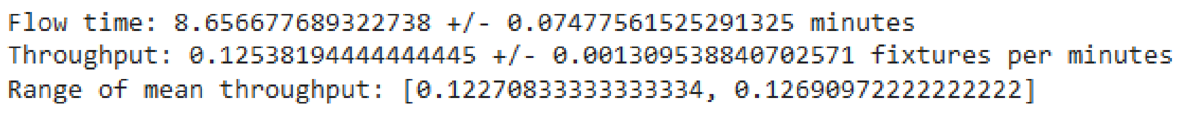
$$\mathrm{ATQ} = \left(\frac{7.5}{2}\right) \cdot \left(\frac{(\frac{15}{32})^{\sqrt{2(2+1)}} + 1}{1 - \frac{15}{32}}\right) \cdot \left(\frac{1 + (0.1925)^2}{2}\right) \approx 1.22$$

Thus,

$$Total\ flow\ Time = p + \mathrm{ATQ} = 7.5 + 1.22 = 8.72 \text{ minutes}$$

b. SimPy simulation

In order to check whether our analytical calculations for the throughput and total flow time are correct, a Simpy code was built to compare the outcomes against a simulated environment. The solutions, including the range of outcomes for the mean throughput, obtained via the simulation can be found below in *figure 6*.



```
Flow time: 8.656677689322738 +/- 0.07477561525291325 minutes
Throughput: 0.12538194444444445 +/- 0.001309538840702571 fixtures per minutes
Range of mean throughput: [0.12270833333333334, 0.12690972222222222]
```

Figure 6: SimPy output for 6.2.1 (throughput and total flow time)

As can be seen, the analytically determined throughput of $\frac{1}{8} = 0{,}125$ falls within the range of the mean throughput. The total flow time, which was earlier analytically determined to be 8,72 minutes, also falls within the range of the total flow time obtained using simulation.

The full Python simulation code accompanied by comments which was used to simulate this exercise can be found in the according notebook file of this project.

6.2.2. MPC to Work Centre

This exercise models two Automated Guided Vehicles (AGVs) transporting (and unloading) new fixtures from the joint buffer at the MPC to the loading dock at the Work Center. The model accounts for the workload of each AGV, ensuring that the AGV with the least workload is selected to transport the next item. The assumption is made that the time it takes for an AGV to unload a fixture into the loading dock at the Work Center is already included in the transport time. Additionally, it is stated in the description that if an AGV completed its transport to the Work Center and has no other assigned tasks, it will park there and wait for a new task. Initially, both AGVs are located at the MPC.

a. Lot time diagram

In order to simulate this model, a deterministic example is provided in the project description, along with multiple parameters, which can be found in the project description. *Figure 7* depicts the lot time diagram that has been created to get an overview of how long an item spends in a particular subsystem.

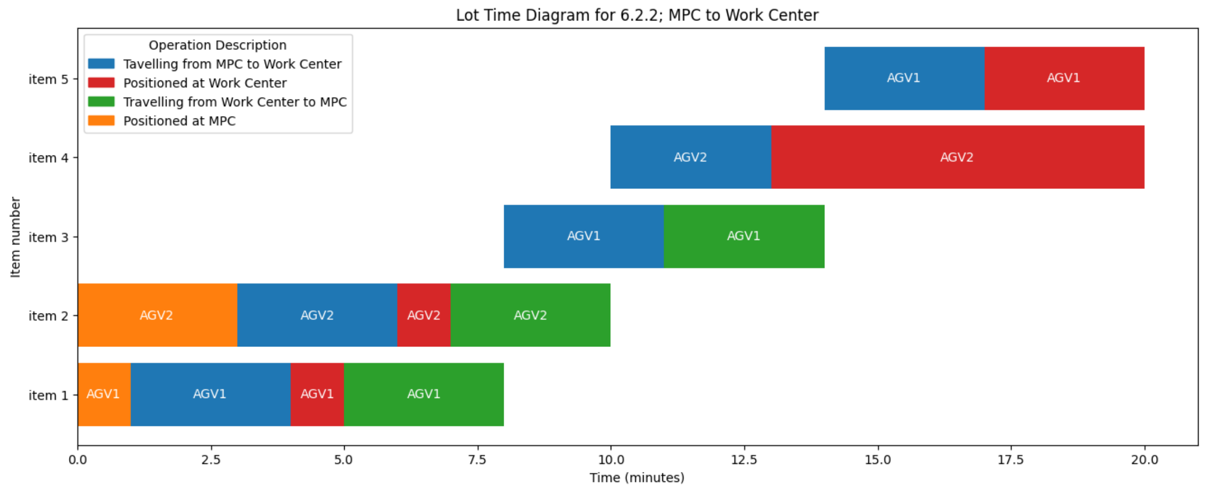
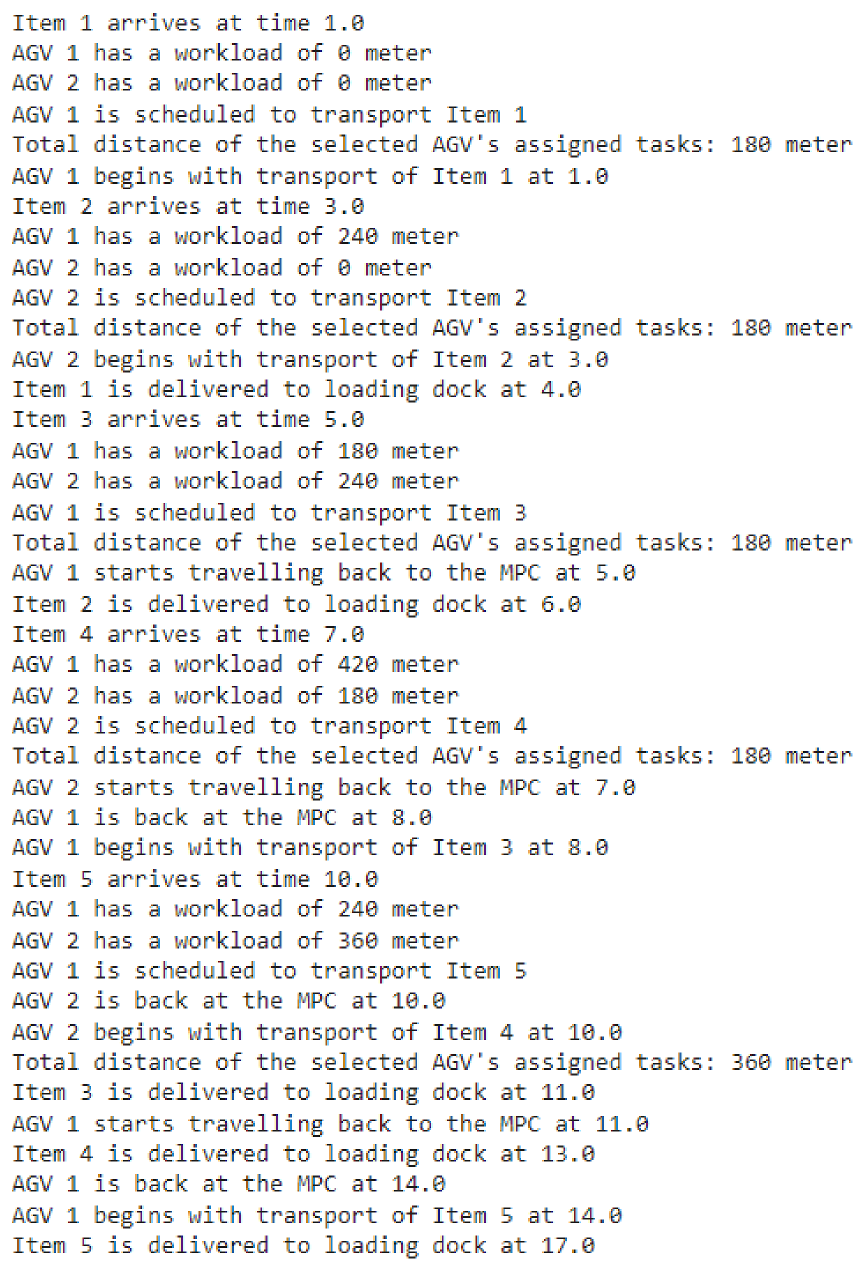


Figure 7: Lot Time Diagram 6.2.2

b. SimPy simulation

In order to solve the deterministic example, a simulation was created in Google Colab notebook using Python and the SimPy library. The output of this simulation can be found in *figure 8* below. The solution obtained through simulation matches with the lot-time diagram depicted in *figure 7*. The full Python simulation code accompanied by comments can be found in the according notebook file of this project.



```
Item 1 arrives at time 1.0
AGV 1 has a workload of 0 meter
AGV 2 has a workload of 0 meter
AGV 1 is scheduled to transport Item 1
Total distance of the selected AGV's assigned tasks: 180 meter
AGV 1 begins with transport of Item 1 at 1.0
Item 2 arrives at time 3.0
AGV 1 has a workload of 240 meter
AGV 2 has a workload of 0 meter
AGV 2 is scheduled to transport Item 2
Total distance of the selected AGV's assigned tasks: 180 meter
AGV 2 begins with transport of Item 2 at 3.0
Item 1 is delivered to loading dock at 4.0
Item 3 arrives at time 5.0
AGV 1 has a workload of 180 meter
AGV 2 has a workload of 240 meter
AGV 1 is scheduled to transport Item 3
Total distance of the selected AGV's assigned tasks: 180 meter
AGV 1 starts travelling back to the MPC at 5.0
Item 2 is delivered to loading dock at 6.0
Item 4 arrives at time 7.0
AGV 1 has a workload of 420 meter
AGV 2 has a workload of 180 meter
AGV 2 is scheduled to transport Item 4
Total distance of the selected AGV's assigned tasks: 180 meter
AGV 2 starts travelling back to the MPC at 7.0
AGV 1 is back at the MPC at 8.0
AGV 1 begins with transport of Item 3 at 8.0
Item 5 arrives at time 10.0
AGV 1 has a workload of 240 meter
AGV 2 has a workload of 360 meter
AGV 1 is scheduled to transport Item 5
AGV 2 is back at the MPC at 10.0
AGV 2 begins with transport of Item 4 at 10.0
Total distance of the selected AGV's assigned tasks: 360 meter
Item 3 is delivered to loading dock at 11.0
AGV 1 starts travelling back to the MPC at 11.0
Item 4 is delivered to loading dock at 13.0
AGV 1 is back at the MPC at 14.0
AGV 1 begins with transport of Item 5 at 14.0
Item 5 is delivered to loading dock at 17.0
```

Figure 8: Output SimPy Simulation 6.2.2

6.2.3. AGV Charging

In this exercise, a lot-time diagram had to be created in order to illustrate how long each item spends in the subsystem, including transport and charging times. Then, for sub question b, a SimPy code had to be written to simulate the AGV's operations, which involves transporting items and managing its battery charge. Finally, it was verified that the simulation results align with the lot-time diagram that was created for sub question a.

a. Lot Time Diagram

The lot-time diagram indicating the outcomes of the process can be found in *figure 9* below.

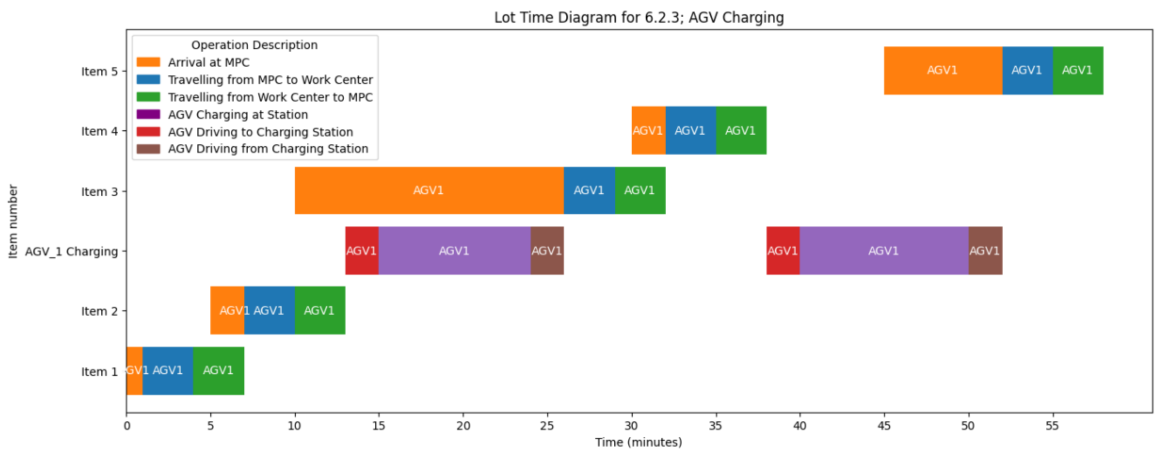
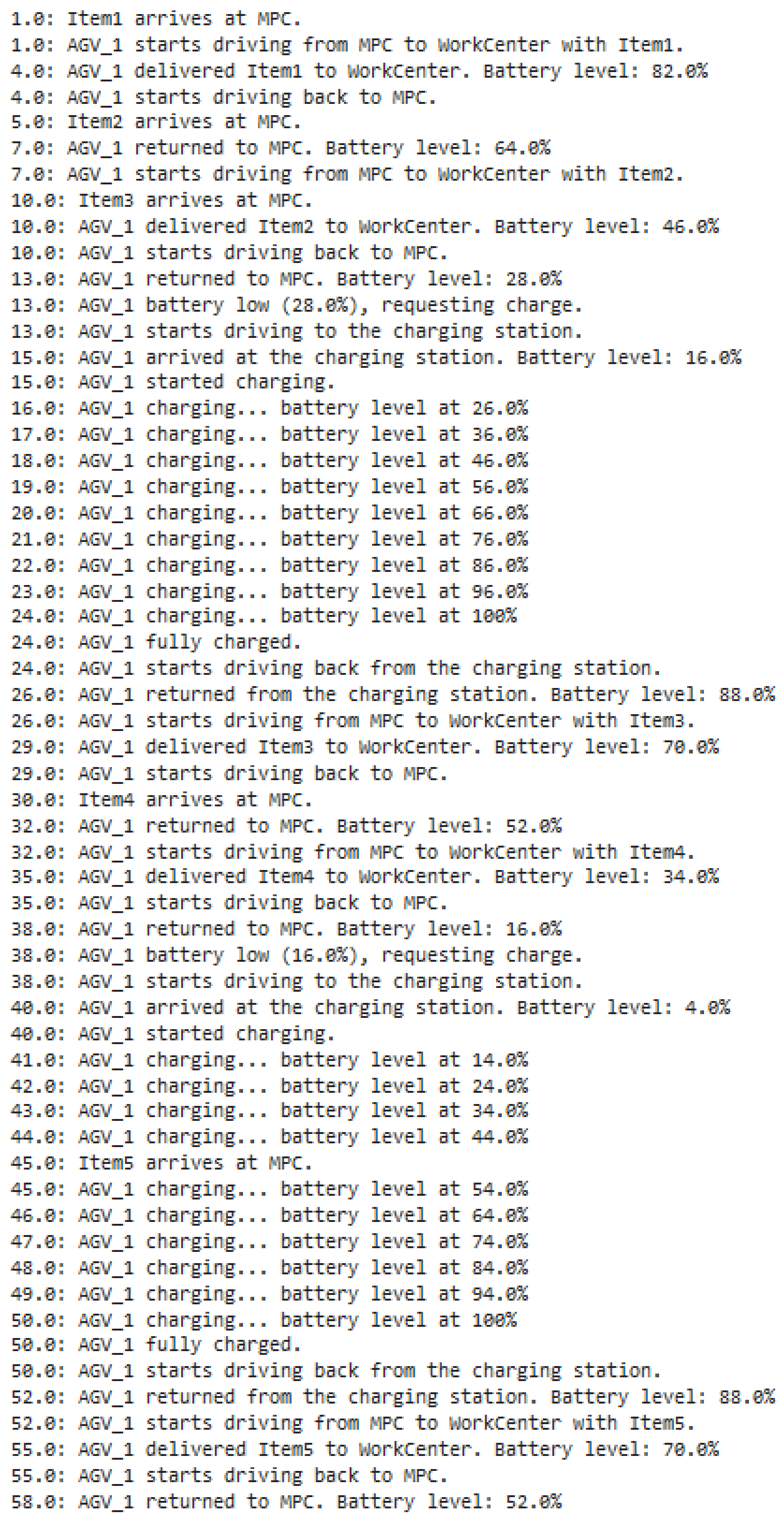


Figure 9: Lot Time Diagram 6.2.3

b. SimPy Simulation

For this exercise, the situation with deterministic values of sub question a of this exercise is simulated. The Python code that was used for this exercise is in the Jupyter Notebook file and can be found in the ZIP file. Below in *figure 10* the output of the code can be found. It matches with the lot time diagram in *Figure 9*, which means that the situation is simulated correctly.



```
1.0: Item1 arrives at MPC.
1.0: AGV_1 starts driving from MPC to WorkCenter with Item1.
4.0: AGV_1 delivered Item1 to WorkCenter. Battery level: 82.0%
4.0: AGV_1 starts driving back to MPC.
5.0: Item2 arrives at MPC.
7.0: AGV_1 returned to MPC. Battery level: 64.0%
7.0: AGV_1 starts driving from MPC to WorkCenter with Item2.
10.0: Item3 arrives at MPC.
10.0: AGV_1 delivered Item2 to WorkCenter. Battery level: 46.0%
10.0: AGV_1 starts driving back to MPC.
13.0: AGV_1 returned to MPC. Battery level: 28.0%
13.0: AGV_1 battery low (28.0%), requesting charge.
13.0: AGV_1 starts driving to the charging station.
15.0: AGV_1 arrived at the charging station. Battery level: 16.0%
15.0: AGV_1 started charging.
16.0: AGV_1 charging... battery level at 26.0%
17.0: AGV_1 charging... battery level at 36.0%
18.0: AGV_1 charging... battery level at 46.0%
19.0: AGV_1 charging... battery level at 56.0%
20.0: AGV_1 charging... battery level at 66.0%
21.0: AGV_1 charging... battery level at 76.0%
22.0: AGV_1 charging... battery level at 86.0%
23.0: AGV_1 charging... battery level at 96.0%
24.0: AGV_1 charging... battery level at 100%
24.0: AGV_1 fully charged.
24.0: AGV_1 starts driving back from the charging station.
26.0: AGV_1 returned from the charging station. Battery level: 88.0%
26.0: AGV_1 starts driving from MPC to WorkCenter with Item3.
29.0: AGV_1 delivered Item3 to WorkCenter. Battery level: 70.0%
29.0: AGV_1 starts driving back to MPC.
30.0: Item4 arrives at MPC.
32.0: AGV_1 returned to MPC. Battery level: 52.0%
32.0: AGV_1 starts driving from MPC to WorkCenter with Item4.
35.0: AGV_1 delivered Item4 to WorkCenter. Battery level: 34.0%
35.0: AGV_1 starts driving back to MPC.
38.0: AGV_1 returned to MPC. Battery level: 16.0%
38.0: AGV_1 battery low (16.0%), requesting charge.
38.0: AGV_1 starts driving to the charging station.
40.0: AGV_1 arrived at the charging station. Battery level: 4.0%
40.0: AGV_1 started charging.
41.0: AGV_1 charging... battery level at 14.0%
42.0: AGV_1 charging... battery level at 24.0%
43.0: AGV_1 charging... battery level at 34.0%
44.0: AGV_1 charging... battery level at 44.0%
45.0: Item5 arrives at MPC.
45.0: AGV_1 charging... battery level at 54.0%
46.0: AGV_1 charging... battery level at 64.0%
47.0: AGV_1 charging... battery level at 74.0%
48.0: AGV_1 charging... battery level at 84.0%
49.0: AGV_1 charging... battery level at 94.0%
50.0: AGV_1 charging... battery level at 100%
50.0: AGV_1 fully charged.
50.0: AGV_1 starts driving back from the charging station.
52.0: AGV_1 returned from the charging station. Battery level: 88.0%
52.0: AGV_1 starts driving from MPC to WorkCenter with Item5.
55.0: AGV_1 delivered Item5 to WorkCenter. Battery level: 70.0%
55.0: AGV_1 starts driving back to MPC.
58.0: AGV_1 returned to MPC. Battery level: 52.0%
```

Figure 10: Output SimPy Simulation 6.2.3

Section 6.3. Work Center

Executed by 1811347 & 1798618

6.3.1. Robot

Step 1: Converting Time Units

- **Simulation run time:**
    - Given: 50 days
    - Converted to minutes: 50 × 24× 60 = 72000 minutes
- **Warmup period**:
    - Given: 10 days
    - Converted to minutes: 10 × 24 × 60 = 14400 minutes

Step 2: Calculating Throughput

- Throughput is defined as the rate at which fixtures are completed by the robot. It is generally equal to the rate at which tasks are handled when the system is stable, and in a simple system like this (single server, infinite queue), it is primarily determined by the slowest part of the process (bottleneck).

Calculation:

1. **Arrival Rate (λ):**
    - Mean arrival rate: $\lambda = 41 = 0.25\lambda = \frac{1}{4} = 0.25$ fixtures per minute

2. **Service Rate (μ)**:
    - Mean service time E[R]: 2 minutes
    - Service rate: $\mu = \frac{1}{E[R]} = 21 = 0.5$ fixtures per minute
3. **Throughput**:
    - Since $\mu > \lambda$, the system is stable.
    - Throughput rate is limited by the arrival rate (λ):
        - Throughput rate = 0.25 fixtures per minute

Step 3: Calculating Flow Time

- The total time a fixture spends in the system, including both waiting time in the queue and the service time.

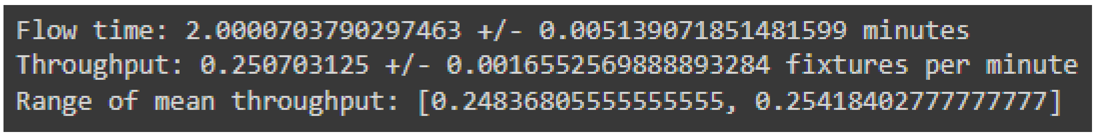
Calculation Using Little's Law and M/M/1 Queuing Theory:

6. **Average Service Time**:
    - E[R] = 2 minutes
7. **Expected Number of Fixtures in the System (L)**:
    - According to Little's Law: $L = \lambda \times W$
8. **Average Flow Time (W)**:

- For an M/M/1 queue: $W = \frac{1}{(\mu - \lambda)}$
- $W = \frac{1}{0.5-0.25} = \frac{1}{0.25}$ = 4 minutes

b) Below it appears the output of the simPy code corresponding to this section.

In order to verify the above calculations, the simpy template provided along with our additions was utilized:



```
Flow time: 2.0000703790297463 +/- 0.005139071851481599 minutes
Throughput: 0.250703125 +/- 0.0016552569888893284 fixtures per minute
Range of mean throughput: [0.24836805555555555, 0.25418402777777777]
```

Figure 10.5. Lot-Time Diagram for Exercise 6.3.2

6.3.2. Internal Buffers to Machines

In this exercise, a lot time-diagram was required to illustrate the time each item spends in the subsystem (including the time spent at the loading dock, the robot handling time, and the time spent in the buffer/machine). This diagram is essential for visualizing the process flow, in order to identify and tackle possible bottlenecks present in the system. The diagram is shown below in figure 11.

a. Lot Time Diagram

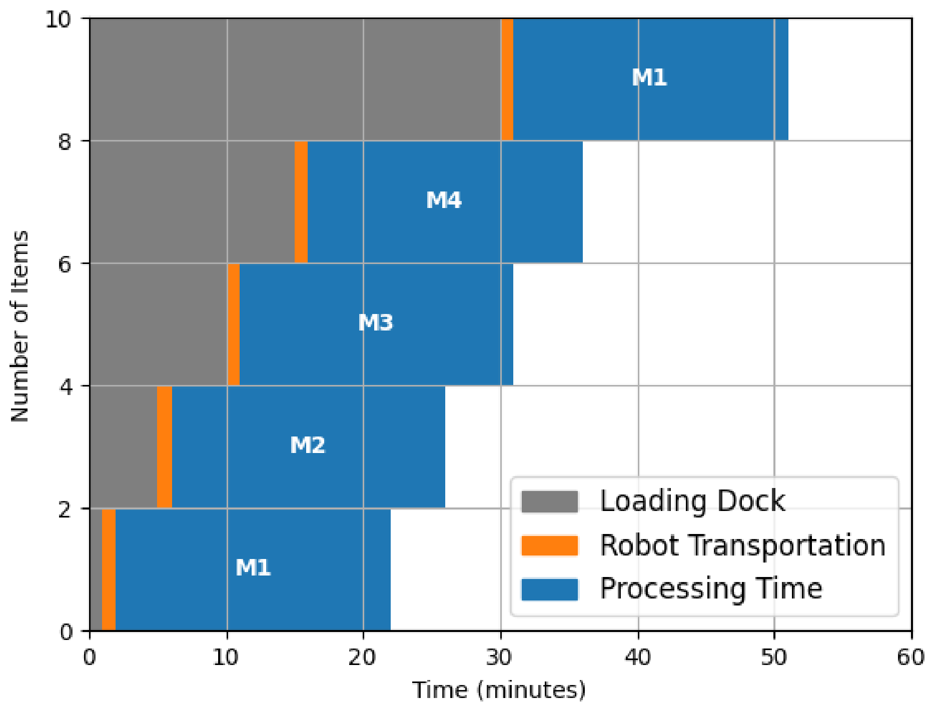
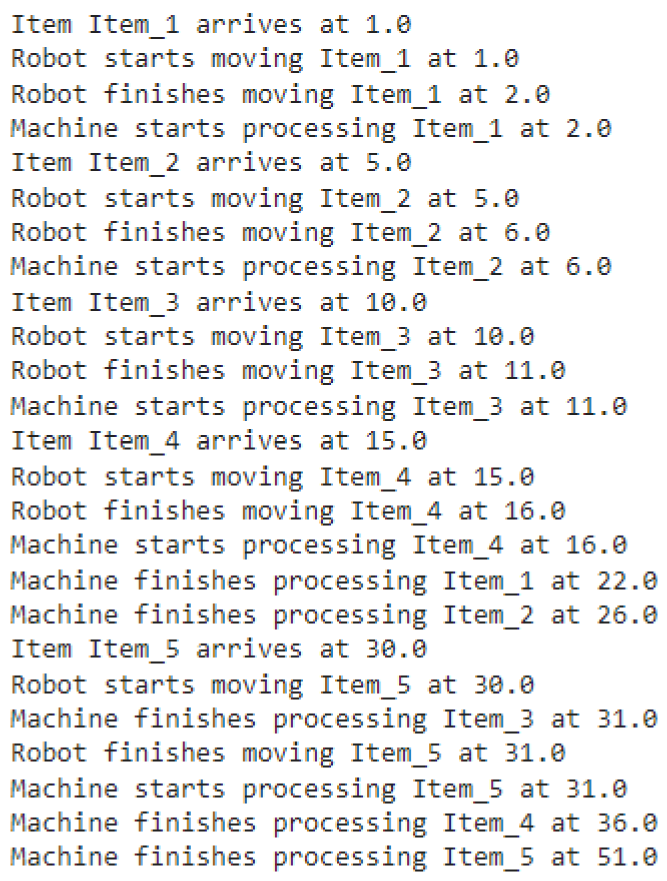


Figure 11. Lot-Time Diagram for Exercise 6.3.2

b. SimPy Simulation

For this part, the situation described in part a was simulated by using the SimPy python library. The python code used for this part can be found in the included .ipynb file in the zip. The output can be seen in figure 12, and as it can be seen, it matches the results from the lot-time diagram, which confirms the accuracy of the simulation.



```
Item Item_1 arrives at 1.0
Robot starts moving Item_1 at 1.0
Robot finishes moving Item_1 at 2.0
Machine starts processing Item_1 at 2.0
Item Item_2 arrives at 5.0
Robot starts moving Item_2 at 5.0
Robot finishes moving Item_2 at 6.0
Machine starts processing Item_2 at 6.0
Item Item_3 arrives at 10.0
Robot starts moving Item_3 at 10.0
Robot finishes moving Item_3 at 11.0
Machine starts processing Item_3 at 11.0
Item Item_4 arrives at 15.0
Robot starts moving Item_4 at 15.0
Robot finishes moving Item_4 at 16.0
Machine starts processing Item_4 at 16.0
Machine finishes processing Item_1 at 22.0
Machine finishes processing Item_2 at 26.0
Item Item_5 arrives at 30.0
Robot starts moving Item_5 at 30.0
Machine finishes processing Item_3 at 31.0
Robot finishes moving Item_5 at 31.0
Machine starts processing Item_5 at 31.0
Machine finishes processing Item_4 at 36.0
Machine finishes processing Item_5 at 51.0
```

Figure 12. Simulation Output for Exercise 6.3.2

6.3.3. Machines to Internal Buffers

This exercise was highly dependent on the previous exercise in that it utilizes the same machine processing function to model the processing of an item at one of four machines. Following the processing, however, the machine will dispatch the finished fixture to its internal buffers and processes a new fixture when required. When the finished fixture is in the internal buffer, the robot will then move it to the loading dock.

a. Lot Time Diagram

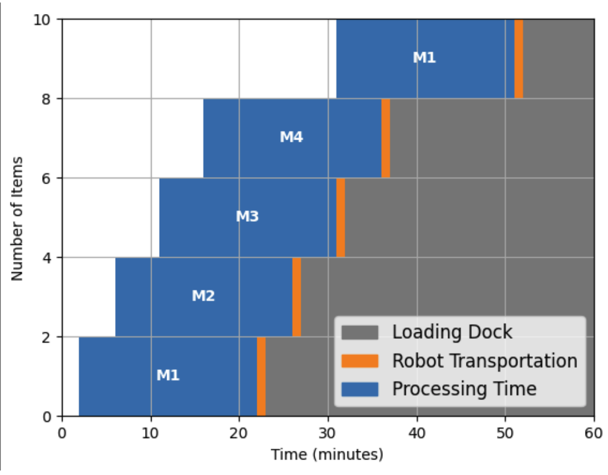
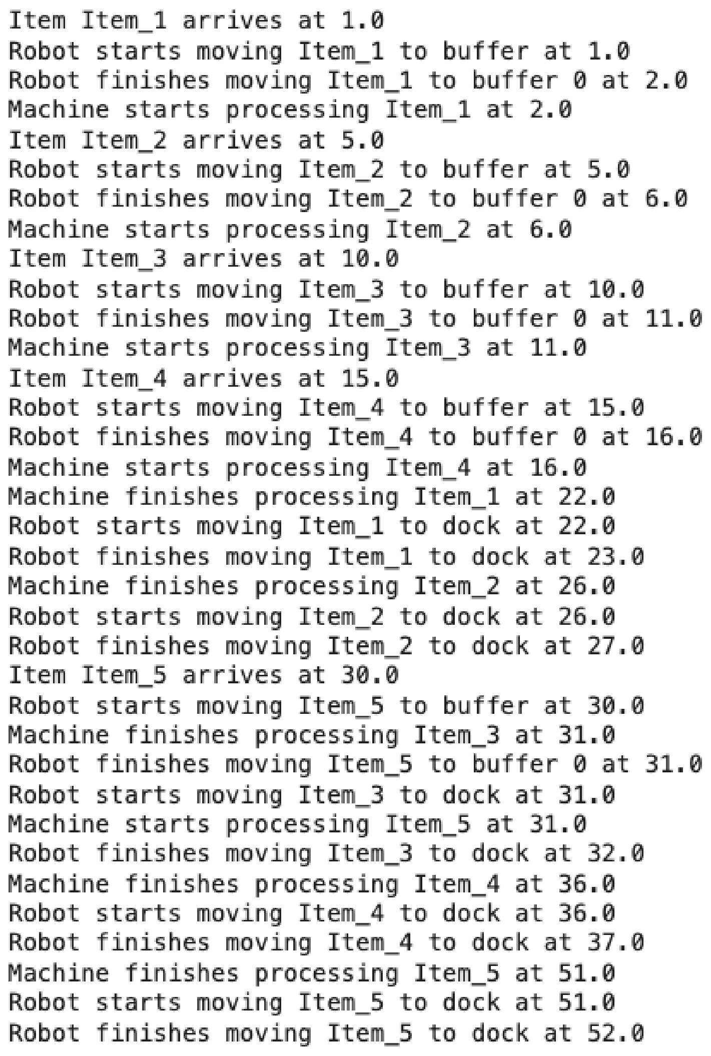


Figure 13 Output Lot Time Diagram 6.3.3

b. SimPy Simulation

In part b of this exercise, we were expected to simulate the process modelled in the lot time diagram for exercise a. Below in Figure X is the output of the code in which it can be clearly seen that the values gathered from the simulation correspond with those of the diagram. This indicates that the simulation model is correct.



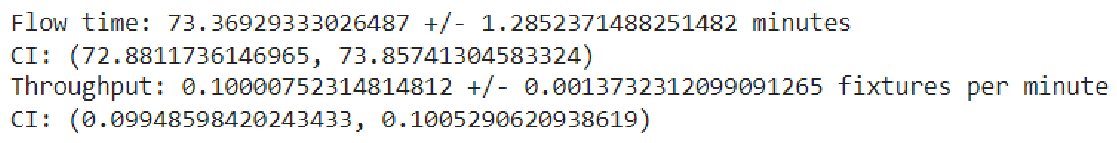
```
Item Item_1 arrives at 1.0
Robot starts moving Item_1 to buffer at 1.0
Robot finishes moving Item_1 to buffer 0 at 2.0
Machine starts processing Item_1 at 2.0
Item Item_2 arrives at 5.0
Robot starts moving Item_2 to buffer at 5.0
Robot finishes moving Item_2 to buffer 0 at 6.0
Machine starts processing Item_2 at 6.0
Item Item_3 arrives at 10.0
Robot starts moving Item_3 to buffer at 10.0
Robot finishes moving Item_3 to buffer 0 at 11.0
Machine starts processing Item_3 at 11.0
Item Item_4 arrives at 15.0
Robot starts moving Item_4 to buffer at 15.0
Robot finishes moving Item_4 to buffer 0 at 16.0
Machine starts processing Item_4 at 16.0
Machine finishes processing Item_1 at 22.0
Robot starts moving Item_1 to dock at 22.0
Robot finishes moving Item_1 to dock at 23.0
Machine finishes processing Item_2 at 26.0
Robot starts moving Item_2 to dock at 26.0
Robot finishes moving Item_2 to dock at 27.0
Item Item_5 arrives at 30.0
Robot starts moving Item_5 to buffer at 30.0
Machine finishes processing Item_3 at 31.0
Robot finishes moving Item_5 to buffer 0 at 31.0
Robot starts moving Item_3 to dock at 31.0
Machine starts processing Item_5 at 31.0
Robot finishes moving Item_3 to dock at 32.0
Machine finishes processing Item_4 at 36.0
Robot starts moving Item_4 to dock at 36.0
Robot finishes moving Item_4 to dock at 37.0
Machine finishes processing Item_5 at 51.0
Robot starts moving Item_5 to dock at 51.0
Robot finishes moving Item_5 to dock at 52.0
```

Figure 14 Output SimPy Simulation 6.3.3

Section 6.4. System Integration

6.4.1. Entire System

a) This exercise aims to integrate the three subsystems that have been built as simulations in the previous sections. An explanation of the working of the entire, integrated system along with the value of multiple parameters are provided in the assignment description. With this given data, a SimPy code was built for simulating this system in order to find the mean and standard deviation of throughput and flowtime. Additionally, 95% confidence intervals where obtained for these parameters. The entire Python code that was used for this exercise is in the Jupyter Notebook file and can be found in the ZIP file. Figure 15 below depicts the output of the simulation.



```
Flow time: 73.36929333026487 +/- 1.2852371488251482 minutes
CI: (72.8811736146965, 73.85741304583324)
Throughput: 0.10000752314814812 +/- 0.0013732312099091265 fixtures per minute
CI: (0.09948598420243433, 0.1005290620938619)
```

Figure 15: Output SimPy from integration

b) After we were done running the SimPy code 30 times, we had to verify somehow that the output of the program was correct, or at least close to what we wanted. So, we thought of verifying our code by replacing the stochastic variables with deterministic variables and create a lot-time diagram to make sure that our output is logical, but that would take a lot of time, so we came up with some more ideas. First of all, we started with logical consistency checks, so we verified that the initial conditions and parameters are set correctly (e.g., arrival rate, number of machines, buffer capacities, etc). Then, we proceed by making sure that items move through the system as expected, from arrival to processing, and then to the final destination. Last but not least, we checked the resources like AGVs, machines, and operators are utilized correctly and not exceeding their capacities.

We also checked if the distributions of inter-arrival times, processing times, and transportation times match the expected distributions (e.g., exponential, uniform) and verified that the computed throughput and flow times are within the expected range based on the input parameters. So, after ensuring that the collected statistics (e.g., mean and confidence intervals) make sense and that the simulation runs for enough replications to get stable estimates of performance measures, we decided to compare the simulation results with analytical models. For example, use queuing theory to estimate expected throughput, flow times and the VUT formula mentioned in the exercises above. So, we thought about adding all the process times to the VUT formula, because the formula itself expresses the time in the queue, but the goal is to calculate the overall time of the process. By summing up the processing time (Ptime), the building time (Btime), the detaching time (DTime), the AGV time, the queuing times in between processes with the robot in between the dock and internal buffers, we can calculate the total processing time.

Figure 16: Calculations for validation

6.4.2. Best Design

The provided data and analysis come from tests of your code in order to investigate different system configurations to see if they are able to improve the performance measure, involving various combinations of stations, AGVs, and machines. The following data is a breakdown of the results:

Combinations*	Flow Time [minutes]	Throughput [minutes]
(2, 2, 4)	73.36929333026487	0.10000752314814812
(3, 2, 4)	66.30015704947235	0.10039930555555558
(2, 3, 4)	72.81109408118274	0.1001261574074074
(2, 2, 5)	69.82616985956868	0.10018865740740741
(3, 3, 4)	65.42313128508337	0.1000966435185185
(3, 3, 5)	59.583358525699296	0.10007812499999998
(3, 2, 5)	60.3868790007414	0.09999016203703702

(2, 3, 5)	69.87218853513907	0.10015567129629631

* Combinations: (No. stations, No. AGVs, No. machines)

Figure 15 Results table depicting the flow time and throughout of various station, machine and AGV combinations

Based on this data, a ranking was devised for both throughput and flowtime:

Combinations	Flow Time Ranking	Throughput Ranking
(2, 2, 4)	8	2
(3, 2, 4)	4	8
(2, 3, 4)	7	5
(2, 2, 5)	5	7
(3, 3, 4)	3	4
(3, 3, 5)	1	3
(3, 2, 5)	2	1
(2, 3, 5)	6	6

* The rankings are in reversed order, with 1 being the lowest and 7 being the highest

Figure 16 Ranking based on the results of figure X

- **(3, 3, 5)** has the best flow time but ranks 3rd in throughput.
- **(3, 2, 5)** has the best throughput and the 2nd best flow time.
- **(2, 2, 4)** ranks lowest in flow time but has the 2nd highest throughput.

This analysis helps in understanding how different combinations of stations, AGVs, and machines affect the performance of the system in terms of flow time and throughput. The combination of three stations, three AGVs, and five machines appears optimal for flow time, while three stations, two AGVs, and five machines are optimal for throughput.