



**Pannon Egyetem
Intézményi Tudományos Diákköri
Konferencia
2024**



**Diszkrét eseményű szimulátor fejlesztése preventív karbantartás
ütemezéshez**

Készítette: Németh Levente

Témavezetők: Dr. Ruppert Tamás, Darányi András Pál

Mérnöki Kar, Rendszermérnöki Intézeti Tanszék

A pályamunka lezárásának dátuma: 2024.11.05.

Tartalomjegyzék

1. Irodalmi áttekintés	1
2. Kidolgozott módszertan	5
3. Az ipari folyamat és a fejlesztett modell bemutatása	6
3.1. Az Ipar 5.0 laboratóriumban lévő folyamat bemutatása	6
3.2. A szimulátor által egy hét alatt legyártandó termék mennyiség meghatározása	8
3.3. Gépidők és ütemezés meghatározása	10
4. Fejlesztett szimulációs modell és validációja	13
4.1. A SimPy-ban használt objektumok	13
4.2. A rendszer bemeneteinek és az ütemezésnek a megvalósítása	14
4.3. A csak szállítás funkcióval rendelkező gépek	16
4.4. Egyszerű feladattal rendelkező gépek	18
4.5. Forgóasztal	20
4.6. Érlelő	22
4.7. Kiszállítás	23
4.8. Validálás	24
5. Karbantartási és gazdasági adatok	25
5.1. Bevétel adatok és hibamentes esetben a profit alakulása az időben.	25
5.2. A meghibásodáshoz és karbantartáshoz kapcsolódó paraméterek	27
6. Monte Carlo módszerrel támogatott karbantartás ütemezés	29
6.1. Egy meghibásodáshoz kapcsolódó karbantartás meghatározása	29
6.2. Eredmények	34
7. Összefoglalás	36
Irodalomjegyzék	41

Kivonat

Dolgozatomban egy tejüzem folyamatai alapján létrehozott laboratóriumi gyártósor preventív karbantartási ütemtervét fogom meghatározni a Pannon Egyetem Ipar 5.0 laborjában, amelyhez egy SimPy (Python) alapú szimulációs modellt fejlesztettem. Az első fejezetben leírom az irodalmi áttekintést, majd a módszertant, és ezután jön a dolgozatom tartalmi része, amit négy részre bontottam. Az elsőben meghatározom, hogy milyen paraméterekkel lesz lehetséges az Ipar 5.0 laboratóriumot tejüzem szimulátoraként kezelni. Ez a gyártani kívánt termékek kiválasztását, a gyártott termékek darabszámának és a ki-szerelési méretek meghatározását, illetve a gépek munkaidejének paraméterezését foglalja magába. A második részben konkrétan leírom, hogyan működik az Ipar 5.0 laboratórium hibamentes SimPy szimulátora, hogy melyik gépnél mire kellett ügyelnem, illetve hogyan oldottam meg azt, hogy a program jól kövesse az ott megvalósuló folyamatot. A harmadik részben azokat az adatokat közlöm, ami a bevétel és költségek modellezéséhez fontos, illetve a meghibásodáshoz és karbantartáshoz kapcsolódó paramétereket írom le, illetve ezeket honnét szereztem, vagy hogy hogyan számoltam ki. Végezetül a negyedik részben leírom, hogyan határoztam meg a karbantartási időpontot egy konkrét meghibásodás elkerülése érdekében, illetve Monte Carlo módszerrel meghatározom ezen módó megközelítéssel a karbantartási ütemtervet, ami figyelembe veszi a karbantartáshoz és meghibásodáshoz kapcsolódó költségeket.

Kulcsszavak: Karbantartás, Monte Carlo, Diszkrét eseményű szimulátor

1. fejezet

Irodalmi áttekintés

A gyártási rendszerekben lévő gépeknél felmerülő nagy probléma, hogy amikor meghibásodnak, akkor átmenetileg kiesnek a termelésből, ezzel gátolva a többi gép munkáját, vagy úgy, hogy nem kapják meg a szükséges anyagot, vagy úgy, hogy nem tudják továbbadni a félkész terméket, mert megtelt a következő gép puffere. Ebből adódóan nem csak a javíttatás költsége merül fel ilyenkor, hanem a termelés kiesés miatt a megrendeléseket nem lehet időben teljesíteni, ami kötbért von maga után. Viszont a túl gyakran végzett karbantartás is kisebb bevételt eredményez, mivel ilyenkor le kell állítani a gépeket, illetve ebben az esetben is meg kell fizetni a karbantartókat. Ezért fontos, hogy megfelelő időközönként végezzék el a cégek a karbantartást.

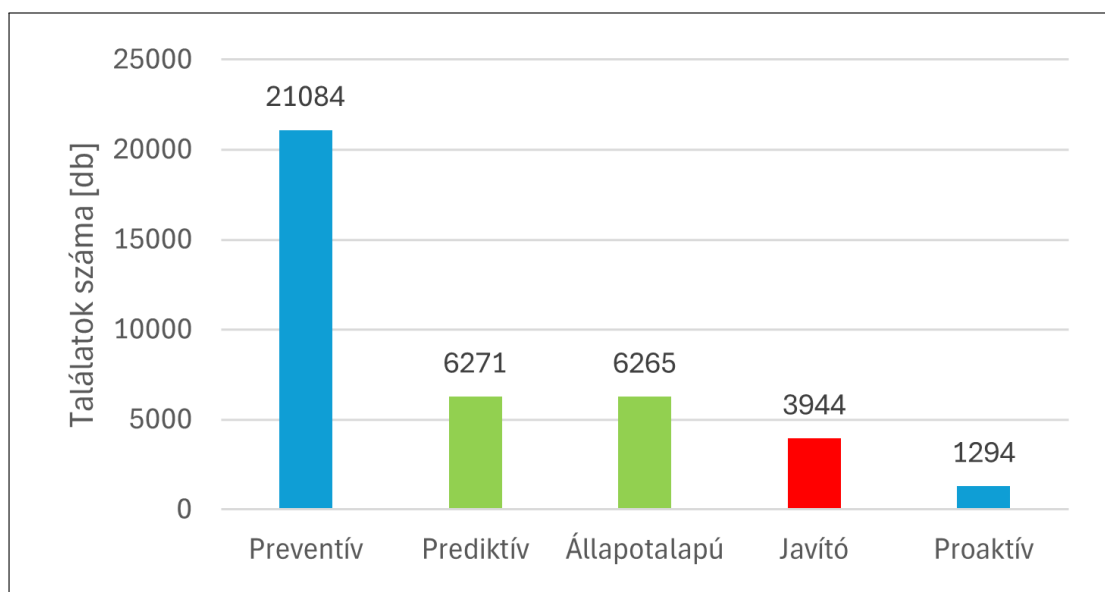
A téma fontosságát jelzi, hogy Gackowiec Paulina 37 karbantartáshoz kapcsolódó kutatást dolgozott fel, amikben összesen 10 fajta megközelítést írtak le. [1] Ezután megkereste, hogy az öt leggyakrabban megjelenő karbantartási terv hány kutatásban szerepel kulcszóként a ScienceDirect, Baztech és a Scopus felületeken lévő cikkekben. Összesítve több mint 38000 cikket talált. A kutatásban lévő karbantartási módok összefoglalva három nagy csoportba oszthatók:

Az első lényege, hogy addig használják a gépeket, amíg az tönkre nem megy, és akkor javítják. Ebbe a csoportba tartozik például a javító karbantartás, ahol a karbantartással nem foglalkoznak [2]. Az ide tartozó módszerek csak a második világháborúig voltak használatosak. A mai korban nem relevánsak, ezt mutatja az is, hogy a javító karbantartáshoz kapcsolódóan csak 3944 találat volt, így ez a csoport kapta a legkevesebb találatot. Ezen karbantartások hátránya, hogy kiszámíthatatlanok lesznek a leállások, és nagy költségek merülnek fel a javításhoz kapcsolódóan.

A második csoportnál ütemezett folyamatok mennek végbe és nem a rendszer jelenlegi állapotából indulnak ki. Ez a karbantartási forma a 70 évekig fejlődött. Ide tartozik a proaktív karbantartás, ahol az ellenőrzések vannak beütemezve és annak eredménye alapján tartják karban a gépet [3]. Ez a módszer bekerült a top ötbe, de csak 1294

kutatás foglalkozott vele. Ennek ellenére a kutatásokban mai napig gyakran megjelenik ez a csoport, mivel a legtöbb találat az ebben a csoportba tartozó preventív karbantartásnál volt. Összesen 21084 találat. Ennél a megközelítésnél magát a karbantartást ütemezik be [3].

A harmadik csoportnál adatgyűjtéssel pontos képet lehet alkotni a gép állapotáról és ez alapján tudják ütemezni a karbantartást. Ide tartozik az állapotalapú és a prediktív karbantartás. Előbbinél csak a szenzorok jelenlegi állapotát [4] utóbbinál a trendeket is figyelembe veszik ahhoz, hogy meghatározzák a karbantartási időpontokat [3]. Az állapot alapú karbantartással 6265 cikk foglalkozott. A prediktívvel pedig 6271. Az egyes típusú karbantartásokhoz kapcsolódó kutatások számát mutatja összefoglalva az 1.1 ábra, ahol az első csoportban lévő karbantartási típust pirossal, a második csoportban lévőket kékkel, a harmadikban lévőket meg zölddel jelöltem.



1.1. ábra. A különböző karbantartási típusokkal foglalkozó cikkek száma

Az eddigiekben bemutattam, hogy milyen bőséges, a karbantartáshoz kapcsolódó szakirodalom. Ezen felül, viszont nem csak a kutatókat, de a gazdaság szereplőit is érdekli ez a téma, mert már 2002-ben a Brit Királyi Légierő támogatásával született egy olyan kutatás, ami a repülőgépek ellenőrzésekor keletkező adatok alapján segít a karbantartás ütemezését [5].

A karbantartás fontossága a fentiek alapján jól látható. Külön a tejiparban is több kutatást végeztek ehhez kapcsolódóan. Az egyikben a cégtől meghibásodási és karbantartási adatok alapján javították a tejpor- és túrógyárak karbantartásának ütemezését [6]. A másokban hasonló adatok alapján Petri-háló segítségével végezték el a karbantartás optimalizálását egy tejüzem esetén [7]. Ezen kutatásokban közlik, hogy az adatokra milyen eloszlást sikerült illeszteniük. Ennek köszönhetően képesek lesznek majd a meghibásodásokat

szimulálni. A futószalaggal történő szállításhoz kapcsolódó meghibásodási paramétereket pedig egy másik kutatásból [8] vettem át.

Alapvetően a munkám a mostani legfejlettebb szimulációs modellezésnek a digitális iker koncepciójának egy kezdő lépése, mert az valós idejű adatokat használ fel a döntéshozatal megkönnyebbítéséhez [9], míg én korábbi meghibásodási adatokra illesztett meghibásodási valószínűséggel számolok, azaz a rendszer digitális modelljét és a döntéshozáshoz szükséges szimulációs modellt fejlesztettem ki. A létrehozott modell könnyedén átalakítható más hasonló struktúrájú rendszerek karbantartási idejének meghatározásához szimplán azzal, hogy újra paraméterezik a programot. Mindkettő esetben a DES (Discrete-event simulation) a szimuláció magja [9]. Ebben az esetben is újra el kell végezni a sorkiegyenlítést, ami, ha a két termék gyártása nem különbözik nagy mértékben, akkor egy modelles determinisztikus egyenes típusú (SM_D_S) problémaként kezelhető [10]. Viszont fontos, hogy alapvetően a sorkiegyenlítésnél a munkaidők adottak és a struktúrát kell hozzá igazítani, itt viszont a struktúra adott és az időket kell megfelelően megválasztani.

Munkám során figyelembe veszem majd, hogy a hazai kereskedők nagy kötbérekkel dolgoznak a helyzetük miatt [11]. A feladatot megnehezíti még, hogy tejjel dolgoznak, és folyamatos a termelés, abból adódóan, hogy tehénészetek termelése folyamatos, amit fel kell dolgozni.

A teljes tejüzem modellezése túlmutat azon, amit meg tudnék valósítani, ezért az általam szimulált rendszer a tejföl és joghurt gyártási folyamatának egy részét valósítja meg jelenlegi formájában. Tejföl esetén az érlelést és a csomagolást, joghurt esetén csak a csomagolást. Azt, hogy miért ezeket a termékeket választottam a dolgozat későbbi részében megfelelően indoklom. További egyszerűsítése a problémának, hogy mind a két termékfajtánál csak egy kiszerezés méretet gyártok, illetve a termelésben lévő ingadozás minimális. Ezzel a könnyítéssel elérem, hogy a gépek meghibásodása csak a munkaórától függjön, és ne legyen rá hatással a terhelése a rendszernek, ezen egyszerűsítéssel más kutatások is gyakran élnek [12].

A jelenleg modellezett savanyú tejtermékek közül a joghurt gyártásának hat fontos lépése az alábbiak [13]. Első a nyers tej hőkezelése, vagy másnéven pasztörizálása. Általánosságban a tejtermékek készítésének ez egy fontos lépése, ami biztosítja a tej sterilizálását. A művelet fontosságát igazolja, hogy a gyártókat az Európai Unió külön rendeletben szólítja fel, így biztosítva az élelmiszer-biztonságot. A megfelelő hőkezelés jelen esetben $90 - 95\text{ }^{\circ}\text{C}$ -on zajlik, 5-10 perces hőn tartással, vagy $105 - 108\text{ }^{\circ}\text{C}$ -on 20 másodperces hőn tartással. Második lépés a homogenizálás: Amikor a tejben lévő nagy tejzsír golyócskákat „feldarabolják”, úgy, hogy kis résen keresztül 40 - 250 bar nyomáson átvezetik a $65\text{ }^{\circ}\text{C}$ -os pasztörözött tejet. Harmadik lépés az alvasztás, amikor a megfelelő hőmérsékletű pasztörözött tejhez hozzáadják a színtenyésztetet és azt megfelelően elkeverik, majd a

folyamatot irányítva, kialakítják a termék jellegét. Itt fontos megkülönböztetni a további lépések szempontjából, hogy a terméket tankban vagy pohárban, illetve papírdobozban oltják-e be. Jelen esetben a joghurtot tankban, a tejfölt pohárban alvasztják. Negyedik lépésként abban az esetben, ha tankban történik az alvasztás, a habarás. Ami során eléri, hogy a termék sima és csomómentes legyen miközben $10\text{ }^{\circ}\text{C}$ alá hűtik. Ötödik lépés az érlelés. Ez a lépés garantálja, hogy a termék íze megfelelő legyen. Ez sokszor spontán is megtörténik, mivel a terméket hűtve kell tárolni a kiszállításig. Hatodik és utolsó lépés a termék csomagolása. Amikor az egymásba rakott előre legyártott poharakat a töltőfejekhez továbbítják, ahol az adagolóhengereken keresztül a termék bekerül a pohárba, majd a gép záróhegesztő feje lezárja a terméket.

A tejföl gyártása csak annyival tér el, hogy a homogenizálás után a szintenyézzettel együtt egyből csomagolják a terméket, és egy külön érlelőben zajlik le az alvasztás és az érlelés, illetve ebben az esetben nincsen habarás.

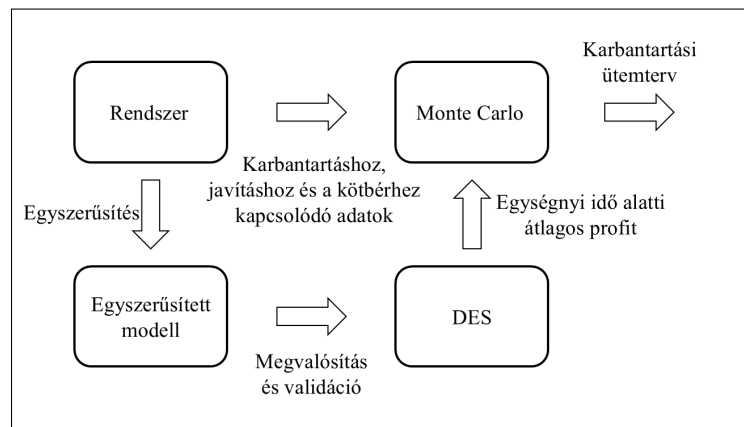
Mielőtt rátérnék a munkám során kidolgozott teljes szimulációs modellre és az ipar által inspirált probléma bemutatására, a következő fejezetben röviden ismertetem a kidolgozott módszertant, amely elősegíti a módszerem és a szimulációs modellem alkalmazhatóságát, akár más iparágakban is.

2. fejezet

Kidolgozott módszertan

Alapvetően a prediktív karbantartás esetében a karbantartást a várható meghibásodási időpont elé ütemezik valamivel [14]. Jelen munkámban egy olyan módszert kidolgozását írom le, ami ezt az időpuffert úgy határozza meg, hogy figyelembe veszi a karbantartás, a javítás és a kötbér költségét, illetve az egységnyi idő alatti átlagos nyereséget.

A karbantartási időpont meghatározásának első lépése, hogy a termelést olyan mértékben leegyszerűsítsem, hogy lehetőségem legyen azt modellezni. Következő lépésként elkészítem a rendszer hibamentes diszkrét eseményű szimulátorát (DES). Ezután validálok, hogy az egyszerűsített modellel követi a termelést. A reális szimulátorból pedig ki tudom nyerni az egységnyi idő alatti átlagos nyereséget. Ezzel és a meghibásodáshoz tartozó véletlent leíró számokkal, illetve a javítás és a karbantartás költségével, továbbá a kötbér adatokkal együtt lehetséges meghatározni egy konkrét hiba előtt az időpuffert. Ami egyenesen arányos a karbantartás, a javítás és a kötbér költségével, és fordítottan arányos az egységnyi idő alatti profittal. Végül Monte Carlo módszerrel figyelembe veszem a folyamat során megjelenő véletleneket, és így határozom meg, hogy milyen karbantartási ütemtervvvel érdemes dolgozni. A módszertant a 2.1 ábra szemlélteti.



2.1. ábra. A módszertant bemutató ábra.

3. fejezet

Az ipari folyamat és a fejlesztett modell bemutatása

Ebben a fejezetben leírom, hogyan sikerült a komplex tejüzemet úgy redukálnom, hogy a megfelelő paraméterekkel a Pannon Egyetem Ipar 5.0 laboratórium a modellje legyen neki. Pontosabban a kész termékek csomagolását, illetve, ha szükséges, a csomagolásban lévő érlelést tudom az egyetemi laborban modellezni. Erre azért van szükség, mert a cég szenzitív adatokra hivatkozva nem adott információkat, aminek a segítségével egy modellt létre tudnék hozni. Az adaptáció azért fontos, mert így tudom validálni a módszert és a szimulátort.

3.1. Az Ipar 5.0 laboratóriumban lévő folyamat bemutatása

Az Ipar 5.0 laboratórium azon része, amit modelleztem, különböző anyagmozgatási mód-szereket mutat meg a hallgatóknak, illetve két féle termék gyártását mutatja be, ahol a terméktípusok több közös gépen osztozkodnak. A legyártott kész termékek négy részből állnak, amik a következők.

Egy paletta, amire hat darab üveg fér. A későbbiekben a félreértés elkerülése végett az Ipar 5.0 laboratóriumban található műanyag tálcákat kartonnak fogom nevezni, mivel a tejüzemben lévő gyártásban is találhatóak tálcák. Viszont funkcióját tekintve a műanyag tálcák a tejüzemben lévő kartonoknak felelnek meg. Minden kartonban hat üveg található, és mindegyik üvegben azonos termék van, ami itt lehet víz vagy magok, illetve mindegyik üveget a gyártás során egy műanyag kupakkal zárja le a rendszer.

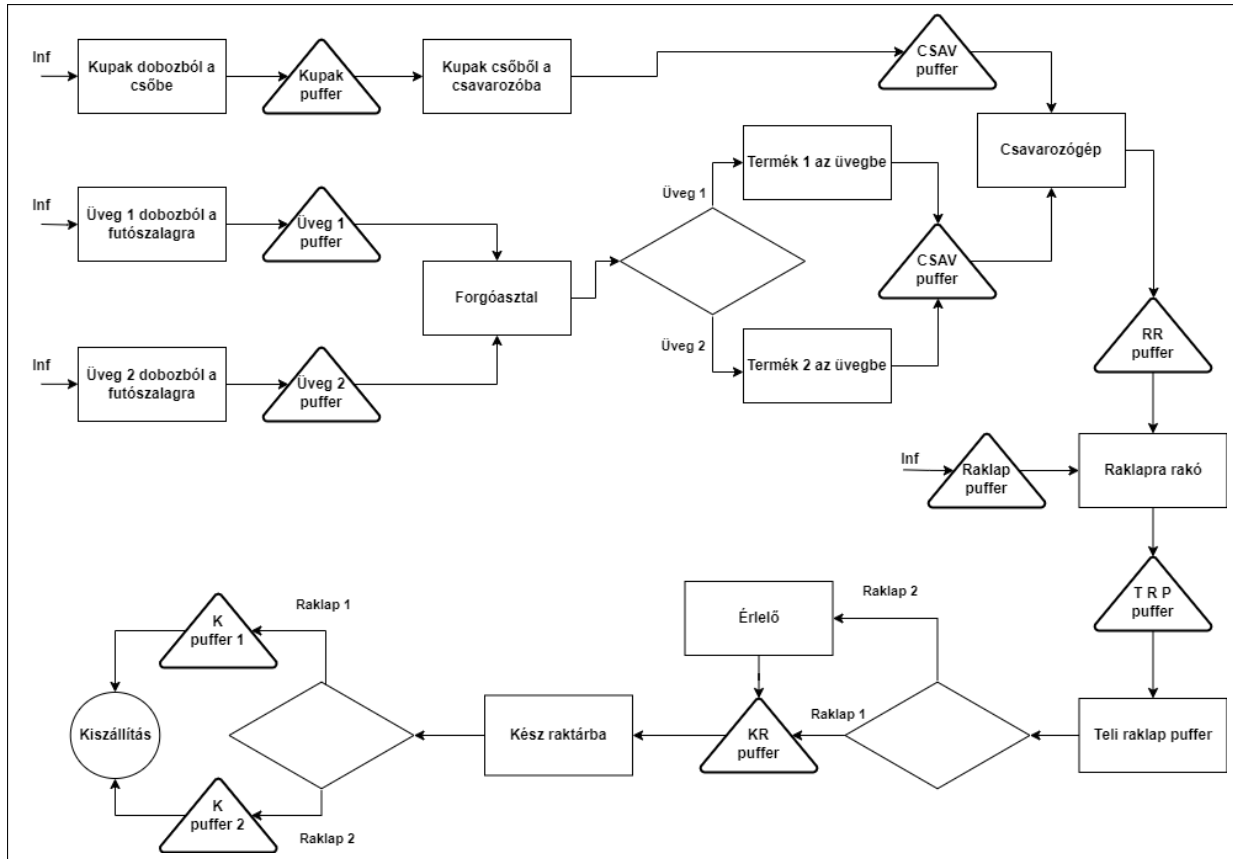
A műanyag kupakok mozgatása során az első lépés a gépsor mellett található dobozból a kupakok a gyártósoron található műanyag csőbe való belerakása, ami manuálisan történik. A későbbiekben erre a lépésre úgy fogok hivatkozni, hogy 'Kupak dobozból a csőbe'.

Az első kupakkal foglalkozó gép neve Kupak csőből a csavarozóba. Ennek a gépnek a puffere, egy műanyag cső, ami maximálisan 10 darab kupak tárolására alkalmas. Amikor át kell szállítani a gépnek egyet a pufferében lévő kupakok közül a következő gép pufferébe, akkor az egymásra halmozott kupakok közül kitolja a gép a legalsót, majd azt egy vákuumos megfogással rendelkező kar felszedi. Ezután a kar átfordul és lerakja a megfogott anyagot a Csavarozó gép kupak pufferébe.

A gyártási folyamat másik ága az üvegek termékekkel való megtöltésével foglalkozik. Ennél az ágnál is első lépésként a dobozból ki kell pakolni a két különböző típusú üvegeket a forgóasztal puffereként szolgáló egymás mellett lévő futószalagokra. Arra, hogy pontosan melyikre, az attól függ, hogy a termék 1 vagy 2-höz kapcsolódó üvegről van-e szó. Ennek a két lépésnek a neve az, hogy Üveg 1 dobozból a futószalagra, illetve Üveg 2 dobozból a futószalagra. A Forgóasztal mindkettő puffere 11 üveg tárolására alkalmas. Amikor egy üveget át akar rakni a következő géphez, akkor a futószalag egy rövid ideig elindul, ezzel elérve, hogy egy másik futószalagra kerüljön az üveg, ami betolja azt a forgóasztal meghatározott szabad nyílásába, majd onnét az asztal elfordulásával együtt az üveg is a megfelelő pozícióba kerül.

A Termék 1, illetve 2 betöltő gép gyakorlatilag egy-egy tartály. Az egyikben magok, a másikban víz van. Mindkettő esetén, amikor egy üres üveg alá érkezik, kinyílik a tartály alján található csap és megtölti anyaggal az üveget. Ezután a forgóasztal az üveget a Csavarozó géphez tartozó pozícióba viszi. A Csavarozó gép, ha üveg érkezik a megfelelő helyre, akkor az öt kapacitású kupak pufferéből a megfogó kar elvesz egyet, majd azt az üvegre rácsavarja. Ezután a forgóasztal tovább fordul, és a megtöltött, illetve lezárt üveg a Kartonra rakodó puffereként funkcionáló futószalagra kerül. A Kartonra rakodó, ha van legalább három lezárt üveg a huszonötös kapacitású pufferében és van legalább három szabad hely a tárolójában lévő kartonban, akkor három üveget egyszerre átrak a kartonba. Abban az esetben, ha a karton teljesen megtelt, elindul az alatta lévő futószalag és átviszi a tele lévő kartont a következő gép pufferébe. A Teli raklap puffer megakasztja a kartonokat, és abban az esetben, ha az érzékelő jele alapján az Érlelőt szimbolizáló polcrendszerre kell helyezni a kartont, és abban van szabad hely, akkor a robotkar megfogja azt, és a szabad helyre teszi. Maximálisan nyolc megtöltött karton tud várakozni arra, hogy megvizsgálják. Az Érlelőnek 16 szabad helye van. A futószalag és a helyek közötti minden mozgatóást ennél a gépnél egy robotkar végez el.

Végül a Kész raktárba gép hetes pufferéből átkerül a karton a Kiszállítás kettő puffere közül a megfelelőre. Az első puffer öt termék 1-gyel megtöltött karton befogadására képes, a másik ugyanennyi termék 2-vel megtöltött karton befogadására alkalmas. Ezekből a pufferekből manuálisan kell elvenni a kartonokat. Összefoglalva az Ipar 5.0. laboratóriumban lévő gépek együttműködését mutatja a 3.1 ábra.



3.1. ábra. Az Ipar 5.0. laboratórium folyamatábrája.

3.2. A szimulátor által egy hét alatt legyártandó termék mennyiség meghatározása

Az első és legfontosabb dolog, hogy meghatároztam pontosan, mely tejtermékek gyártását szeretném modellezni, mert az Ipar 5.0 laboratóriumban csak kettő félére van lehetőség. Ez összhangban van azzal, hogy ennél az üzemnél is csak két fő típust kell a gyártás szempontjából megkülönböztetni, ahol a terméket az érlelőbe kell rakni, ebből van három élelmiszerfajta, és van nyolc, aminél erre nincs szükség.

A meghatározásnál minden típusnál kiszámolom, hogy mekkora tömeget gyártottak le összesen a heti termelési adatok alapján (M). Ezt úgy tettem, hogy az egyes tejtermék típusok altípusainak a kiserelés tömegét (m_{gy}) megszoroztam a legyártott darabszámmal (n). Végül az egyes altípusok tömegét összegeztem. A konkrét képlet látható a 3.1 egyenletben, ahol a (N) az altípusok száma.

$$M = \sum_{i=0}^N m_{gy_i} \cdot n_i \quad (3.1)$$

Ezután kiválasztottam, hogy az egyes fő típusokon belüli típusok közül azt, ahol ez a tömeg a legnagyobb, és majd ezeknek megfelelően fogom az Ipar 5.0 labort paraméterezni.

Azért nem a darabszám alapján döntök, mert jelentős eltérés van az egyes típusokban található altípusok tömegei között. Például van 175 és 800 grammos tejföl is.

A számolás eredménye az lett, hogy azok közül, amit külön érlelőben érlelnek, a tejfölből gyártották a legtöbbet, és azok közül, ahol erre nincs szükség, a joghurtból készült a legtöbb. Fontos megjegyezni, hogy attól még, hogy külön érlelőbe nem kellett helyezni a joghurttermékeket, ezeknél is van érlelődés, de ez itt a tankban történik, nem pedig egy külön erre kialakított helyen.

Következő lépés ezután, hogy meghatároztam, hogy mekkora legyen egy termék tömege (m_l), amit a szimulátor legyárt. Ennek a meghatározására azért van szükség, mert jelen dolgozaton túlmutatna, ha több kiszereelési méretet és a hozzá kapcsolódó eltérő gépidőket is figyelembe kellene vennem. Én a továbbiakban a megrendelés szám alapján súlyozott tömegátlaggal számolok, amit úgy kapok, hogy a termékek tömegét (m_{gy}) összeszorzom a legyártott darabszámmal (n), majd ezeket összegezem, végül elosztom az összes megrendelés darabszámával. A konkrét képlet látható 3.2 egyenletnél, ahol a N az altípusok száma.

$$m_l = \frac{\sum_{i=0}^N m_{gy_i} \cdot n_i}{\sum_{i=0}^N n_i} \quad (3.2)$$

Ezzel a számítással azt kaptam, hogy egy joghurt tömege körülbelül 382 g, illetve egy tejföl tömege egész g-ra kerekítve 342 g.

A harmadik lépésben meghatároztam, hogy az Ipar 5.0 laborban található palettára hány kartonnak feleljen meg az üzemben. Az átszámításhoz a két rendszerben található érlelőket vettem alapul. Ehhez meg kellett határoznom, hogy az üzemben átlagosan az ottani érlelési kapacitás hány százalékát használták a tejfölök érlelésére (η). Ezt úgy tettem meg, hogy legyártott összes tejföl tömegét ($M_{tej\text{f}\ddot{o}l}$) megszoroztam az érleléshez szükséges idővel ($t_{tej\text{f}\ddot{o}l}$), majd elosztottam az összes érlelőben savanyított termék tömegével ($M_{tej\text{f}\ddot{o}l}$, M_x és M_y) a hozzá kapcsolódó érlelési idők ($t_{tej\text{f}\ddot{o}l}$, t_x és t_y) szorzatainak összegével. A konkrét képlet látható 3.3 egyenletnél.

$$\eta = \frac{M_{tej\text{f}\ddot{o}l} \cdot t_{tej\text{f}\ddot{o}l}}{M_{tej\text{f}\ddot{o}l} \cdot t_{tej\text{f}\ddot{o}l} + M_x \cdot t_x + M_y \cdot t_y} \quad (3.3)$$

Ezzel az arányszámmal megszoroztam az ottani érlelő befogadó képességét, majd a szorzatot elosztottam az Ipar 5.0 laboratóriumban található polcrendszer szabad helyeinek a számával. A számolás eredménye az lett, hogy az Ipar 5.0 laboratóriumban található paletta 3,45 kartonnak felel meg a valóságban.

A cégből kapott adatok alapján megállapítottam, hogy mind a tejfölnél, mind a joghurtnál ekkora tömegű kiszereelés esetén 960 termék kerül egy kartonba. Azt, hogy pontosan hogyan számoltam ki, hogy hány kartont kell leszimulálnom (O) az alábbi 3.4 képlet

mutatja.

$$O = \frac{M}{960 \cdot 3,45 \cdot m_l} \quad (3.4)$$

Miután a joghurtra és a tejföldre vonatkozó adatokat behelyettesítettem, az eredmény az lett, hogy átlagosan 75 joghurtot és 66 tejfölt kell egy hét alatt legyártani.

3.3. Gépidők és ütemezés meghatározása

A gépidők meghatározásának első lépéseként megkerestem, hogy heti hány nap és napi hány órában megy a termelés. Ehhez álláshirdetéseket kerestem az interneten. A kutatás eredménye az lett, hogy a hét minden napján, három műszakban megy a termelés, azaz folyamatos a gyártás, ami teljesen logikus, mivel a termeléshez kapcsolódó biológiai folyamatokat például a tehenek tejtermelését, illetve a termékek érlelődését nem lehet leállítani.

Második lépésként meghatároztam a takt time-ot (t_{takt}). Ehhez az egy hét idejét másodpercben elosztottam az egy hét alatt legyártandó joghurt és tejföl kartonok számával. A számolás menetét mutatja a 3.5 egyenlet.

$$t_{takt} = \frac{7 \cdot 24 \cdot 60 \cdot 60}{66 + 75} = \frac{604800}{141} = 4289,36... s \quad (3.5)$$

Harmadik lépésként a gépek konkrét munkaidejét meghatároztam. Az egyes gépeknél a számolás menete a következő volt.

Kupak csőből a csavározóba: Itt a gép munkaideje a takt time hatoda, mivel egy kartonban hat üveg van és mindegyikre kell egy kupak. Fontos viszont, hogy mivel diszkrét idejű szimulátorral fogok dolgozni, ezért a kapott értéket egészre kell kerekítenem. A konkrét számolást mutatja a 3.6 egyenlet.

$$t_{Kupak\ csőből\ a\ csavározóba} = \frac{t_{takt}}{6} = 714,89.. \approx 715 s \quad (3.6)$$

Forgóasztal, Termék 1 az üvegbe és Termék 2 az üvegbe: Ahhoz, hogy folyamatos legyen a termelés, fontos, hogy amíg a kupak csőből a csavározóba gép átpakol egy kupakot, addig a forgóasztal és a termék betöltő gépek közül az egyik végezzen a munkával. Ez a kritériumot írja le a 3.7 és 3.8 egyenletek.

$$t_{Kupak\ csőből\ a\ csavározóba} = t_{Forgóasztal} + t_{Termék\ 1\ az\ üvegbe} \quad (3.7)$$

$$t_{Kupak\ csőből\ a\ csavározóba} = t_{Forgóasztal} + t_{Termék\ 2\ az\ üvegbe} \quad (3.8)$$

A másik fontos feltétele, hogy ne akadjon meg a termelés, hogy lehetőleg egyenlő ideig

dolgozzon mind a három gép. Ezt írja le a 3.9 egyenlet.

$$t_{Forgóasztal} = t_{Termék\ 1\ az\ üvegbe} = t_{Termék\ 2\ az\ üvegbe} \quad (3.9)$$

A 3.7, 3.8 és 3.9 egyenletek által leírt több ismeretlenes egyenlet rendszer megoldása figyelembe véve, hogy mindegyik érték egész szám kell, hogy legyen, a következő lett.

$$t_{Forgóasztal} = 357\ s, \ t_{Termék\ 1\ az\ üvegbe} = 358\ s \text{ és } t_{Termék\ 2\ az\ üvegbe} = 358\ s$$

Csavarozógép: A gyártás folyamatosságához itt csak az kell, hogy pontosan annyi idő alatt csavarja rá a kupakot az üvegre, amíg a Kupak csőből a csavarozóba gép oda nem adja neki a következő csavart. Matematikailag ezt írja a 3.10 egyenlet.

$$t_{Csavarozógép} = t_{Kupak\ csőből\ a\ csavarozóba} = 715\ s \quad (3.10)$$

Raklapra rakó: Ez a gép az, ami a teletöltött és lezárt üvegeket a kartonba rakja. Egyszerre három üveg átszállítására képes és minden kartonba hat üveg kell, hogy kerüljön. Ebből adódik, hogy a munkaidejének a takt time felének kell lennie. A munkaidejét konkrétan a 3.11 egyenlet írja le.

$$t_{Raklapra\ rakó} = \frac{t_{takt}}{2} = 2144,68... \approx 2145\ s \quad (3.11)$$

Teli raklap puffer: Ez a gép egy kartonnal csak egyszer foglalkozik. Ebből adódóan a munkaideje a takt time egészre kerekített értékével egyenlő. A konkrét értékét pedig a 3.12 egyenlet írja le.

$$t_{Teli\ raklap\ puffer} = t_{takt} = 4289,36... \approx 4289\ s \quad (3.12)$$

Érlelő: A termelési adatok alapján minden tejfől terméket 18 órán át érlelnek. Azaz az érlelő "munkaideje" 18 óra.

Kész raktárba: Ennek a gépnek a puffere kettő helyről kaphatja az anyagot (Érlelő és Teli raklap puffer). Azért, hogy ne teljen meg a puffere a munkaideje fele a Teli raklap puffer gépnek. A pontos értékét írja le a 3.13 egyenlet.

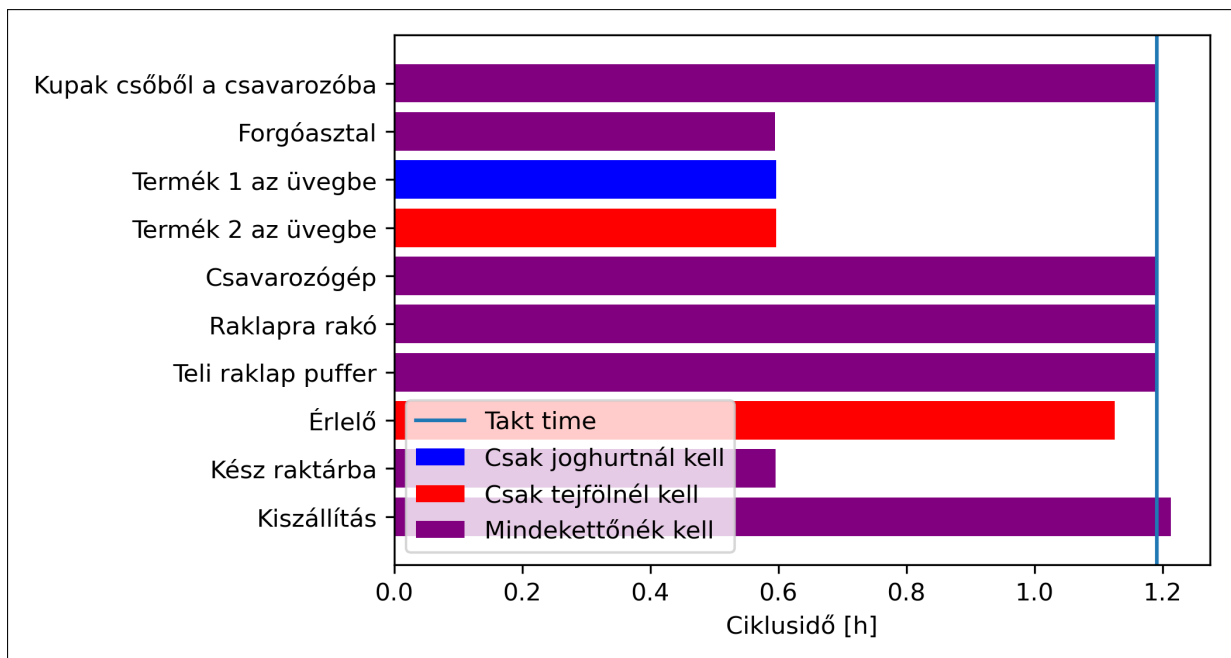
$$t_{Kész\ raktárba} = \frac{t_{Teli\ raklap\ puffer}}{2} = \frac{4289}{2} \approx 2145\ s \quad (3.13)$$

Kiszállítás: Azért, hogy biztosan ne okozzon torlódást az előző gép (Kész raktárba) nem kerekített munkaidejét megszorozom a Kiszállítás gép legkisebb pufférének a kapacitásával. Mivel a kettő puffer mérete megegyezik, ezért lényegtelen, melyikkel szorzom meg. A

pontos érték számolását írja le a 3.14 egyenlet.

$$t_{Kiszállítás} = 5 \cdot t_{Kész raktárba} = 10723 s \quad (3.14)$$

A ciklusidők a Raklapra rakó gép előtt a munkaidő hatszorosai, mivel ezeknek a gépeknek hatszor kell elvégezniük a munkát, hogy egy kartonhoz kellő anyaggal végezzenek. A Raklapra rakó gép ciklusideje a munkaidejének kétszerese, mivel két részletben rakja bele a kartonba a megtöltött üvegeket. Az érlelő esetén a ciklusidőt úgy kapom meg, hogy a munkaidejét elosztom a benne lévő szabad helyek számával. Mivel ez 16 a ciklus ideje ennek a gépnek 4050 s. A kiszállításciklusideje egyenlő a munkaidejének és az átlagosan elszállított termékek számának a hányadosával, azaz körülbelül 4367 s-mal. A többi gép esetén a ciklusidő egyenlő a munkaidővel. A takt time-mal együtt az összes gép ciklusideje látható 3.2 ábrán.



3.2. ábra. A folyamat Yamazumi chartja

Az ütemezés leírása. Azért, hogy az érlelő kihasználtsága maximális legyen a termelés elején elkezdek annyi tejfölt legyártani, ami az érlelőbe befér, azaz tizenhatot. Ezután annyi joghurt gyártását indítom el, ami egyenlő az érlelési idő és a tack time hányadosának egészre felkerekített értékével, ami szintén 16. Ezt a kettes ciklust minden héten háromszor megismétlem, majd elindítok annyi tejfölt, amennyi még kell, hogy meglegyen a 66, azaz kettőt. Végül annyi joghurtot, amennyi kell, hogy meglegyen a 75, azaz tizenegyet. Természetesen a torlódások elkerülése végett a betöltés úgy történik, hogy egyszerre egy kartonhoz szükséges anyagot rakok be, majd, ha azt a rendszer feldolgozta, rakom be a következőt.

4. fejezet

Fejlesztett szimulációs modell és validációja

A fejezetben bemutatom az általam fejlesztett SimPy (Python) diszkrét eseményű szimulátort, ami az eddigiekben leírt folyamatot szimulálja. A fejezet végén pedig validálom a munkámat.

4.1. A SimPy-ban használt objektumok

A SimPy egy Pythonhoz tartozó programcsomag, ami folyamat alapú diszkrét események szimulációjához ad keretrendszert. A teljes szimulációt a `simpy.Environment()` paranccsal fogom össze, amit a szimulátorhoz kapcsolódó összes objektumban és felhasználói függvényben beleírtam.

A program objektumai, a gépek és a termékek, amiknek több saját változójuk van. A gépekhez kapcsolódó információkat a program egy külön, ezekkel az adatokkal rendelkező Excel fájlból olvas ki.

A termék objektumok első három változóját `bool` típusúként kezelem. Ezek megadják, hogy az adott termék rendelkezik-e kupakkal, üveggel és hogy kartonban van-e. A következő három változója egész szám. Az első megadja, hogy hány egység tejföl vagy joghurt kell az adott termékhez. A második, hogy jelenleg hány egység van benne. Ennek az értéke kezdetben nulla. A harmadik változó pedig definiálja, hogy az adott termék kupak, üveg 1 vagy üveg 2-e. A beletöltött anyagot nem kezelem külön objektumként, mert a munkám jelenlegi formájában csak a csomagoláshoz kapcsolódó folyamatokkal foglalkozik. Végül egy külön változóban tárolom, hogy az ütemezés szerint mikor kezdték el annak a kartonnak a gyártását, amihez ez a rész tartozik.

A gép objektum elemeit két csoportra lehet bontani. Az első a futtatáshoz szükséges elemek. A második a szimulációhoz kapcsolódó adatokat gyűjtő listák.

A szimuláláshoz szükséges elemek közül a kettő legfontosabb, amit a SimPy program-csomag ad be, a gép kettő puffere és az erőforrása. A legtöbb gép az egyik puffert soha nem használja, de a program átláthatósága szempontjából érdemes volt csak egyfajta gép osztályt létrehozni. A többi csak szimpla változó. Ide tartozik a gép neve, munkaideje másodpercben, melyik a következő gép, amibe át kell majd töltenie a kész anyagot, illetve egy változó tárolja azt az objektumot, amivel éppen dolgozik a gép.

Az adatgyűjtő listákat megkülönböztettem aszerint, hogy az adat melyik termék típushoz tartozik. Azaz vannak, amik a joghurt és vannak, amik a tejföl gyártáshoz kapcsolódnak. A folyamathoz kapcsolódó adatok a következők. Mikortól és mennyi ideig várakozott alapanyagra, dolgozott, esetleg arra várt, hogy szabaduljon fel hely a következő gép puffereben. Az alapanyagra várakozásnál alapvetően a következő megérkező anyag dönti el, hogy pontosan mire várakozott. A termékekhez kapcsolódóan gyűjtöm, hogy mikor mennyi anyag van a gép puffereben, illetve hogyan alakul az elkészített termékek száma az időben. A gép és termék objektumokat átadom a megfelelő felhasználói függvényeknek, amikbe leprogramoztam az egyes gépek működését. Ezután már csak el kell indítanom a szimulációt, megadva, hogy hány időegységig (másodperc) szimulálja le a működést.

4.2. A rendszer bemeneteinek és az ütemezésnek a megvalósítása

Azt, hogy pontosan mikor, melyik gépbe és mennyi kartonhoz elegendő alapanyag kerüljön, egy Excel fájl vezérli. Viszont, mivel akár csak egy hét leszimulálásához is 141 sornyi adatot kellene rögzítenem, írtam egy kódot, ami ezt a fájlt létrehozza.

Ennek a programnak az első szakaszában vannak azok az adatok, amik szükségesek a fájl megírásához. Azaz, hogy pontosan hány heti működést legyen képes megfelelően ütemezni a kész fájl, hogy kettő karton betöltése között mennyi idő teljen el, ha egy heti anyag betöltésével végzett mennyit várjon, amíg elkezdi a következő hét betöltését, és hogy pontosan melyik típusú termékből összesen hány kartonnyi anyagot töltsön be egymás után. Ez egy listában van, aminek a páros indexű elemei a tejfölök, a páratlan számúak a joghurt kartonok gyártását indítják el.

A második szakaszban megírja a program a fájlt. Első lépésként a fejlécébe beírta a megfelelő szövegeket. Az Excel következő sorába beleírja, hogy mikor kezdje el az első tejfölt tartalmazó karon gyártását. A kezdési időpontot eltolva megismétli ezt annyiszor, ahányszor a lista első eleme ezt meghatározza. Ezután ugrik a lista következő elemre, ami a joghurt kartonok gyártását indítja el. Ezt követően ismét tejföl kartonok gyártását előíró sorokat ír a program az Excelbe. Ha végzett a heti anyagok bepakolását ütemező

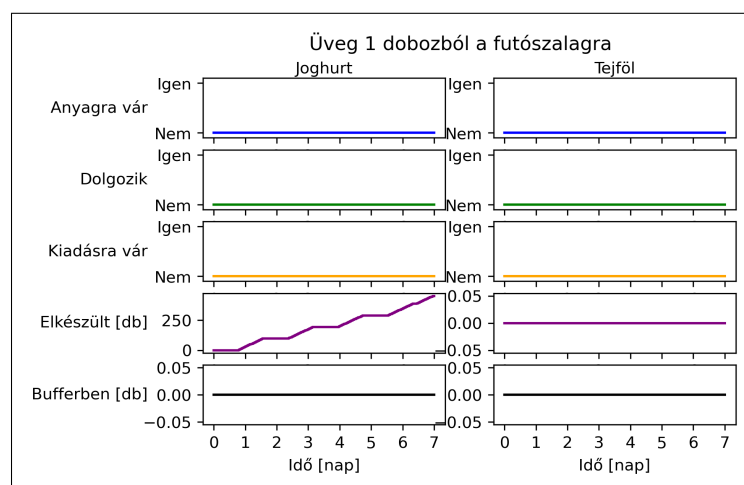
sorokkal, a bepakolási időt úgy módosítja, hogy pontosan abban az órában és percben kezdődjön el a következő heti anyag betöltése, mint az előző héten.

Az így elkészített fájlt a szimulátorban három gép olvassa be, a Kupak dobozból a csőbe, az Üveg 1 dobozból a futószalagra és az Üveg 2 dobozból a futószalagra. Mivel ezek közös felhasználó függvénnyel rendelkeznek, a kód az első soraiban név alapján azonosítja, hogy pontosan melyik gép működését szimulálja, és ennek megfelelően állítja be, hogy hova és milyen típusú termék objektumokat pakoljon.

Ezután elindul egy while ciklus, ami egészen addig fut, amíg az összes sort be nem olvassa, vagy véget nem ér a szimuláció. A ciklus legelején a program eldönti, hogy az adott időpillanatban van-e olyan termék, aminek a gyártását el kell-e indítani, ha nem, vár egy másodpercet, és újra megnézi. Ha igen, akkor hat paraméterezett objektumot bepakol a megfelelő pufferbe. Felírja, hogy mikor és összesen hány terméket pakolt be, majd lépteti azt, hogy melyik sort olvassa be legközelebb eggyel. Abban az esetben, ha az összes sort beolvasta, de még nem futott le a teljes szimuláció, akkor eltolja a saját belső idejét a szimulált idő végéig.

A gépek közötti anyagmozgatást legtöbbször egy külön függvény végzi ennek a neve az, hogy **push_function**. Ha van a megadott következő gép pufferében szabad hely, akkor rögzíti, hogy a megjelent plusz egy anyag, illetve, ha volt várakozási idő, azt is felírja ez a program. Ha nincsen, vár egy másodpercet, és újra megnézi, van-e hely a következő gép pufferében. Természetesen, ha a szimuláció legvégén egy termék arra vár, hogy bekerüljön a következő gép pufferébe, akkor a várakozási időt feljegyzi a program.

Fontos, hogy ezeknél a gépeknél nem értelmezem a munkát, mivel egy Excel fájl mondja meg, mikor rakjon anyagot a program a következő gép pufferébe. Valóságban ez megfeleltethető annak, hogy egy munkás betölti akkor az anyagot. Az egyik betöltő gép esetén az egy hetes működés alatt gyűjtött adatok vizualizációja látható a 4.1 ábrán.



4.1. ábra. Példa a bemenetet megvalósító gépeknél a gyűjtött adatok vizualizációjára.

4.3. A csak szállítás funkcióval rendelkező gépek

Ide két gép tartozik a kupak csőből a csavarozóba és a kész raktárba. Ezeknél a program csak az alábbi három lépést ismétli folyamatosan a szimuláció hossza alatt. A következő fejezetben lévő gépeknél is ugyanaz a hármas ciklus jelenik meg, de ott még gépenként egy eltérő extra lépés is van a folyamatban. Viszont az alap lépések a következők. Az első, hogy megvárja a gép, amíg kerül anyag a pufferébe. Ezután jön a második lépés, hogy a beérkezett anyaggal elkezd dolgozni, olyan hosszan, ahogyan a gép paraméterei között szerepel. Végül a harmadik lépésben áthelyezi az anyagot a következő gép pufferébe. Ha ez nem lehetséges, mert az tele van, akkor kivárja azt az időt, amíg felszabadul egy hely. Eközben persze a program gyűjti, hogy az egyes lépések mikor kezdődtek és mennyi időn át tartottak, illetve a pufferekben lévő anyagmennyiség és az összes elkészített termék hogyan változik az időben.

A működés hasonlósága miatt a mostani és a következő fejezetben lévő gépek nagy részének a munkáját is egy közös felhasználói függvény a **working_normal** vezérli. A szimuláció elindítása után ez a függvény belelép egy végtelen ciklusba. Ez biztosítja, hogy majd csak akkor álljon le a függvény, ha a megadott szimulációs idő modellezése megtörtént. Ezután egy több lépcsős if struktúrával azonosítja, hogy pontosan melyik gép működését kell követnie a programnak. Azért nem switch case struktúrával dolgoztam, mert a Pythonba nincsen alapvetően benne ez a parancs.

A következő lépésben **working_normal** meghívja az alapanyagra várakozást és a munkát megvalósító **resource_normal** függvényt. Ennek át kell adni, hogy pontosan a gép melyik pufferéből akarja kivenni az anyagot, és hogy szükséges-e a munka végzés, vagy egyből kiléphet-e a függvényből. Ez a következő fejezetben lévő gépeknél lesz ez fontos.

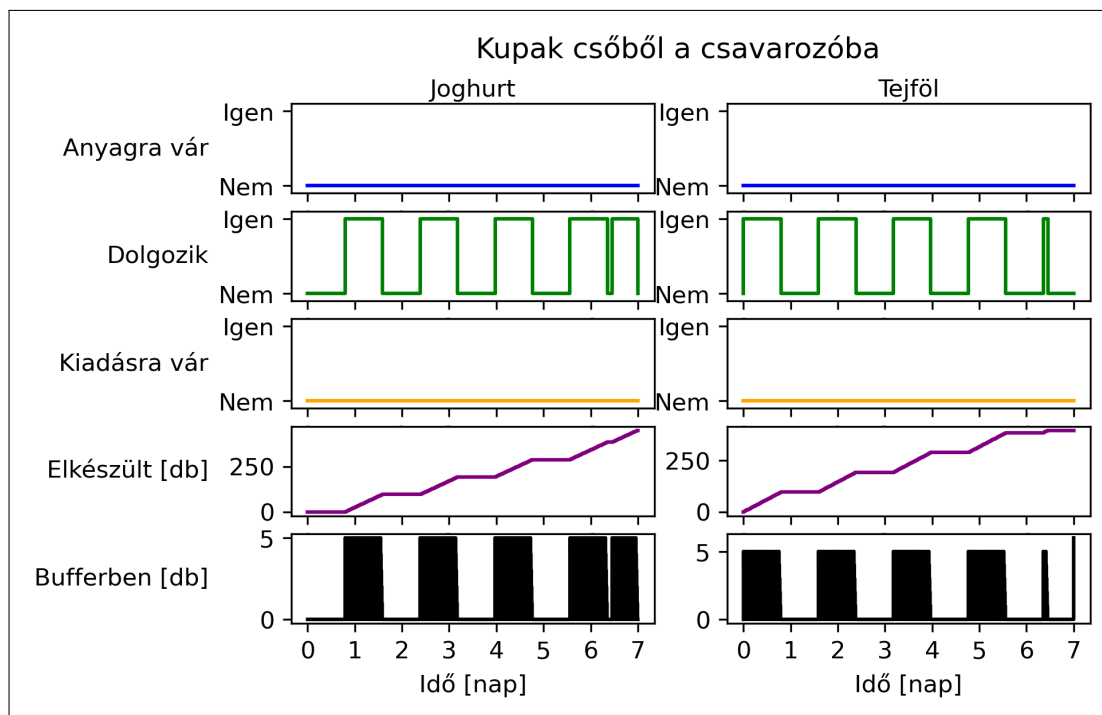
A **resource_normal** függvény elindításakor belelép egy while ciklusba, ami egészen addig fut, amíg üres a bemenetként megadott puffere a gépnek. A ciklus első lépésként ellenőrzi, hogy a szimuláció végére ért-e. Ha nem, akkor vár egy másodpercet, és újra megnézi, van-e anyag a megadott pufferben. Abban az esetben, ha elérté várakozás közben a szimuláció a végét, akkor a maradék időt az eddigi várakozás arányában osztja szét. Ha nem várakozott a gép, akkor az elkészített termékek arányában osztja szét a maradék időt. Végül, ha egy terméket se gyártott le (nagyon rövid ideig futó szimuláció esetén) akkor a várakozási időt megfelezi, és az egyiket a joghurthoz, a másikat a tejfölhöz adja hozzá.

Ezután bekéri az anyagot a pufferből és rögzíti, hogy egyel kevesebb tejföl vagy joghurt van a pufferében, illetve a várakozáshoz kapcsolódó adatokat.

Végül abban az esetben, ha szükséges a munkavégzés, akkor annyi ideig vár a szimulátor, amennyi idő alatt a munkát befejezné a gép. Itt a munka kezdetét és hogy

ennyi ideig tartott terméktípusonként gyűjti a program, majd visszalép a **working_normal** függvénybe, ami már csak annyit csinál ezeknél a gépeknél, hogy felparaméterezi a **push_function** függvényt. Ezen függvény működését részletesen leírtam korábban a 15 oldalon, amit, ha gépről olvassa ezt az olvasó **ide kattintva** egyből odaugorhat.

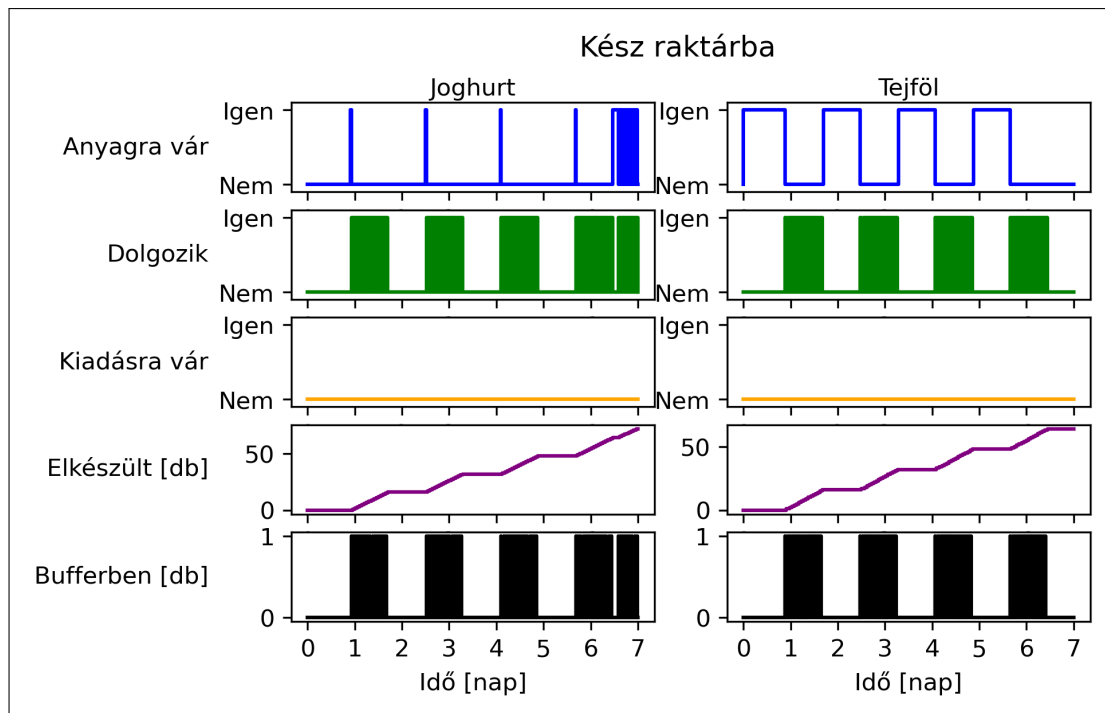
Egy hetes működés modellezése után a működéshez kapcsolódó fontos adatok vizualizált formája látható a 4.2 ábrán a kupak csőből a csavározóba gép esetén.



4.2. ábra. A kupak csőből a csavározóba gép munkáját rögzítő adatok vizualizációja.

Az ábrához kapcsolódó fontos információ, hogy ha folyamatosan tudja a munkáját végezni a gép, akkor a dolgozik függvény értéke nem változik, annak ellenére, hogy többször befejezte a munkát. Ez jól látható, onnét, hogy amikor az elkészült függvény értéke nő, a dolgozik függvény értéke folyamatosan az, hogy igen.

A kész raktárba gép ábrája annyival tér el, hogy az érlelő miatt egymást felváltva érkeznek a joghurt és a tejföl kartonok, ebből adódóan nincsen olyan hosszú szakasz, aminél folyamatosan munka állapotban van a gép. A működéshez kapcsolódó adatok feldolgozott formában láthatóak a 4.3 ábrán.



4.3. ábra. A kész raktárba gép munkáját rögzítő adatok vizualizációja.

4.4. Egyszerű feladattal rendelkező gépek

Ebben a fejezet azt írom le, hogy az egyes gépeknél megjelenő sajátosságokat hogyan valósítja meg a program. A fő működést már a 4.3 fejezetben leírtam, amit, ha gépről olvassa ezt az olvasó, akkor **ide kattintva** egyből odaugorhat. Az egyszerű feladattal rendelkező gépek csoportba az alábbiak tartoznak:

- Termék 1 az üvegbe
- Termék 2 az üvegbe
- Csavározógép
- Raklapra rakó
- Teli raklap puffer

A Termék 1 az üvegbe és a Termék 2 az üvegbe gépek működése két ponton több mint a csak szállító funkcióval rendelkező gépeké. Az első, hogy mielőtt továbbadná az üveget a következő gépnek, a benne lévő anyagmennyiséget jelképező változót egyenlővé teszi a szükséges anyagmennyiséget rögzítő változóval. Ezt azért tehetem meg, mert jelenleg végtelenként kezelem a kész anyag mennyiségét. A második, hogy mielőtt továbbadná a terméket a következő gépnek, megszámolja, hogy a másik termék betöltő gép egész számú karton betöltésével végzett-e. Ha nem, akkor megvárja, hogy a másik gép befejezze a

munkát. Ezt a plusz időt torlódásként rögzíti a program. Az Ipar 5.0 laboratóriumba, mivel egy forgóasztal végzi a mozgatást, ilyen jellegű probléma nem merülne fel, de a tejüzem szempontjából realisabb, hogy egyszerre tudjon történni a joghurtok és a tejfölök betöltése a poharakba. Ha ez a lépés nem lenne benne, akkor nem lenne biztosított az, hogy egy kartonba csak egyféle termék kerüljön.

A termék betöltő gépek működése közben keletkező adatok bemutatására szolgáló ábrák a forgóasztal működését bemutató 4.4 ábrára hasonlítanak leginkább, mivel ezeknél a gépeknél a struktúrából adódóan nem lehetett ennél jobban kiegyenlíteni a működést. Természetesen ezeknél az ábráknál csak az egyik, vagy másik oszlopban vannak információval szolgáló adatok. Ha gépről olvassa ezt, akkor **ide kattintva** egyből az ábrához ugorhat a tisztelt olvasó.

A csavarozógép működése annyiban tér el a normálistól, hogy ez a gép mind a kettő pufferét használja. Az egyikben a kupakok, a másikban a termékekkel megtöltött üvegek találhatóak. Először a **resource_normal** függvény segítségével, munkavégzés nélkül a kupak pufferéből vesz ki anyagot. Majd miután elmentette a kupak változó értékét, ugyanezzel a függvénnyel egy termékkel megtöltött üveget vesz ki a másik pufferéből. Ekkor már a gép munkát is végez, ezért ezt a lépést már nem tekinti a program pillanat szerűnek. Ennél a gépnél pontosan el lehetett végezni a sorkiegyenlítést, ezért az ábrája nagyon hasonlít a 4.1 ábrára, amit, ha gépről olvassa ezt a szöveget, akkor **ide kattintva** egyből odaugorhat.

A raklapra rakó gépet külön felhasználói függvény a **carton_loader** vezérli. Viszont a gépre jellemző sajátosságokat figyelembe véve ugyanazt a munkát végzi el a szimulátor, mint a **working_normal** és a **resource_normal** függvények, csak azért választottam szét, mert túlságosan rontotta volna e függvények átláthatóságát, ha a sajátosságokat ezekbe írom.

Három fontos dolog van, amiben ez a gép eltér a csak szállítás funkcióval rendelkező gépektől. Az első, hogy megvárja, hogy legalább három anyag legyen a raktárában és csak ezután pakolja ki a pufferéből az anyagot. A várakozási időt az első anyag átszállítása előtt, a munkát az utolsó átszállításakor jegyzi fel a program. A második, hogy minden második munka után a termék azon változóját, ami azt jelzi, hogy kartonban van-e, egyre állítja. Harmadik, hogy csak minden második munka után hívja meg a **push_function**-t és adja tovább a következő gépnek az anyagot.

Mivel ennél a gépnél is a sorkiegyenlítést pontosan meg lehetett valósítani, ezért az adatokat vizualizáló ábra nagyon hasonlít a 4.1 ábrára, amit, ha gépről olvassa ezt a szöveget, akkor **ide kattintva** egyből odaugorhat.

A teli raklap puffer esetén az alapvető működés csak annyiban tér el, hogy attól függően, hogy tejföllel vagy joghurttal dolgozik-e a gép, változik, hogy az érlelőbe vagy a kész

raktárba küldi a terméket.

Ennél a gépnél a sorkiegyenlítés miatt minden munka előtt vár pár másodpercet, ebből adódóan nincsen a munka fázisban olyan hosszan. Ezért a gyűjtött adatokat bemutató ábra leginkább a 4.3 ábra utolsó fél napjára hasonlít. Ha gépről olvassa ezt, akkor **ide kattintva** odaugorhat.

4.5. Forgóasztal

A forgóasztal működésében több eltérés is van, ami megkülönbözteti a csak szállítás funkcióval rendelkező gépektől. Viszont a fő működése hasonló, mint azoknak. A különbségek miatt külön felhasználói függvények (**working_turntable** és **resource_turntable**) bonyolítják le a működést. Az alap működést már a 4.3 fejezetben leírtam, amit, ha gépről olvassa ezt az olvasó, akkor **ide kattintva** egyből odaugorhat.

A legfontosabb különbség, hogy ez a gép is használja mind a kettő pufferét, de nincsen egy előre meghatározott sorrend, hogy melyik pufferéből veszi ki az anyagot. Hanem egy saját logika mentén dönt arról, hogy melyik pufferéből dolgozzon.

A logika első fontos eleme, hogy forrást csak akkor változtathatja a gép, ha az elszállított üveg 1 és üveg 2 száma külön-külön osztható hattal. Ezzel hozzájárulva ahhoz, hogy a raklapra rakó csak ugyanolyan termékeket rakjon a kartonokba. Magyarul a forgóasztal elszállít egy kartonnyi üveget, megnézi, hogy a következő kartonnyi milyen legyen majd.

Abban az esetben, ha meg kell határoznia, hogy melyik pufferéből dolgozzon. A program megnézi, melyikben van anyag egyáltalán. Ha csak az egyikben, akkor értelemszerűen abból kezdi el tovább szállítani az anyagot. Ha egyikben sincsen, akkor vár egy másodpercet, és újra megnézi, jelent-e meg valamelyikben anyag. Ha mindkettőben van, akkor az alapján dönt, melyiket rakták be korábban a pufferbe. Egyenlőség esetén az Üveg 2-t tartalmazó pufferből kezd el dolgozni a forgóasztal.

A jobb áttekinthetőség érdekében a döntés igazság táblája látható a 4.1 táblázatban. Az átláthatóság érdekében külön felsorolom az egyes logikai döntéseket, amik a következők:

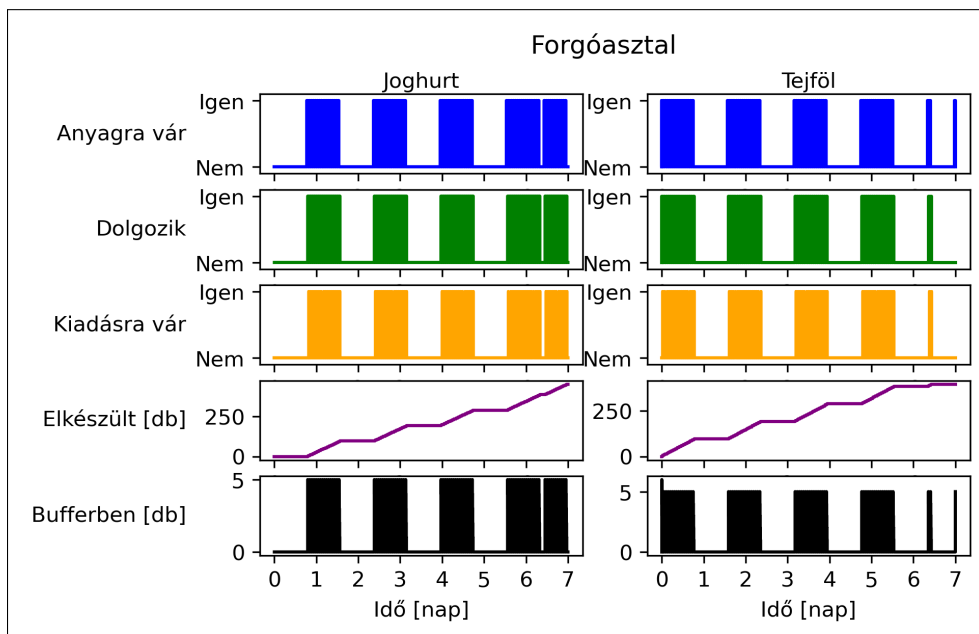
- a: Osztható-e hattal az átszállított üveg 1-ek és az üveg 2-ők száma.
- b: Üveg 1 pufferben van üveg.
- c: Üveg 2 pufferben van üveg.
- d: Az Üveg 1 puffer régebb óta nem üres

a	b	c	d	Kimenet
0	X	X	X	Előző forrásból dolgozik tovább
1	1	0	X	Üveg 1 pufferból kezd el dolgozni
1	0	1	X	Üveg 2 pufferból kezd el dolgozni
1	1	1	1	Üveg 1 pufferból kezd el dolgozni
1	1	1	0	Üveg 2 pufferból kezd el dolgozni
1	0	0	X	Vár 1 s-ot

4.1. táblázat. A puffer kiválasztás igazság táblája

Ezután a kiválasztott pufferból kivesz egy anyagot a program. Abban az esetben, ha nincsen benne anyag, akkor minden esetben a forrástól függően dönti el, hogy tejföldre vagy joghurtra várakozott-e. Következő lépésben elvégzi a munkát és feljegyzi a munkavégzéshez kapcsolódó adatokat. Végül a termék típusától függően a **push_function**-el a termék 1 az üvegbe vagy a termék 2 az üvegbe géphez szállítja az anyagot.

A forgóasztalnál folyamatosan megjelennek torlódások és anyagra várakozások, mivel a forgóasztal nem várja meg, hogy a termék betöltő gépek végezzenek a munkával. Az adatok vizualizált formája látható a 4.4 ábrán.



4.4. ábra. A forgóasztal munkáját leíró adatok vizualizációja.

4.6. Érlelő

Az érlelés szimulálásakor minden egyes szabad helyet külön modelleztem. Ezek a kis objektumok mindegyike csak szállítással foglalkozó gépnek feleltethetők meg, de a program áttekinthetőségének érdekében külön felhasználói függvényt definiáltam erre a célra a **refrigerat_work**-ot. A csak szállítással foglalkozó gép működését a 4.3 fejezetben leírtam, amit, ha ezt gépről olvassa a tisztelt olvasó **ide kattintva** egyből odaugorhat.

A lényeges eltérés a **refrigerat_work**-kel kapcsolatban csupán annyi, hogy a munkához és torlódáshoz kapcsolódó adatokat nem csak a kis objektumok saját változóikban rögzítik, hanem egy fő irányító objektumnak is átadják.

Ezt a fő objektumot a **refrigerating** függvény vezérli, és azt a feladatot valósítja meg, hogy ha anyag érkezik a pufferébe, akkor keres egy éppen nem dolgozó kis objektumot, aminek átadja az anyagot.

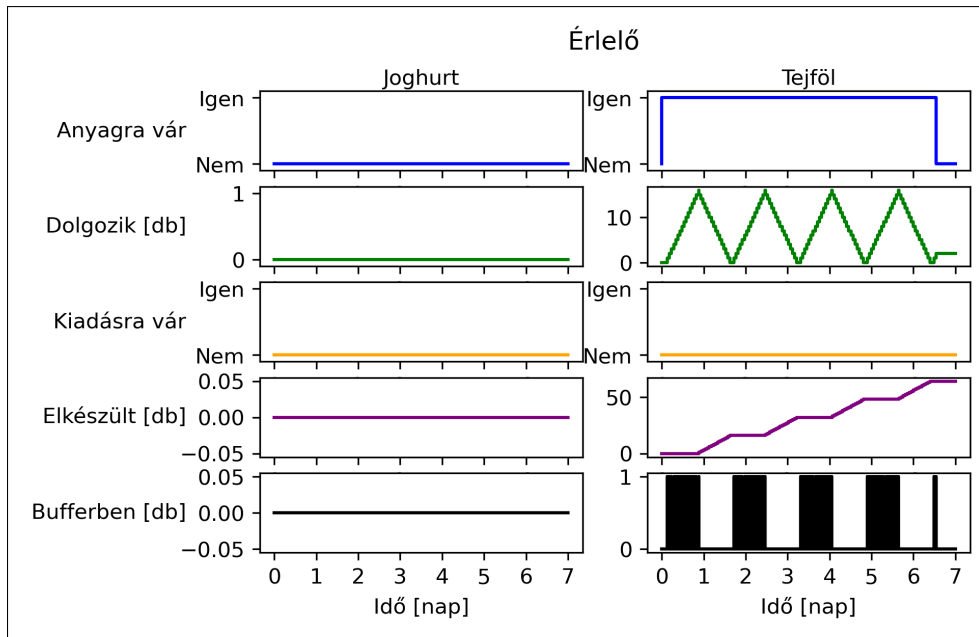
A **refrigerating** függvény pontos működése a következő. Miután elindul, belép egy végtelen ciklusba, ami a következő folyamatokat ismétli.

Első lépésként egy while ciklus segítségével megvárja, amíg lesz egy anyag a pufferében. Ha várakozás közben fejeződne be a szimuláció, akkor mielőtt leállna, rögzíti a várakozáshoz kapcsolódó adatokat. Abban az esetben, ha érkezett anyag a pufferébe, a program rögzíti a várakozáshoz kapcsolódó adatokat.

Ezután megnézi, hogy van-e szabad kis objektum, aminek átadhatná az anyagot. Ha nincsen, szintén elkezd várni addig, amíg nem lesz egy szabad hely. Abban az esetben, ha eközben állna le a szimuláció, nem kell adatot gyűjteni, mert a leálláshoz kapcsolódó adatot majd a kis objektum fogja felírni.

Végül már csak annyit tesz a program, hogy a pufferében lévő anyagot átrakja a szabad helyet reprezentáló gép pufferébe, ahonnan az azonnal bekéri és elkezd vele dolgozni.

Az érlelő esetében mivel egyszerre akár 16 raklapot is tud érlelni, így a dolgozik ábránál az a fontos, hogy hányal, ezért módosítottam az ábrát. Azt, hogy melyik másodpercben éppen hány anyagot érlel a szimulátor a **refrigiter_data** függvény gyűjti, ami csak akkor fut, ha az adatot vizualizált formában is megszeretnénk jeleníteni. Az érlelő egy hetes működését a mutatja be a 4.5 ábra.



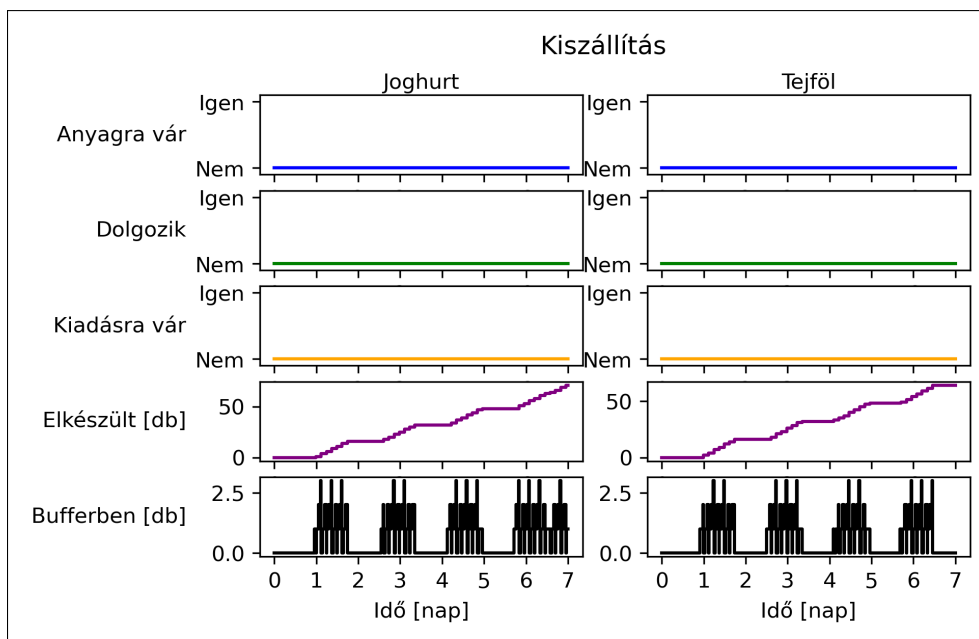
4.5. ábra. Az érlelő munkáját leíró adatok vizualizációja.

4.7. Kiszállítás

Ebben az esetben a program nem egy gép működését, hanem egy fix időnként bekövetkező eseményt modellez a **delivery** felhasználó függvénnyel segítségével, az alábbi módon.

Első lépésként belép egy végtelen ciklusba a függvény. Ezután a gép megvárja, hogy elteljen a két szállítás közötti szükséges idő. Ezután egymás után kiüríti azokat a puffereket, amibe a kész raktárba a gép pakolta a kartonokat. A program rögzíti, hogy a pufferben kevesebb anyag van, hogy többet szállítottak el a rendszerből, illetve feljegyzi, hogy mennyi profit érkezett az eladásnak köszönhetően a céghez. A kartonokat szimbolizáló objektumok két listába kerülnek, így onnét felhasználhatóak még arra, hogy a program megfelelő működését lehessen tesztelni.

Mivel ez egy esemény, amikor kiszállítják a kész kartonokat, nem értelmezem az anyagra várakozást, a munkát, és a kimenetnél bekövetkező torlódást. Itt a legfontosabb az, hogy a pufferben lévő anyagok elhagyják a rendszert. Ahogyan ez látható a 4.6 ábrán.



4.6. ábra. A kiszállításhoz kapcsolódó adatok vizualizációja.

4.8. Validálás

Alapvetően akkor jó a szimulátor, ha képes egy hét alatt legyártani azt az anyagmennyiséget, mint a rendes üzem. Természetesen ez alól kivétel az első hét, amikor elindul a rendszer, mert ebben az esetben, mivel üresen indulnak a gépek, kevesebb az elvárt mennyiség. Jelen esetben a tejfölöknél tökéletesen tudja követni az elvárást. A joghurt esetében viszont az első hét után egy öt hetes ciklusban ingadozik a termelés, de átlagosan hozza az elvárt 75 kartont. A 4.2 táblázat mutatja, az első hat szimulált hétben hetente hány karton tejföl és joghurt készül el.

Hét	1	2	3	4	5	6
Joghurt	71	74	77	74	76	74
Tejföl	64	66	66	66	66	66

4.2. táblázat. A hetente legyártott termékek mennyisége

5. fejezet

Karbantartási és gazdasági adatok

5.1. Bevétel adatok és hibamentes esetben a profit alakulása az időben.

Ahhoz, hogy megtudjam határozni, hogy körülbelül mennyibe kerül előállítás és mekkora profitot tud realizálni a cég egy karton tejfölön és joghurton. Kikerestem a Központi Statisztikai Hivatal (KSH) adatbázisából, hogy 2024 áprilisában mennyi volt ezeknek a termékeknek a fogyasztói ára. Forrás: https://www.ksh.hu/stadat_files/ara/hu/ara0044.html. Ha gépről olvassa ezt, akkor **ide kattint** egyből megnyílik a forrás.

Fontos, adat még, hogy pontosan mekkora a kiszerelés. Jelen esetben alsó és felső tömegkorlátot adtak meg a termékeknek, így majd én az átlaggal fogok számolni a továbbiakban. A KSH adatbázisában a joghurtnál csak gyümölcsös volt, illetve a tejfölnél a 12%-os zsírtartalmúval számolok tovább, mert a kiszerelés mérete itt van a legközelebb az korábbi számolás eredményéhez. Az 5.1 táblázat tartalmazza a kigyűjtött adatokat.

Termék típus	Tömeg alsó korlát [g]	Tömeg felső korlát [g]	Fogyasztói ár [Ft]
Joghurt	125	150	198
Tejföl	325	450	510

5.1. táblázat. A kigyűjtött KSH adatok.

Ez alapján úgy határoztam meg egy szimulált karton fogyasztói árát (F), hogy vettem a tömeg alsó és felső korlátjának az átlagát, majd ezzel elosztottam a fogyasztói árát. Ezzel megkaptam, hogy egy gramm termék mennyibe kerül. Ezt a számot már csak meg kellett szoroznom a kiszerelés tömegével, hogy egy kartonba mennyi termék kerül, illetve, hogy a szimulált karton hány kartonnak felel meg a valóságban. A két terméknel a számolást és egészen felkerekített értékét mutatja az 5.1 és 5.2 egyenletek. Ez viszont csak az ár,

amit a kereskedő a vásárlótól kap.

$$F_{Joghurt} = \frac{198}{\frac{125+150}{2}} \cdot 342 \cdot 960 \cdot 3,45 = 1631094 \text{ Ft} \approx 1,63 \text{ M Ft} \quad (5.1)$$

$$F_{Tejföl} = \frac{510}{\frac{325+450}{2}} \cdot 382 \cdot 960 \cdot 3,45 = 1665146 \text{ Ft} \approx 1,66 \text{ M Ft} \quad (5.2)$$

Ez úgy alakul ki, hogy az önköltségre, ami a termék előállításának az ára, rájön a haszonkulcs, ami a termék előállításának valahány százaléka. Én úgy számoltam, hogy ezen az áron tudja eladni a kereskedőnek, aki szintén ráteszi a termékre a saját hasznát és az általános forgalmi adót.

Az általános forgalmi adónál (ÁFA) az ár 18%-ával számoltam. Az Áfa tv. 82. § (3) bekezdése és az Áfa tv. 3/A. számú melléklet I. része alapján. A forrásdokumentumot, ha gépről olvassa ezt a tisztelt olvasó **ide kattintva** letöltheti.

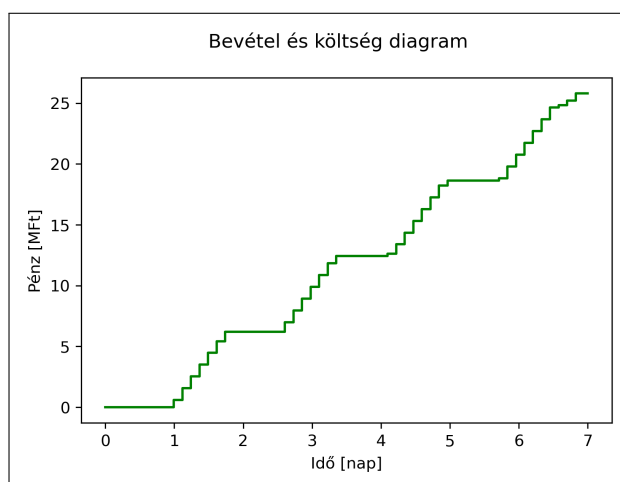
A kereskedői példatárban legtöbbször 20 %-os haszonkulccsal számolnak, így én is ezzel fogok [15].

Azaz a fogyasztói árat el kell osztanom 1,6992 -vel ($1,2 \cdot 1,2 \cdot 1,18 = 1,6992$), hogy megkapjam az előállítási árat. Ha ennek veszem a 20%-át, akkor megkapom a terméken megjelenő hasznat. A konkrét eredményeket mutatja az 5.2 táblázat.

Termék típus	Előállítási ár [Ft/karton]	Haszon [Ft/karton]
Joghurt	959919	191984
Tejföl	979959	195992

5.2. táblázat. A termelőhöz kapcsolódó fontos pénzügyi adatok.

A haszon adatot betápláltam a szimulátorba, és így lehetséges már a hibamentes esetben ábrázolni a bevétel alakulását az időben. Az első héten a nyereség alakulását mutatja a 5.1 ábra.



5.1. ábra. Az első héten az üzem nyereségének alakulása hibamentes működés esetén.

5.2. A meghibásodáshoz és karbantartáshoz kapcsolódó paraméterek

A cégtől a meghibásodáshoz kapcsolódóan nem kaptam adatokat, ezért nekem kellett a paramétereket más forrásból beszerezniem.

Hogy ezt megtudjam tenni, első lépésként csoportosítam a szimulátorban lévő gépeket úgy, hogy a csoport elemeinek a meghibásodását tudjam közelíteni ugyanolyan paraméterekkel rendelkező véletlenekkel. Négyféle kategóriát sikerült így létrehoznom, amik a következők: futószalag, csomagológép, poharazó és hűtőkamra. Azt, hogy a konkrét kategóriákba melyik gépek kerültek, mutatja az 5.3 táblázat.

Csoport megnevezése	Ide tartozó gépek
Futószalag	Kupak csőből a csavározóba Forgóasztal Teli raklap puffer Kész raktárba
Poharazó	Termék 1 az üvegbe Termék 2 az üvegbe
Csomagológép	Csavarozógép Raklapra rakó
Hűtőkamra	Érlelő Kiszállítás

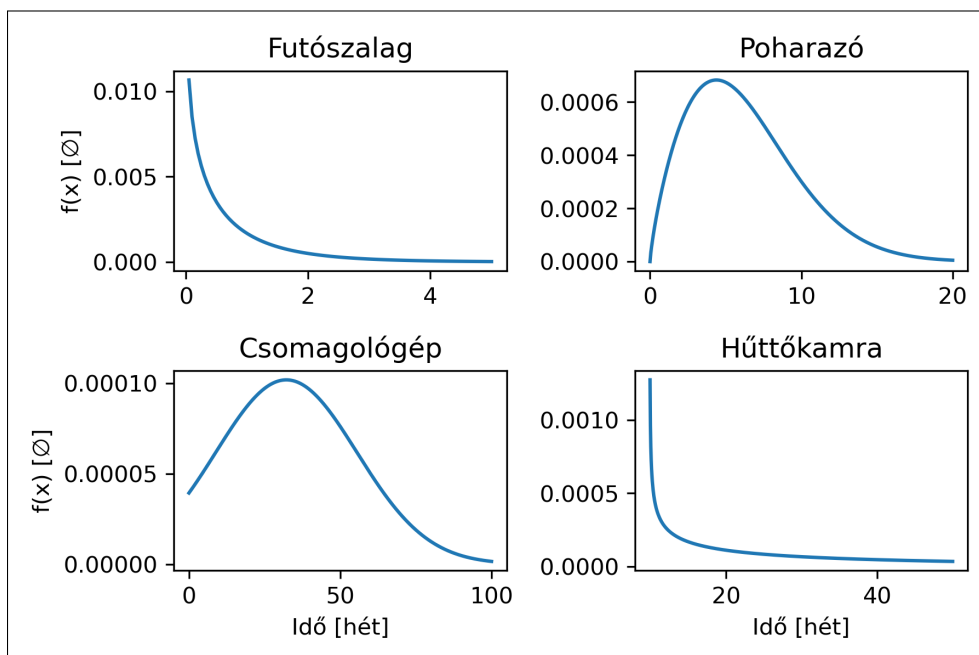
5.3. táblázat. A szimulátorban lévő gépek csoportosítása.

Ezután olyan kutatásokat kerestem, amik reálisak lehetnek a mostani feladatnál és feltűntetik, hogy a meghibásodásokat milyen típusú véletlenekkel közelítették, illetve ezeket hogyan paraméterezték. A következő 5.4 táblázat feltűnteti, hogy az egyes kategóriákhoz milyen forrást találtam, illetve abban milyen véletlennel dolgoztak és ezeket hogyan paraméterezték.

Csoport megnevezése	Véletlen típus	Paraméterek
Futószalag [8]	Három paraméteres weibull	$\alpha = 109,508 \text{ h}$ $\beta = 0,818$ $\gamma = 0,4 \text{ h}$
Poharazó [7]	Kettő paraméteres weibull	$\beta = 1,78$ $\eta = 1171 \text{ h}$
Csomagológép [6]	Normál eloszlás	$m = 5407 \text{ h}$ $\sigma = 3918,58 \text{ h}$
Hűtőkamra [6]	Három paraméteres weibull	$\alpha = 5421,31 \text{ h}$ $\beta = 0,634$ $\gamma = 1657,24 \text{ h}$

5.4. táblázat. Az egyes géptípusoknál a meghibásodási paraméterek.

Azért, hogy jobban meg lehessen érteni az adatokat, ábrázoltam a különböző véletlenek sűrűségfüggvényeit. Ezt mutatja a 5.2 ábra.



5.2. ábra. A különböző gépfajták meghibásodásának sűrűség függvényei

A meghibásodásokhoz kapcsolódóan fontos még az, hogy mennyi ideig tart a javítás és a karbantartás, illetve mekkora ezeknek a költsége. Azért, hogy az eredmények tükrözzék a hazai viszonyokat magyar karbantartó cégekkel vettem fel a kapcsolatot és tőlük kértem el ezen adatokat. Kérték, hogy elérhetőségüket ne közöljem így csak a kapott adatokat mutatja az 5.5 táblázat.

Csoport megnevezése	Karbantartás költsége [Ft]	Karbantartás hossza [h]	Javítás költsége [Ft]	Javítás hossza [h]
Futószalag	33600	4	50400	6
Poharazó	20000	4	300000	4,5
Csomagológép	45000	3	95000	3
Hűtőkamra	34500	3	69000	6

5.5. táblázat. A karbantartó cégektől elkért adatok.

A karbantartási idő meghatározásához kellő utolsó adat, hogy a rendelés nem teljesítése esetén mekkora kötbért kell fizetni. Egy haza jelentés szerint ennek az értéke az eladási árnak 30 – 50% a nagy kereskedelmi láncoknál [11]. Ezért én 40 % fogok számolni, azaz a kötbér értéke joghurt esetén 460762 Ft és tejföl esetén 470381 Ft

6. fejezet

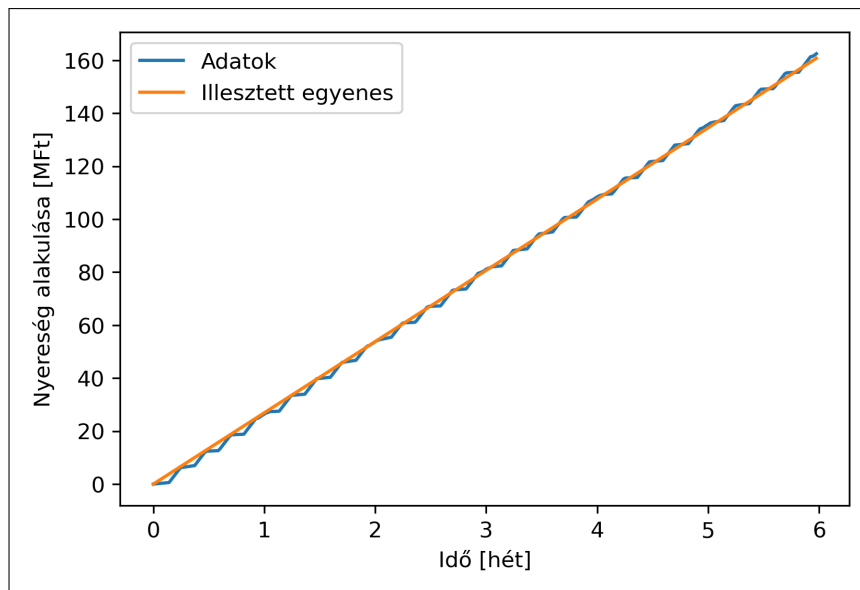
Monte Carlo módszerrel támogatott karbantartás ütemezés

6.1. Egy meghibásodáshoz kapcsolódó karbantartás meghatározása

Magától értetődő lenne, hogy a meghibásodást és a karbantartást a szimulátorba építsem be, és majd azt futtatva abból nyerjem ki az adatokat. Viszont ebben az esetben nagyon hosszú futtatási idők lépnének fel.

Az érlelőnél például ahhoz, hogy 90% valószínűséggel egyszer meghibásodjon, körülbelül 130 hetet kellene modellezni. Az általam írt program egy hetet nagyjából egy perc alatt szimulál le a saját számítógépemen, így ebben az esetben egy szimuláció kettő óra tíz perc hosszú lenne. Abban az esetben, ha csak 1000 futtatást szeretnék végezni, akkor is több mint 90 napig tartana a program lefutása. Ezért kidolgoztam egy olyan módszert, amihez kapcsolódó programot lényegesen gyorsabban lehet futtatni.

A módszerhez szükségem van egy olyan egyenesre, ami a hibamentes működést közelíti. Ezért a szimulátorral hat hetes működést modelleztem. Azért hat hétnyi adattal dolgoztam, mert az első hét után a joghurt előállítás így pont egy teljes ciklust ingadozik, mivel öt hetes az eladott mennyiség változásának periódus ideje. A bevételt leíró adatokra illesztettem a legkisebb négyzetek módszerével egy olyan egyenest, ami átmegy az origón. Az adatok és az illesztett egyenes látható a 6.1 ábrán.



6.1. ábra. A hibamentes működés leíró egyenes az adatokkal együtt.

Az ábrán jól látható, hogy a bevétel növekedése alapvetően lineáris, ezért az egyenes jól illeszkedik az adatokra.

Ahhoz, hogy meghatározzam a karbantartási időpontot, szükségem van még egy olyan egyenesre, ami a meghibásodás következményeit figyelembe veszi, illetve a karbantartáshoz kapcsolódó költségek összességére.

A meghibásodást tartalmazó egyeneshez öt pontra van szükségem. Amik a következők:

- Origó
- Meghibásodáskor elvesző költség
- Javítás ideje alatt nincs bevétel
- A javítást költségeinek a kifizetése
- A hiba miatt nem leszállított termékek utáni kötbér

Az origót. Ezt azért használom ezt fel, hogy az illesztett egyenesnél figyelembe vegyem, hogy kezdetben nulla a bevétel.

A második, ami a meghibásodáskor elvesző anyag, illetve az állás miatti költség. A meghibásodás időpontját a kettő vagy három paraméteres Weibull eloszlású véletleneknél úgy generáltam, hogy egy nulla és egy közötti egyenletes eloszlású véletlen számot behelyettesítettem az eloszlás függvény inverzébe. A normál eloszlású meghibásodásoknál pedig egy a Pythonba beépített véletlen generátort használtam, aminél ügyeltem arra, hogy mindenképpen nem negatív számot generáljon. Ezzel megkapom, hogy hány üzemóra után történik a meghibásodás. Azért, hogy meg tudjam mondani, hogy pontosan

mikor hibásodik meg, a meghibásodási üzemórát osztom a gép kihasználtságával, aminek a kettő tizedes jegyre kerekített értékét hat hetes szimulációnál mutatja a 6.1 táblázat.

Gép neve:	Kihasználtság [%]
Kupak csőből a csavarozóba	100
Forgóasztal	49,94
Termék 1 az üvegbe	26,64
Termék 2 az üvegbe	23,44
Csavarozógép	99,98
Raklapra rakó	99,92
Teli raklap puffer	99,78
Érlelő	100
Kész raktárba	49,74
Kiszállítás	100

6.1. táblázat. A szimulátorban lévő gépek kihasználtsága hat hetes szimuláció esetén.

Az érlelő és a kiszállítás kihasználtsága azért 100%, mert folyamatosan kell mennie ezeknek a gépeknek, biztosítva, hogy mindenkor megfelelő körülmények közé kerüljön a termék.

A második pontnál fontos még, hogy milyen termék kerül kidobásra. A legtöbb gép esetében körülbelül 53%-os eséllyel ($\frac{75}{141}$) joghurt és hozzávetőlegesen 47%-os valószínűséggel ($\frac{66}{141}$) tejföl vész kárba. Ez alól kivétel a termék 1 az üvegbe, termék 2 az üvegbe gép, az érlelő és a kész raktár, amiből a kiszállításkor elviszik az anyagot. A termék 1 az üvegbe gépnél 100%-os eséllyel joghurt vész el, a termék 2 az üvegbe gépnél meg biztosan tejföl. Az érlelőnél, hogy a hiba időpontjában pont mennyi karton tejföl van benne, úgy határoztam meg, hogy külön gyűjtöttem a hat hetes szimuláció alatt, hogy mikor mennyi karton tejföl érlelődött. Ha a meghibásodás az első hat hétben történik, akkor konkrétan meg tudom mondani, az adatok alapján mennyi karton van benne. Ha nem, akkor a meghibásodás időpontjából levonok hat hetet, majd megnézem, hogy a maradék időnek mennyi az öttel való osztási maradéka, ezt az időt hozzáadom az első hétkez, így az adatok alapján meg tudom mondani, hogy hány karton érlelődött éppen. A logika pontosan ugyanaz a kész raktár esetében is, csak ott a kiszállítás gép puffer adataival dolgoztam.

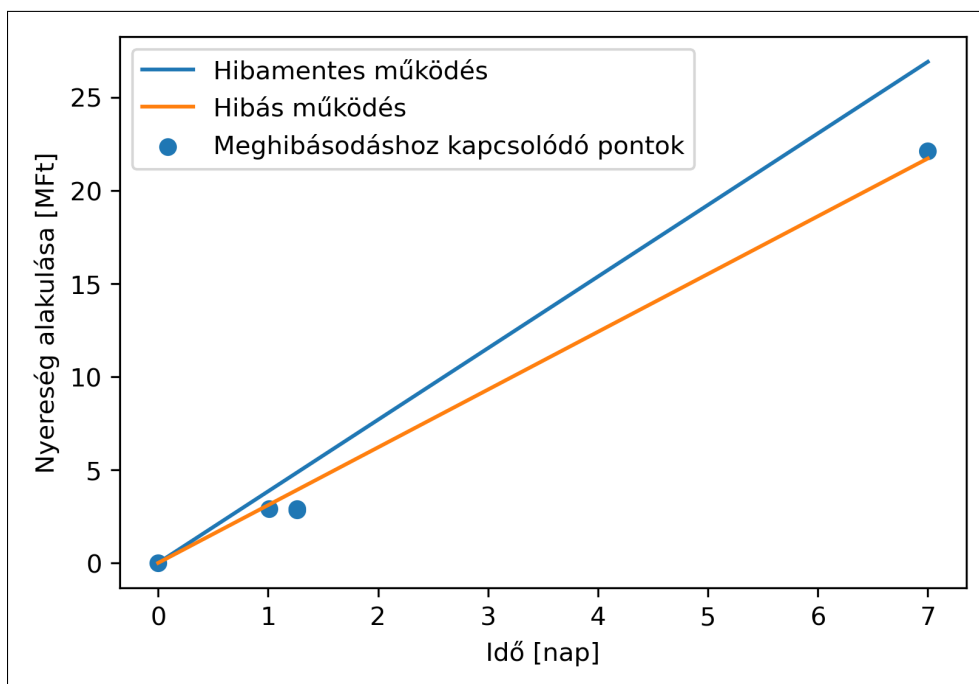
A harmadik pont reprezentálja, hogy a javítás ideje alatt nincs bevétel. Az, hogy mennyi ideig áll a gép, típusától függ. Az állás költségét a második pontban veszem figyelembe. Sajnos konkrét adatok hiányában nem tudtam ennél jobban szétbontani a működést.

A negyedik pont írja le, hogy a javítás után a cég kifizeti a karbantartókat, ennek a költsége szintén a gép típusától függ.

Az utolsó pont pedig, hogy az állás miatt a hét végén a le nem szállított anyag miatt a cégnek kötbért kellett fizetnie. Itt, hogy konkrétan mennyi volt az összes bevétel, úgy

határoztam meg, hogy a hibamentes működéssel párhuzamost húztam a negyedik ponton keresztül a hét végéig, majd abból levontam a kötbér értékét. Amit úgy határoztam meg, hogy a javítás idejét elosztottam a takt time-mal, így megkaptam, átlagosan mennyi anyagot nem szállítanak le, ha ennyi ideig áll a rendszer. Viszont a nem teljesített rendelések számának egész számnak kell lennie. Ezért generáltam egy nulla és egy közötti egyenletes eloszlású véletlen számot, ha ennek az értéke nagyobb, mint a nem teljesített rendelések szám nem egész része, akkor lekerekítettem, ellenkező esetben felkerekítettem. Azt pedig, hogy pontosan tejfölt vagy joghurtot nem szállítottak le úgy határoztam meg alapvetően a gépeknél, hogy körülbelül 53%-os eséllyel ($\frac{75}{141}$) joghurt és hozzávetőlegesen 47%-os valószínűséggel ($\frac{66}{141}$) tejföl legyen az. Kivétel ez alól a termék 1 az üvegbe, termék 2 az üvegbe gép és az érlelő. Az elsőnél biztosan joghurtot nem tudnak gyártani. A többinél, meg tejfölt.

Erre az öt pontra egy az origón átmenő egyenest illesztettem. A hibamentes és a meghibásodást figyelembe vevő egyenes a meghibásodáshoz kapcsolódó öt ponttal együtt látható a 6.2 ábrán, ami a kupak csőből a csavarozóba gép egy meghibásodásakor meghatározandó egyenest mutat be. Fontos megjegyezni, hogy a javítás alacsony költsége miatt a hármas és a négyes pont átfedi egymást. Ezért tűnik úgy, mintha csak négy pont lenne az ábrán.

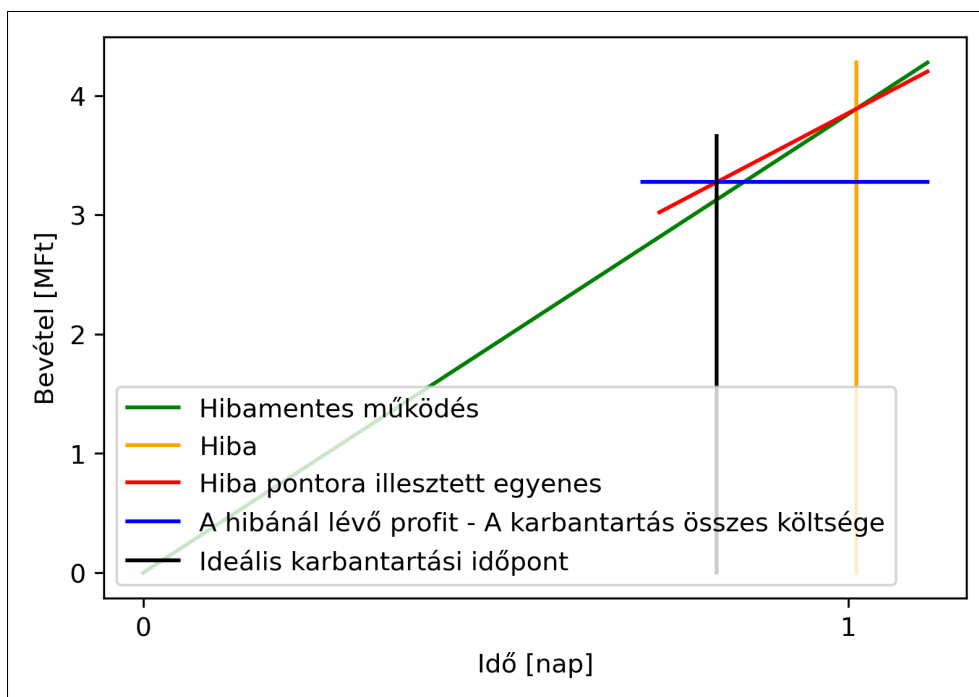


6.2. ábra. A hibás működést leíró illesztett egyenes.

A karbantartás költsége kettő elemből áll. Az első, hogy meg kell fizetni a karbantartókat, ez csak a gép típusától függ. A második, hogy mennyivel kevesebb terméket lehet a karbantartás miatt legyártani. Ennek az értékét pontosan ugyanúgy határoztam meg,

mint a kötbér értékét, csak itt a karbantartás hosszát osztottam el a takt time-mal és a darabszámnál nem a kötbért, hanem a profitot vettem.

A karbantartási időpont meghatározása úgy történik, hogy a meghibásodáskor lévő költségből levonom a karbantartás költségét, majd a meghibásodást figyelembe vevő egyenessel párhuzamost húzok a hibamentes egyenes értékénél a meghibásodáskor, és ahol el metszi a karbantartás költségét, oda ütemezném be az abszolút időben a karbantartást. A karbantartási időpont meghatározása látható a 6.3 ábrán.



6.3. ábra. Abszolút időben a karbantartási időpont meghatározása

Ezzel csak azt tudom megmondani, hogy mennyi időnek kell eltelnie két karbantartás között, de mivel nem folyamatosan dolgozik a gép, ezért ezt a két karbantartás közötti időhosszt meg kell szoroznom a gép kihasználtságával, hogy megtudjam, hány üzemóra után kell azt karban tartani. Előfordulhat, hogy negatív lenne a javasolt karbantartási üzemóra, ez azt jelenti, hogy nagyon hamar meghibásodott, ezért a karbantartásnál a gép állapotát érdemes alaposabban felmérni.

Ez egy karbantartási időpont meghatározásának a módszere. A Monte Carloval pedig képes vagyok ennek a több véletlentől függő számnak a várható értékét meghatározni úgy, hogy veszem a szimulált eredménynek átlagát.

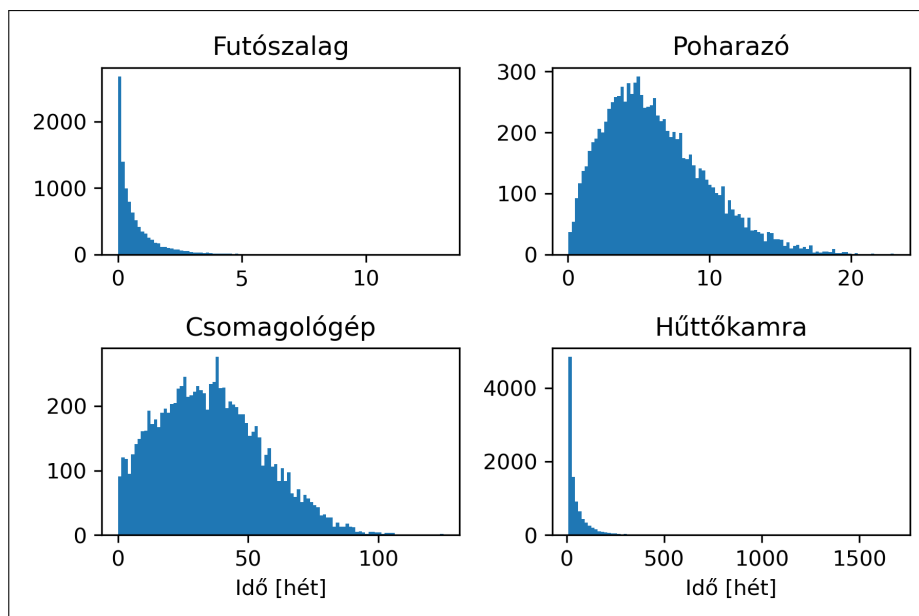
6.2. Eredmények

Az összes gép esetében 10000 véletlen időpontban lévő meghibásodásnál határoztam meg a 6.1 fejezetben leírt módon, hogy mikorra kellett volna ütemezni a karbantartást a hiba elkerülése érdekében. Ezeknek az üzemóra számoknak vettem az egy, a 10, 50, 90, és a 99%-os percentilisét, illetve, hogy a szimulációk hány százalékában lett negatív az üzemóra. Ezeket tartalmazza a 6.2 táblázat. A karbantartásig lévő üzemórát egészre kerekítve adtam meg. Ha annál a percentiliséknél negatív lett volna a javasolt üzemóra, akkor azt egy X jelöltem.

Gép neve	A karbantartási üzemórák ... %-os percentilise [h]					Korai a hiba [%]
	1	10	50	90	99	
Kupak csőből a csavarozóba	X	2	65	301	741	7,29
Forgóasztal	X	5	68	295	715	3,99
Termék 1 az üvegbe	88	328	950	1866	2742	0
Termék 2 az üvegbe	85	330	961	1854	2748	0
Csavarozógép	228	1681	5788	10619	14618	0,03
Raklapra rakó	257	1786	5835	10626	14814	0,02
Teli raklap puffer	X	2	65	301	718	7,41
Érlelő	1657	1804	4646	21288	61665	0
Kész raktárba	X	4	68	293	716	4,25
Kiszállítás	1658	1813	4725	22061	59921	0

6.2. táblázat. Az eredményeket bemutató táblázat.

Jól látható, hogy az eredmények nagyon hasonlóak az ugyanolyan meghibásodási paraméterekkel rendelkező gépeknél. A következő 6.4 ábra mutatja milyen az egyes géptípusoknál a kiszámolt karbantartási időpontok hisztogramja.



6.4. ábra. Az egyes gép típusoknál a következő karbantartásig kiszámolt üzemóra hisztogramja

Jól látható, hogy a hisztogram kinézete nagyon hasonlít a különböző gépek meghibásodását leíró meghibásodási sűrűség függvényekre, azaz a 5.2 ábrára. Ennek az az oka, hogy a meghibásodáshoz kapcsolódó véletlenek közül ez a legmeghatározóbb. Végezetül a 6.3 táblázat mutatja, egésze kerekítve, a várható értékét a következő karbantartásig eltelő üzemóráknak.

Gép neve:	A karbantartási időpont várható értéke [h]
Kupak csőből a csavarozóba	119
Forgóasztal	120
Termék 1 az üvegbe	1040
Termék 2 az üvegbe	1042
Csavarozógép	6001
Raklapra rakó	6097
Teli raklap puffer	119
Érlelő	9172
Kész raktárba	119
Kiszállítás	9240

6.3. táblázat. Az egyes gépeknél a várható értéke az üzemóráknak ami után karban kell tartani azokat.

A kidolgozott diszkrét eseményű szimulációs modell a Monte Carlo módszertannal további lehetőségeket kínál, amelyek során vizsgálhatóak nem csak a karbantartási események eloszlásainak hatásai, hanem akár a javítási idők bizonytalanságának hatása és az időben dinamikusan változó degradációs értékek hatásai és a folyamatban való egymásra hatásai is.

7. fejezet

Összefoglalás

A munkám során a karbantartás ütemezéssel foglalkoztam egy tejüzem esetében. Ami itt nagyon fontos, mert az alapanyag romlandó, illetve a biológiai folyamatok miatt folyamatosnak kell lennie a termelésnek. Illetve, jelen esetben a kereskedők nagy kötbérekkel dolgoznak a helyzetük miatt.

A munkám tartalmi része a harmadik fejezetben kezdődik, ahol a tejüzem működését leképeztem a Pannon Egyetem Ipar 5.0 laborjára. Ehhez meg kellett határoznom, hogy a modellnek mit és mennyit kell gyártania. A számolásom eredménye az lett, hogy az első hetet leszámítva átlagosan 75 kartonnyi joghurtot és 66 kartonnyi tejfölt kell elkészítenie a szimulátornak, hogy kövesse a reális működést. Ezen adatok segítségével ki tudtam számolni, hogy az egyes gépek mennyi idő alatt végeznek el egy munkát. Az eredményeket tartalmazza a 7.1 táblázat. Illetve a fejezet végén leírtam, hogy milyen ütemezéssel kell elindítani az egyes termékek gyártását.

Gép neve:	Munkához szükséges idő [s]
Kupak csőből a csavározóba	715
Forgóasztal	357
Termék 1 az üvegbe	358
Termék 2 az üvegbe	358
Csavározógép	715
Raklapra rakó	2145
Teli raklap puffer	4289
Érlelő	64800
Kész raktárba	2145
Kiszállítás	10723

7.1. táblázat. Egy munka elvégzéséhez szükséges idő.

A negyedik fejezetben azt írtam le, hogy hogyan épül fel a szimulátor. A fejezet végén pedig validáltam a modellt.

Az ötödik fejezetben leírtam, hogy honnan szereztem és vagy hogyan számoltam ki a karbantartási és gazdasági adatokat, amikkel dolgoztam. A gazdasági adatokat tartalmazza a 7.2 táblázat, a karbantartási adatokat meg a 7.3 táblázat.

Termék típus	Előállítási ár [Ft/- karton]	Haszon [Ft/karton]	Kötbér [Ft/karton]
Joghurt	959919	191984	460762
Tejföl	979959	195992	470381

7.2. táblázat. A termelőhöz kapcsolódó fontos pénzügyi adatok.

Csoport megnevezése	Véletlen típus	Paraméterek
Futószalag [8]	Három paraméteres weibull	$\alpha = 109,508 \text{ h}$ $\beta = 0,818$ $\gamma = 0,4 \text{ h}$
Poharazó [7]	Kettő paraméteres weibull	$\beta = 1,78$ $\eta = 1171 \text{ h}$
Csomagológép [6]	Normál eloszlás	$m = 5407 \text{ h}$ $\sigma = 3918,58 \text{ h}$
Hűtőkamra [6]	Három paraméteres weibull	$\alpha = 5421,31 \text{ h}$ $\beta = 0,634$ $\gamma = 1657,24 \text{ h}$

7.3. táblázat. Az egyes géptípusoknál a meghibásodási paraméterek.

A hatodik fejezetben leírtam, hogy egy konkrét meghibásodás esetében hogyan számoltam ki azt az időpuffert, aminél érdemes lett volna karbantartani. Minden gépnél a 10000-szer futtattam a programot, majd Monte Carlo módszerrel kiértékeltem az eredményeket. Külön megnéztem hány esetben jelezte a módszer, hogy nagyon korán meghibásodott az adott gép. A karbantartási időpontok fontos percentilisét a 7.4 táblázat tartalmazza, ahol X jelöltem, ha a modell olyan korai meghibásodást jelzett, amit el kellene kerülni.

Gép neve	A karbantartási üzemórák ... %-os percentilise [h]					Korai a hiba [%]
	1	10	50	90	99	
Kupak csőből a csavarozóba	X	2	65	301	741	7,29
Forgóasztal	X	5	68	295	715	3,99
Termék 1 az üvegbe	88	328	950	1866	2742	0
Termék 2 az üvegbe	85	330	961	1854	2748	0
Csavarozógép	228	1681	5788	10619	14618	0,03
Raklapra rakó	257	1786	5835	10626	14814	0,02
Teli raklap puffer	X	2	65	301	718	7,41
Érlelő	1657	1804	4646	21288	61665	0
Kész raktárba	X	4	68	293	716	4,25
Kiszállítás	1658	1813	4725	22061	59921	0

7.4. táblázat. Az eredményeket bemutató táblázat.

Az eredményeket azaz különböző gépeknél a várható értékét a karbantartásnak a 7.5 táblázat tartalmazza.

Gép neve:	A karbantartási időpont várható értéke [h]
Kupak csőből a csavarozóba	119
Forgóasztal	120
Termék 1 az üvegbe	1040
Termék 2 az üvegbe	1042
Csavarozógép	6001
Raklapra rakó	6097
Teli raklap puffer	119
Érlelő	9172
Kész raktárba	119
Kiszállítás	9240

7.5. táblázat. Az egyes gépeknél a várható értéke a meghatározott üzemórának ami után karban kell tartani.

A munkám eredménye egy új karbantartási ütemterv lett, amit át tudok adni a tej-üzemnek. Továbbá az általam létrehozott szimulátor segítségével más hasonló struktúrájú rendszereknél lehet létrehozni karbantartási ütemtervet szimpla átparaméterezéssel. A munkám folytatása az lenne, hogy bővítem, hogy több különböző kiszerezésű termék gyártására is legyen lehetőség, illetve, hogy a tejföl és joghurt gyártás több lépését is beleveszem a modellbe.

Köszönetnyilvánítás

Köszönettel tartozom Dr. Ruppert Tamásnak és Darányi András Pálnak, hogy sok egyéb teendőjük mellett vállalták, hogy a témavezetőim legyenek. A közös munka során olyan precízséget és alaposágot mutattak a téma feldolgozásakor és leírásakor, hogy biztos vagyok benne, hogy ez a munka a hosszú távú karrierutamra hatással lesz.

A cégnek hálás vagyok, hogy megbíztak bennem és elláttak a munkámhoz kapcsolódó adatokkal.

Szüleimnek is köszönetet szeretnék mondani, mert úgy érzem, hogy nélkülük nem tudtam volna létrehozni ezt a dolgozatot és nem csak azért, mert nélkülük nem is léteznék. Édesanyámnak azért, mert ott volt és támogatott a teljes munka során. Édesapámnak meg azért, hogy a munkafolyamat során rengeteg ötletemet türelmesen meghallgatta és gyakorlatias tanácsaival segített nekem, ha elakadtam valamiben.

Végezetül meg szeretném köszönni páromnak Csigó Anitának, hogy sokszor, amikor már nem úgy álltam ehhez a dolgozathoz, ahogyan ez megérdemli, életszerű példáival és hangzatos beszédeivel a helyes útra terelt.

Irodalomjegyzék

- [1] Paulina Gackowiec. General overview of maintenance strategies–concepts and approaches. *Multidisciplinary Aspects of Production Engineering*, 2, 2019.
- [2] Qi Hao, Yunjiao Xue, Weiming Shen, Brian Jones, and Jie Zhu. A decision support system for integrating corrective maintenance, preventive maintenance, and condition-based maintenance. In *Construction Research Congress 2010: Innovation for Reshaping Construction Practice*, pages 470–479, 2010.
- [3] Ernest C Fitch. *Proactive maintenance for mechanical systems*. Number 5. Elsevier, 2013.
- [4] Jong-Ho Shin and Hong-Bae Jun. On condition based maintenance policy. *Journal of Computational Design and Engineering*, 2(2):119–127, 2015.
- [5] Les Warrington, Jeffery A Jones, and Neil Davis. Modelling of maintenance, within discrete event simulation. In *Annual Reliability and Maintainability Symposium. 2002 Proceedings (Cat. No. 02CH37318)*, pages 260–265. IEEE, 2002.
- [6] Vaishali Govindrao Talkit, Pratiksha Honrao, Shivani Bhosale, Anand Bewoor, and Rajkumar Bhimgonda Patil. Application of the reliability analysis for modifications in maintenance scheduling. *Materials Today: Proceedings*, 77:579–585, 2023.
- [7] Narendra Kumar, PC Tewari, and Anish Sachdeva. Performance assessment of milk processing system using petri nets. *SN Applied Sciences*, 2:1–12, 2020.
- [8] Meiyang Li, SUN Yingqian, and Chuan Luo. Reliability analysis of belt conveyor based on fault data. In *IOP Conference Series: Materials Science and Engineering*, volume 692, page 012009. IOP Publishing, 2019.
- [9] Tamas Ruppert and Janos Abonyi. Integration of real-time locating systems into digital twins. *Journal of industrial information integration*, 20:100174, 2020.

- [10] Panneerselvam Sivasankaran and P Shahabudeen. Literature review of assembly line balancing problems. *The International Journal of Advanced Manufacturing Technology*, 73:1665–1694, 2014.
- [11] Szenczy Dániel Udvardi Attila Bíró Péter, Dr. Molnár László. A beszállító-védelmi szabályok érvényesülése a kereskedelemnek beszállító agráripari vállalatok szemszögéből. *Gazdasági Versenyhivatal*, 2010.
- [12] Leonardo Leoni, Filippo De Carlo, and Mario Tucci. Developing a framework for generating production-dependent failure rate through discrete-event simulation. *International Journal of Production Economics*, 266:109034, 2023.
- [13] József Fenyvessy, József Csanádi, János Csapó, and Zsuzsanna Csapó-Kiss. *Tejipari technológia: tej és tejtermékek a táplálkozásban*. Scientia Kiadó, 2014.
- [14] endaq - differences between condition-based, predictive, and prescriptive maintenance. <https://blog.endaq.com/differences-between-condition-based-predictive-and-prescriptive-maintenance>. Accessed: 2023-11-14.
- [15] Kovács Marianna. *KERESKEDŐ MESTERVIZSGÁRA FELKÉSZÍTŐ PÉLDATÁR*. Magyar Kereskedelmi és Iparkamara, 2019.