



SZÉCHENYI ISTVÁN EGYETEM
GÉPÉSZMÉRNÖKI, INFORMATIKAI ÉS
VILLAMOSMÉRNÖKI KAR

Alternatív ipari titkosítás fejlesztése

Korpai Richárd
Villamosmérnöki MSc
Automatizálási rendszerek

Konzulens:
Szántó Norbert

2023. január 11.
Győr

Tartalomjegyzék

1.	Bevezetés	2
2.	Szakirodalom áttekintés	3
2.1.	Ipar 4.0 és ipari IoT.....	3
2.2.	Kibervédelem és kriptográfia.....	5
2.3.	Az Enigma kódológép.....	10
2.4.	Programozható logikai vezérlők	15
3.	Alternatív ipari titkosítás fejlesztése	20
3.1.	Felhasznált eszközök és szoftverek.....	20
3.2.	A fejlesztés folyamata	23
3.2.1.	A titkosító függvény.....	24
3.2.2.	Kapcsolótábla.....	25
3.2.3.	Rotorok	25
3.2.4.	Reflektor	26
3.2.5.	Dekódoló függvény.....	27
3.2.6.	Kommunikációs függvény.....	27
3.2.7.	OPC UA szerver létrehozása.....	28
3.2.8.	Node-RED.....	29
4.	Eredmények értékelése és további fejlesztési lehetőségek	34
5.	Összefoglalás	37
6.	Irodalomjegyzék.....	38

Ábrajegyzék

1. ábra A kommunikáció és kapcsolatvédelem funkcionális felbontása [8]	7
2. ábra Az Enigma [12]	10
3. ábra A rotor belső felépítése [14]	11
4. ábra: Enigma kapcsolási rajza [11]	11
5. ábra A kapcsolótábla [14]	12
6. ábra Az irányítási folyamat általános blokkvázlata [18]	15
7. ábra: PLC programnyelvek [19]	16
8. ábra Kommunikációs piramis [21]	17
9. ábra Ipari vezérlőrendszerek rendszermodellje	20
10. ábra Kommunikációs híd	23
11. ábra A kapcsolótábla függvénye	25
12. ábra A rotor függvénye	26
13. ábra Balra forgató függvény	26
14. ábra A reflektor függvénye	27
15. ábra A kommunikációs adat blokk	27
16. ábra SiOME layout	28
17. ábra A változó OPC UA azonosítója	29
18. ábra S7 kommunikáció konfigurálása	30
19. ábra Változók listája	30
20. ábra Kommunikáció elindítása	31
21. ábra Kommunikáció megállítása	31
22. ábra Adatok lementése	31
23. ábra Felbontás bitekre	32
24. ábra A dekódoló függvény	32
25. ábra OPC UA szerver konfigurálása a Node-RED-en	33

1. Bevezetés

Napjainkban a legértékesebb árucikkek az adatok. Ezek az adatok értéket képviselnek, tehát a megőrzése kiemelten fontossá vált, főleg az iparban résztvevők körében. Az Ipar 4.0 megjelenésével egyre több eszköz kapcsolódik a globális internethez, így megnőtt az igény a megfelelő kibervédelem kialakítására. Sajnos azonban, amíg az ember részét képezi az információcsere folyamatának, addig még inkább fennáll a veszélye a sikeres behatolásoknak.

A dolgozatom célja egy olyan alternatív ipari titkosítás fejlesztése, aminek a segítségével a kommunikációban résztvevő eszközök között az adatok védelmét nem csak az ipari kommunikációs protokollok védelme biztosítja, hanem maga az adat is. Ez úgy lehetséges, hogy az adatot a kommunikációban lévő eszközök között titkosítjuk.

A fejlesztésem a II. világháborúban használt Enigma gépen alapszik, mellyel szöveges üzenetet kódoltak. Mivel a mai világban minden digitális és a kettes számrendszeren alapszik, így a fejlesztés során a betűk helyett számokat fogok alkalmazni, amit egy egyedi kriptográfiai módszerrel titkosítani fogok. A célom az, hogy a kommunikáció két végpontja közötti adat titkosítva legyen, és ennek a dekódolását a helyi ipari hálózaton kívül, egy erre kialakított rendszer végezzen. Ehhez két algoritmust létrehozása szükséges. Az egyik algoritmus a rendszerben található PLC-ken kódolja az elküldésre szánt információt, míg a másik az erre kialakított rendszerben dekódolja azt. Ezáltal hiába fér hozzá bárki az eszközök közötti kommunikációhoz a helyi hálózaton, az ott található adatok értelmezhetetlenné váltak. Innentől kezdve az adatfeldolgozásra csakis egy erre felkészített rendszer képes, így tovább növelve a biztonságot.

2. Szakirodalom áttekintés

Ebben a fejezetben a dolgozatomban előforduló és azzal kapcsolatos szakmai fogalmak kerülnek ismertetésre. Egyes témák csak általános lesznek bemutatva.

2.1. Ipar 4.0 és ipari IoT

Az Ipar 4.0, mint új ipari forradalom az egyik legnépszerűbb téma az ipar és a tudományos élet körében. Az Ipar 4.0 jelentős szerepet játszik abban a stratégiában, amely a termelési és szolgáltatási rendszerek minden szakaszának digitalizálásában rejlő lehetőségeket igyekszik megragadni. A negyedik ipari forradalom számos fizikai és digitális technológia, például a mesterséges intelligencia, a felhőalapú számítástechnika, az adaptív robotika, a kiterjesztett valóság, az additív gyártás és a dolgok internete (IoT) kombinációjával valósul meg. [1]

Az Ipar 4.0 víziója szerint a jövőben az ipari vállalkozások globális hálózatokat építenek ki, hogy a gépeket, gyárakat és raktárakat kiber-fizikai rendszerekként összekapcsolják, amelyek a cselekvéseket kiváltó információk megosztása révén, intelligens módon kapcsolódnak egymáshoz és irányítják egymást. Ezek a kiber-fizikai rendszerek intelligens gyárak, intelligens gépek, intelligens raktárak és intelligens ellátási láncok formájában fognak megjelenni. Ez a gyártás egészen belül az ipari folyamatok javulását fogja eredményezni a tervezés, az anyagfelhasználás, az ellátási láncok és a termékéletrajz-menedzsment révén. Ezeket nevezzük horizontális értékláncnak, és a jövőkép szerint az Ipar 4.0 mélyen integrálódik a horizontális értéklánc minden egyes szakaszával, hogy óriási javulást biztosítson az ipari folyamatokban. [2]

Az Ipar 4.0 kilenc technológiai pillérré épül. Ezek az innovációk áthidalják a fizikai és a digitális világot, és lehetővé teszik az intelligens és autonóm rendszereket. A vállalkozások és az ellátási láncok már most is használnak néhányat ezek közül a fejlett technológiák közül, de az Ipar 4.0 teljes potenciálja akkor használható ki, ha ezek együttesen kerülnek alkalmazásra. [3]

Ezek a technológiai pillérek következők:

- Big data és AI elemzések
- Autonóm robotok
- Szimuláció/digitális iker
- Horizontális és vertikális rendszer integráció
- IIoT

- Kibervédelem
- Felhőalapú számítás
- Additív gyártás
- Kiterjesztett valóság [2]

A dolgok internete (Internet of Things, IoT) egy olyan világ vízióját foglalja magában, amelyben beágyazott intelligenciával, kommunikációs eszközökkel, érzékelő és aktuátor képességekkel rendelkező tárgyak milliárdjai kapcsolódnak majd össze IP (Internet Protocol) hálózatokon keresztül. A jelenlegi internetünk alapvető átalakuláson ment keresztül, a hardvervezérelt (számítógép, optikai szál és Ethernet-kábel) lehetőségektől a piacvezérelt (Facebook, Amazon) lehetőségek felé. Ez az erős horizontális szoftveres képességekkel rendelkező, egymástól elszigetelt intranetek összekapcsolásának köszönhető. Az IoT nyílt környezeteket és interoperábilis platformok integrált architektúráját igényli. Az intelligens tárgyak és a kiber-fizikai rendszerek az IoT új egységei: a mindennapi élet tárgyai, amelyeket mikrovezérlőkkel, optikai és/vagy rádiós adó-vevővel, érzékelőkkel, aktuátorokkal és olyan protokollcsomagokkal egészítettek ki, amelyek alkalmasak a kommunikációra korlátozott környezetben, ahol a célhardver korlátozott erőforrásokkal rendelkezik, lehetővé téve számukra, hogy adatokat gyűjtsenek a környezetből és azok alapján cselekedjenek, és ezáltal interfészt biztosítva számukra a fizikai világgal. Ezeket a tárgyakat a felhasználók viselhetik, vagy a környezetükbe beépíthetik. Általában erősen korlátozottak, korlátozott memóriával és energiatárolókkal rendelkeznek, és alacsony költségigényűek. Az adattárolás, -feldolgozás és -elemzés alapvető követelmények, amelyek szükségesek a nyers IoT-adatok gyűjtéséhez és hasznos információvá alakításához. [4]

Az ipari IoT az általánosabb IoT egy részhalmaza. Ezért néhány jellemzőjük hasonló. A legáltalánosabb jellemzőjük, hogy mindegyik beágyazott számítástechnikai és kommunikációs technológiát tartalmaz. Ezek a rendszerek nagyrészt a szenzortechnológiára összpontosítanak az érzékszervi adatok gyűjtésével, továbbításával és feldolgozásával együtt. A kommunikáció nyilvánvalóan az ipari dolgok internetének kulcsfontosságú eleme. Az ipari dolgok internete vezetékes és vezeték nélküli kommunikációt egyaránt használhat. Az ipari IoT-eszközök által használt protokollok közé tartozik az Ethernet, Wi-Fi, WiMax, LR-WPAN, 2G/3G/4G telefon, IPv4, IPv6, 6LoWPAN, HTTP, CoAP, MQTT, XMPP, DDS és AMQP, Profinet, ModBus és DNP. A különböző felhasználási esetekhez különböző protokollok léteznek, amelyeket általában környezeti tényezők és erőforráskorlátok vezérelnek. A HTTP és az MQTT például alkalmazási szintű protokollok. A HTTP, a hipertext-továbbítási protokoll egy szövegalapú

protokoll, amelyet általában webalapú rendszerek, azaz webkiszolgálók használnak. Ez egy jó protokoll az ügyfél-kiszolgáló kommunikációhoz, amikor inkább csak egyirányú adatátvitelre van szükség. Bár mindkét irányba több adatsomagra van szükség ahhoz, hogy egy kliens lehívjon egy weboldalt egy kiszolgálóról, a protokollt tisztán kliens-kiszolgáló architektúrákra tervezték. Az IoT-eszközök esetében azonban gyakori, hogy kliensként és kiszolgálóként is működnek. Ezekben az esetekben a HTTP-t nehezebb megvalósítani, bár a lekérdezési módszertan segítségével ez is megoldható. Az MQTT-t kifejezetten ipari hálózati környezetekre tervezték. Ez egy publish-subscribe üzenetküldési protokoll, amely megkönnyíti a kétirányú kommunikációt, ahol egy eszköz ügyfélként és kiszolgálóként is működhet. Továbbá az átviteli többletköltségek szempontjából könnyű protokoll, és úgy tervezték, hogy támogassa a veszteséges adatátviteli hálózatokat. [5]

2.2. Kibervédelem és kriptográfia

A dolgok ipari internetének fényes és csillogó jövője lehet. Az ördög azonban a részletekben rejlik. Az ipari IoT első számú kihívása a biztonság és az adatvédelem. A kiberbiztonsági és adatvédelmi kérdések komoly akadályokat és nehézségeket jelentenek az ipari IoT-technológiák alkalmazói számára. Ha nem tudjuk enyhíteni az ipari IoT-t érintő számos biztonsági és adatvédelmi problémát, akkor nem fogjuk tudni teljes mértékben kihasználni a benne rejlő lehetőségeket. [5]

A „biztonság” kifejezés a tervezők fejében leginkább a „safety”, azaz üzembiztonság értelemben jelenik meg, melyre természetesen komoly erőforrásokat fordítanak. A „cybersecurity”, azaz kiberbiztonság megvalósítása viszont az Ipar 4.0 területén inkább kérdéseket vet fel egyelőre, tekintettel arra, hogy sokkal kevesebb tapasztalat áll rendelkezésre a komplex ipari rendszerek, azaz a kiber-fizikai rendszerek informatikai szempontú védelmével kapcsolatban, mint a hagyományos, fizikai térben történő fenyegetések kezelésében. [6]

A meglévő internetes technológiákat azonban olyan kiberbiztonsági és adatvédelmi problémák gyötrik, amelyek komoly kihívásokat és akadályokat jelentenek az Ipar 4.0 technológiák alkalmazói számára. Az Ipar 4.0 a hagyományos kiberbiztonsági kérdésekkel, valamint a saját, egyedi biztonsági és adatvédelmi kihívásokkal fog szembesülni. Ha ezeket a kihívásokat nem kezelik megfelelően, az Ipar 4.0-ban rejlő valódi lehetőségeket talán soha nem lehet kihasználni. [5]

A hagyományos biztonsági algoritmusok kriptográfiai technikákat használnak nyilvános és privát kulcsú mechanizmusok kifejlesztésére. A titkosítással együtt sok könnyűsúlyú biztonsági

technikát fejlesztettek ki, amelyek különböző személyazonosság-kezelési módszereket, hozzáférés-szabályozási technikákat, hitelesítési és engedélyezési mechanizmusokat tartalmaznak, és amelyeket a kutatók nagyra értékelnek. [7]

A dolgok internetével kapcsolatos biztonsági és adatvédelmi elveknek, a megfelelő eszközgazdálkodás, a kockázatok és fenyegetések időben történő, józan ítélőképességgel történő azonosítás részét kell képeznie. A szervezetekkel, az emberekkel és a folyamatokkal kapcsolatos ügyintézésnek tartalmaznia kell a dolgok életciklusának végét lezáró, bevált megoldásokkal történő támogatást, a biztonsági sebezhetőségek és incidensek megfelelő kezelésének biztosítását, a humán erőforrás biztonsági képzsét és tudatosítását, valamint a harmadik féllel való kapcsolatokra vonatkozó irányelveket. Az IoT biztonságának technikai oldala figyelembe veszi a hardverkomponensek biztonságát, a bizalom és az integritás kezelését, a magánélet védelmével kapcsolatos kérdéseket, az adatvédelmet és az adatok teljesítését, a rendszer biztonságát és megbízhatóságát, a szoftver és a firmware biztonságos és időben történő frissítését, a hitelesítést, az engedélyezést, a hozzáférés-szabályozást, a kriptográfiát, a biztonságos és megbízható kommunikációt, a biztonságos hálózati interfészeket és szolgáltatásokat, a biztonságos be- és kimeneti viselkedést, a naplózás részleteit, a teljes rendszer felügyeletét és ellenőrzését stb. [7]

A kiberbiztonság stratégiai alapelvei

A hatékony IoT-biztonság elsődleges biztonsági alapelveivel hat szempontból foglalkozik. Ezeket az elveket biztosítani kell ahhoz, hogy a biztonság a teljes IoT-rendszerben garantált legyen.

- **Bizalmasság:** A bizalmasság az információ elrejtésének képessége olyan személyek előtt, akiknek nincs jogosultságuk hozzáférni ahhoz, és ezért védelemre van szükségük a jogosulatlan hozzáférés ellen. A bizalmasság a dolgok internetének fontos biztonsági jellemzője. A legtöbb helyzetben és forgatókönyvben az érzékeny adatokat, például a betegadatokat, a magánkereskedelmi adatokat és/vagy a katonai adatokat, valamint a biztonsági hitelesítő adatokat és a titkos kulcsokat el kell rejteni a jogosulatlan hozzáférés előtt.
- **Integritás:** Az információ integritása az információ védelmére utal a jogosulatlan, nem várt vagy nem szándékos módosításokkal szemben. Az integritás a legtöbb esetben kötelező biztonsági tulajdonság annak érdekében, hogy megbízható szolgáltatásokat nyújthassunk a dolgok internetének felhasználói számára. A dolgok internetének különböző rendszerei különböző integritási igényekkel rendelkeznek.

- **Elérhetőség:** A rendelkezésre állás az eszköz felhasználójának (vagy magának az eszköznek) az információhoz való hozzáférése, amikor csak szüksége van rá. Ezért az IoT-erőforrásoknak időben rendelkezésre kell állniuk az igények kielégítése vagy a jelentős veszteségek elkerülése érdekében.
- **Hitelesség:** A hitelesség tulajdonsága lehetővé teszi, hogy a hálózatban csak az arra jogosult entitások végezzenek bizonyos műveleteket. A különböző hitelesítési igények különböző megoldásokat igényelnek. Egyes megoldásoknak erős ellenőrzésűnek kell lenniük, például a pénzügyi rendszerek hitelesítésének. Másrészt a legtöbbnek nemzetközinek kell lennie, például az ePassportnak, míg másoknak helyi szintűnek.
- **Letagadásmertesség:** Az IoT-szolgáltatásnak megbízható ellenőrzési nyomvonalat kell biztosítania. A letagadhatatlanság tulajdonsága bizonyos bizonyítékot jelent olyan esetekben, amikor a felhasználó vagy az eszköz nem tud letagadni egy műveletet, például a fizetési műveletet.
- **Adatvédelem:** Az adatvédelem egy entitás joga, hogy meghatározza, milyen mértékben lép kölcsönhatásba a környezetével, és milyen mértékben hajlandó személyes információkat megosztani másokkal. [1]

A kommunikáció és a csatlakoztathatóság védelme biztosítja a végpontok hálózathoz való kapcsolódásának a fizikai biztonságát, a hálózaton belüli információáramlás védelmét, valamint a végpontok közötti kommunikáció kriptográfiai védelmét. Ezt a két funkciót az 1. ábrán látható középső négy építőelemben található területek támogatják, melyek a következők: Hálózati konfiguráció és menedzsment, Hálózatfelügyelet és-elemzés, Kommunikációs végpontok védelme, A kapcsolatok fizikai biztonsága. [8]



1. ábra A kommunikáció és kapcsolatvédelem funkcionális felbontása [8]

A kommunikáció és az csatlakoztathatóság biztonságának a következő funkciókat kell figyelembe vennie a mozgásban lévő adatok védelme és ellenőrzése érdekében.

- A kapcsolatok fizikai biztonsága biztosítja a hálózathoz való fizikai kapcsolódási réteg (kábelek, rádiók) védelmét.
- A kommunikációs végpontok védelme biztosítja a funkcionális biztonsági építőelemek egy részét, például a kriptográfiai kulcsokat, a végpontok közötti biztonságos kommunikációhoz.
- A kriptográfiai védelem kriptográfiai technológiákat használ a kommunikáló felek hitelességének, valamint a kicserélt adatok és metaadatok integritásának és bizalmas jellegének védelmére.
- Az információáramlás-védelem biztosítja, hogy csak az engedélyezett üzenetek és tartalmak jussanak el az érzékeny rendszerekbe és hálózatokba a hálózati adatfolyam elszigetelésével, a hálózati szegmentációs és perimétermegvédelmi technológiák segítségével.
- A hálózati konfiguráció és menedzsment ellenőrzi az összes hálózati elem frissítését, és biztosítja a kommunikációs biztonsági politika és konfiguráció érvényesítését, beleértve a hálózati szegmentációt, a kriptográfiaileg védett kommunikációs beállításokat, valamint az átjárók és tűzfalak konfigurálását.
- A hálózati felügyelet és elemzés hálózati adatokat gyűjt elemzés céljából, és magába foglalja a behatolásérzékelést, a hálózati hozzáférés-ellenőrzést, a mély csomagvizsgálatot és a hálózati naplóelemzést.
- A mozgásban lévő adatok (Data-in-Motion) védelme az adatok integritásának, titkosságának és rendelkezésre állásának megőrzését szolgáló ellenőrzéseket biztosít. [8]

A kommunikációs és kapcsolati védelemre vonatkozó biztonsági irányelvek szabályozzák a kommunikációs biztonsági funkciók végrehajtását. A fent felsorolt funkciók egymástól függenek, és kölcsönhatásban vannak egymással a biztonsági képességek biztosítása érdekében. Például a biztonságos kommunikáció létrehozásához maguknak a kommunikációs végpontoknak is biztonságosnak kell lenniük, és kriptográfiai védelmet kell alkalmazniuk. Az összes funkciót átfogó adatmozgás-védelmi irányelv biztosítja a két végpont között közlekedő összes adatnak a titkosságát, sértetlenségét és rendelkezésre állását. Ezen túlmenően a kommunikációs és kapcsolódási védelem biztonsági irányelvei meghatározzák, hogy a hálózat elemei hogyan kommunikálhatnak egymással. Mindkét szabályzatnak átfogónak és egymással összhangban lévőnek kell lennie, és figyelembe kell vennie a rendszer egyéb kulcsfontosságú

jellemzőit, köztük a biztonságot, az adatvédelmet, a megbízhatóságot és a rugalmasságot, hogy megvédje a kommunikáció rendelkezésre állását, integritását és bizalmas jellegét. [8]

A kriptográfia

A kriptológia klasszikusan a titkos kommunikáció tudománya, amelynek a két fő ága a kriptográfia és a kriptanalízis. A kriptográfia azon algoritmikus módszerekkel foglalkozik, amelyek biztosítják az üzenetek (tárolt információk) titkosságát vagy hitelességét. A kriptanalízis a titok megfejtésére szolgáló módszerek tudománya. Az információk védelmi rendszerének az algoritmikus módszerek általában csak az egyik pillérét jelentik, amelyek megfelelő fizikai védelemmel valamint ügyviteli-rendszabályi eljárások alkalmazásával együtt biztosíthatják a valóságos rendszerekben az információvédelmet. A kriptográfia az algoritmikus módszerek alapja.[9]

Egy kriptográfiai rendszer használata során az elküldendő eredeti üzenettel, az úgynevezett nyílt szöveggel (plain text) kell kezdeni. A kriptográfiai rendszer segítségével az egyszerű szöveget olyan rejtjelezett szöveggé (cyper text) alakul át, amelyet a küldő szándéka szerint a címzetteknek kívül senki más nem érthet. Az egyszerű szöveg rejtjelezett szöveggé alakításának folyamatára felváltva használják a titkosítás és a kódolás szavakat, a rejtjelezett szöveg egyszerű szöveggé való visszaalakítására pedig a visszafejtés és a dekódolás szavakat. A titkosítás és a visszafejtés mindig valamilyen kulcsot használ. Egészen a közelmúltig a kriptorendszerek szimmetrikusak voltak, azaz a titkosításhoz és a visszafejtéshez ugyanazt a kulcsot használták. Ez szükségessé tette, hogy a kulcs olyan információ legyen, amelyet csak a feladó és a címzett ismer, és a titoktartás megőrzése kiemelkedő fontosságúvá vált. A modern kriptográfia ma már aszimmetrikus rendszerekre támaszkodik, amelyeknél a titkosításhoz használt kulcs nem azonos a visszafejtéshez használt kulccsal; a nyilvános kulcsot a feladó használja, a visszafejtéshez pedig egy csak a címzett által ismert (és így logisztikailag sokkal kevésbé veszélyeztetett) magánkulcsot. [10]

Az előbbi gondolatok alapján joggal merül fel a kérdés, hogy kell-e egyáltalán foglalkozni a klasszikus, szimmetrikus rejtjelező rendszerekkel. A válasz meglepő módon igen. A helyzet ugyanis az, hogy a szimmetrikus rendszerek lényegesen gyorsabbak, mint a nyilvános kulcsú eljárások, ezért a legtöbb esetben egy-egy konkrét üzenetváltás előtt a nyilvános kulcsú rejtjel segítségével a két partner kicserél egy kulcsot, és a továbbiakban az aktuális információt az így megismert kulcs segítségével, egy klasszikus módszerrel küldik egy nyilvános csatornán keresztül. [11]

2.3. Az Enigma kódológép

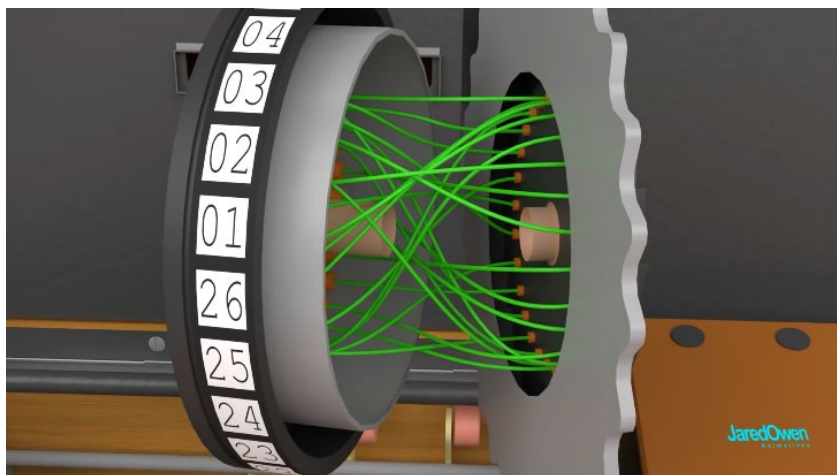
Az Enigma egy forgótárcsás rejtjelező gép (2. ábra), melyet az első világháború idején fejlesztettek ki és az évek során hetvennél is több különböző típust hoztak létre belőle. Egy ilyen rejtjelező gép három főbb egységből áll: a billentyűzetből, az átalakító egységből és a kijelző panelből. A billentyűzet segítségével vihető be a nyílt szöveg, amelyből az átalakító egység rejtjeles betűket készít. A felvillanó betűket leírva kapjuk meg a rejtjeles szöveget, amelyet azután rádióon továbbítottak a címzettnek. A vevő oldalon egy azonos beállítású eszközzel írták le a szöveget, és a lámpák felvillanása adta vissza a nyílt üzenetet.[11]



2. ábra Az Enigma [12]

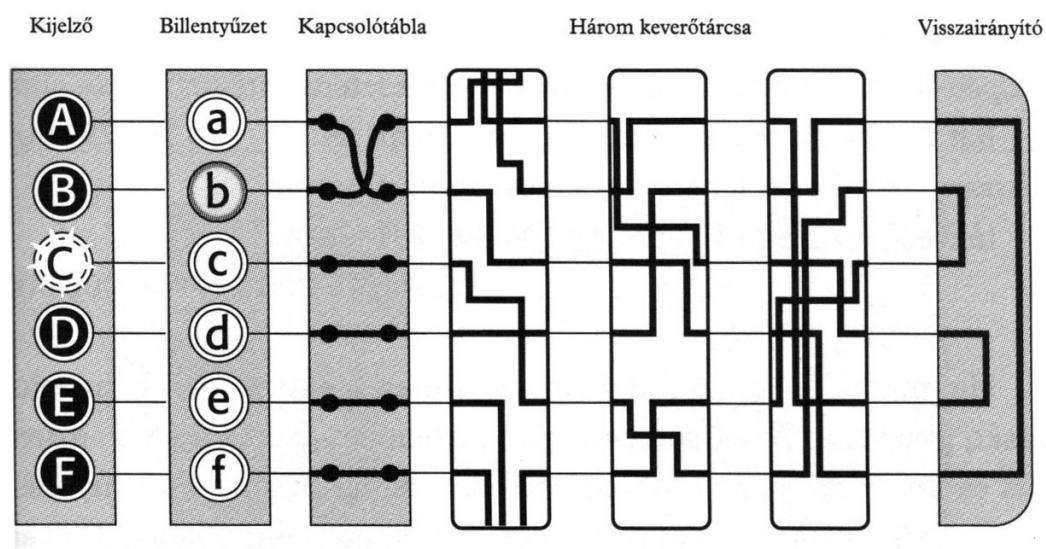
A gép legfontosabb része az átalakító egység. Mindegyik gép egy vagy több rotorral (keverőtárcsa) rendelkezik, a három és hat közötti kialakítás a leggyakoribb, de létezik, amelyik akár tízzel van felszerelve. [13]

A rotor egy vezetékekkel sűrűn teleszött, vastag gumitárcsa (3. ábra). A nyílt szöveg betűinek sifrírozását a rotorok belső huzalozása határozza meg. Ha a rotorok fix helyzetűek lennének, akkor a rotorok huzalozása egy egyszerű egyábécés helyettesítési eljárást valósítana meg. Sherbius gépének viszont a legfontosabb jellemzője, hogy a rotorok forognak. Minden egyes betű sifrírozása után az első rotor $1/26$ -odnyival elfordul (26 betűs ábécé esetén). A következő rotor csak akkor fordul $1/26$ -odnyit, ha az első rotor megtett egy teljes fordulatot. Ez a mechanizmus hasonlít az autók kilométerórájához. [11]



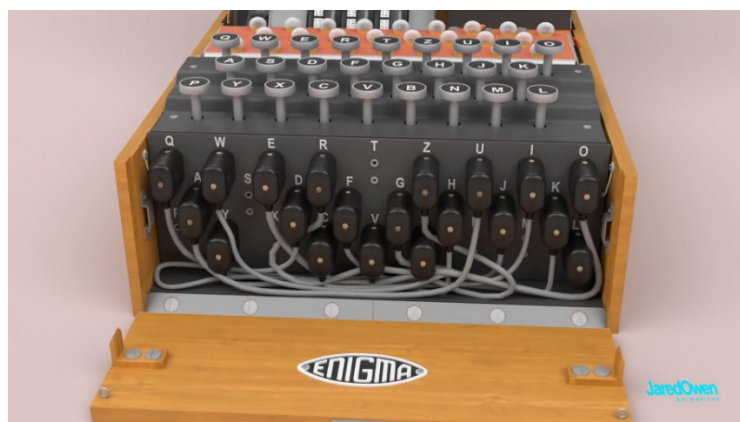
3. ábra A rotor belső felépítése [14]

Az Enigma kapcsolási rajzán (4. ábra) a rotorokon kívül látható még két további szerkezeti elem. A visszairányító és a kapcsolótábla. A visszairányító szintén egy belső huzalozású gumitárcsa, de nem forog. Mikor a kezelő begépel egy betűt, azzal egy elektromos jelet küld át a három rotoron. A visszairányító ezt a beérkező jelet küldi vissza, de más útvonalon. Az 4. ábrán látható rotor állások esetén a leütött „b” betű a „C”-t villantja fel, ha azonban a „c” lett volna leütve, akkor a kijelzőn a „B” villant volna fel. Ebből látható a visszairányító szerepe, a gép a nyílt szöveg egyik betűjét a rejtjeles szöveg egyik betűjévé alakítja, és ha egy másik gép ugyanígy van beállítva, akkor az előbb megkapott rejtjeles szöveg betűjét leütve megkapjuk az eredeti nyílt szöveg betűjelét, vagyis a sifrírozáshoz és a desifrírozáshoz ugyanaz a gép szükséges megegyező kezdő beállítással. [11]



4. ábra: Enigma kapcsolási rajza [11]

A másik új elem a kapcsolótábla (5. ábra), mely a billentyűzet és az első rotor közé van iktatva. E kapcsolótábla lehetővé teszi néhány vezeték beiktatását, amelyek még az első rotorba való belépés előtt felcserélnek bizonyos betűket. [11]



5. ábra A kapcsolótábla [14]

Három elemet kell figyelembe venni a lehetséges dugaljcsatlakozások számának kiszámításakor: a felhasznált kábelek számát, a kábelek fogadására kiválasztott aljzatszoportot és az aljzatszoporton belüli összeköttetéseket (azaz az egyes kábelek által létrehozott konkrét betűpárokat). Először az aljzatok kiválasztásával kell foglalkozni. Huszonhat aljzat van a dugaljzaton. Minden egyes kábel két aljzatot foglal el (egyet-egyet a kábel két vége). Mivel p ($0 \leq p \leq 13$) kábelt lehet a kapcsolótáblába illeszteni, ezért $\binom{26}{2p}$ különböző aljzatkombinációt lehet kiválasztani. [12,15]

Ezután meg kell határozni, hogy hányféleképpen lehet ezeket a p kábeleket a $2p$ kiválasztott aljzatba illeszteni. Miután az első kábel első vége behelyezésre kerül egy aljzatba, az első kábel második végének $2p - 1$ szabad aljzat áll rendelkezésre, amelyek közül választani lehet (az adott csoporton belül). A második kábel első végének behelyezésével, a kábel második végén $2p - 3$ szabad aljzat van. Ez a minta folytatódik a p -edik számú kábelig; ahol a második vég beillesztéséhez már csak egy szabad aljzat marad szabadon. Ezen a ponton egyértelművé válik, hogy a p kábel $2p$ szabad aljzatba történő behelyezésének összes lehetséges módját (úgy, hogy minden kábel két aljzatot foglal el) az (1) egyenlet adja meg.

$$(2p - 1) \cdot (2p - 3) \cdot (2p - 5) \cdot \dots \cdot 1 \quad (1)$$

Ezért a dugaljzatba illesztett p kábel esetén a fenti két elem kombinációjával kapott különböző csatlakozások száma:

$$\binom{26}{2p} \cdot (2p - 1) \cdot (2p - 3) \cdot (2p - 5) \cdot \dots \cdot 1 = \frac{26!}{(26-2p)! \cdot p! \cdot 2^p} \quad (2)$$

A harmadik és egyben utolsó elem, amelyet figyelembe kell venni, a felhasznált kábelek száma, vagyis p . Az imént kiszámított (2) egyenletet használva a p összes lehetséges értékére vonatkozó dugaljkombinációk számát az 1. táblázat tartalmazza. A gép egyik érdekes jellemzője, hogy a kombinációk maximális száma nem tizenháromnál következett be, ahogyan az várható lenne, hanem akkor, amikor a kezelők tizenegy dugaljkábelt használtak. [12,15]

1. táblázat [13,16]

p	Kombinációk	p	Kombinációk
0	1	7	1 305 093 289 500
1	325	8	10 767 019 638 375
2	44 850	9	53 835 098 191 875
3	3 453 450	10	150 738 274 937 250
4	164 038 875	11	205 552 193 096 250
5	5 019 589 575	12	102 776 096 548 125
6	100 391 791 500	13	7 905 853 580 625

Mivel az itt összeállított kombinációk p minden egyes értékére kölcsönösen kizárják egymást, így a lehetséges dugaljkombinációk teljes száma a fenti esetek összegzésével adódik, vagyis

$$\sum_{p=0}^{13} \frac{26!}{(26-2p)! \cdot p! \cdot 2^p} = 532\,985\,208\,200\,576 \quad (3)$$

A második változó komponens a három sorrendben (balról jobbra) elhelyezkedő rotor, amelyek huszonhat bemeneti érintkezési pontot huszonhat kimeneti érintkezési ponthoz kötnek, amelyek a rotor váltakozó felületein helyezkednek el. Ez az egyenlet egyszerű. Természetesen $26!$ egyedi rotort lehetne megépíteni. Ebből a $26!$ közül bármelyik kiválasztható, hogy a bal szélső pozíciót foglalja el. A középső pozíciónál már csak $26! - 1$ rotor közül lehet választani. A jobb szélső rotor pedig a $26! - 2$ rotor bármelyikéből kiválasztható. Tehát az összes lehetséges rotorkombinációnak a száma:

$$26! \times (26! - 1) \times (26! - 2) \quad (4)$$

vagyis 65 592 937 459 144 468 297 405 473 480 371 753 615 896 841 298 988 710 328 553 805 190 043 271 168 000 000. [12,15]

Az Enigma harmadik változó összetevője a drótozott lemezeket tartalmazó három rotor kezdeti forgási pozíciója. Ezt a kriptográfusok határozták meg, és a gépkezelők állították be a rotor peremén lévő huszonhat fogazás segítségével. Mivel a három rotor mindegyike huszonhat különböző pozícióba állítható, ezért a rotor kulcsbeállításainak a száma $26^3 = 17\,576$. [12,15]

A gép negyedik változó eleme egy mozgatható gyűrű a rotorok mindegyikén. A gyűrű minden egyes meghatározott helyen egy bevágást tartalmaz. A bevágás célja, hogy a rotor forgását közvetlenül balra kényszerítse, amikor a bevágás egy adott pozícióban van. A jobb szélső rotor minden egyes billentyű lenyomásakor elfordul. A jobb oldali rotor bevágása hoszonhat billentyűleütésenként egyszer kényszeríti a középső rotor elfordulását. A középső rotor bevágása $26 \cdot 26$ billentyűleütésenként egyszer kényszeríti a bal oldali rotor elfordulását. Mivel nincs több rotor, a bal szélső rotor bevágásának egyáltalán nem volt hatása. (Az összes rotor bal oldalán elhelyezkedő reflektor nem mozdul el.) Ezért a gép kriptográfiai erejéhez csak két rovátka járul hozzá. Mivel mindegyikük huszonhat lehetséges hely közül bármelyikre kerülhet, 26^2 vagyis 676 kombináció lehetséges. [12,15]

Az Enigma ötödik és egyben utolsó változó eleme a reflektor. A reflektornak huszonhat érintkezési pontja van, mint a rotornak, de csak az egyik oldalán. A huszonhat érintkezési pontot belsőleg tizenhárom vezeték köti össze páronként, hogy a rotorokból a reflektorba érkező kapcsolat a rotorokon keresztül egy másik útvonalon másodszor is visszakerüljön. A belső vezetékvezést a következőképpen lehet kialakítani. Amikor az első vezeték egyik végét az egyes érintkezési ponthoz csatlakoztatjuk, a vezeték másik végén huszonöt különböző érintkezési pont van, amelyekhez csatlakozhat. Így az első vezeték két érintkezési pontot foglal el, és huszonöt különböző lehetőség marad. A második vezeték szintén két érintkezési pontot használ fel, és csak huszonhárom különböző csatlakozási lehetőség marad a fel nem használt érintkezési pontokból. A harmadik vezeték további két érintkezési pontot foglal el, és huszonegy csatlakozási lehetőség marad. A mintázatnak mostanra már nyilvánvalónak kell lennie: az Enigmába helyezhető különböző reflektorok száma a következő:

$$25 \cdot 23 \cdot 21 \cdot \dots \cdot 1 = \frac{26!}{13! \cdot 2^{13}} = 7\,905\,853\,580\,625 \quad (5)$$

(Érdekes megfigyelni, hogy a különböző reflektorkombinációk száma megegyezik a lehetséges dugaljkombinációk számával $p = 13$ kábel használata esetén. Ez nem meglepő: mindkét esetben az érték a lehetséges páros kombinációk számát jelenti, amelyek huszonhat választási lehetőség és tizenhárom csatlakozó vezeték mellett lehetségesek). [12, 15]

Most már minden megvan az Enigma összes lehetséges beállítási variáció elméleti számának a meghatározásához. Ez egyszerűen a fent kiszámított öt érték szorzata.

Ez a megdöbbentően nagy szám a következő:

3 283 883 513 796 974 198 700 882 069 882 752 878 379 955 261 095 623 685 444 055
315 226 006433 615 627 409 666 933 182 371 154 802 769 920 000 000 000

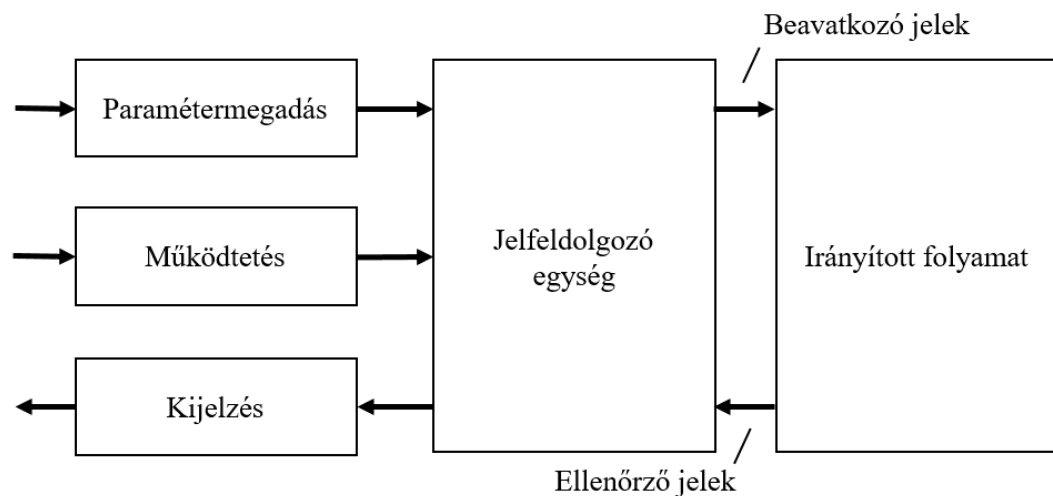
ami körülbelül $3 \cdot 10^{14}$. A szám nagyságának a szemeltetéseképpen érdemes megemlíteni, hogy a világegyetemben csak körülbelül 10^{80} atom van, így látható, hogy mégis milyen nagy ez a szám. [12, 15]

2.4. Programozható logikai vezérlők

Vitathatatlan, hogy az IIoT az ipari vezérlőrendszerekben vagy a kritikus infrastruktúrákban való alkalmazása kiváló lehetőségeket rejt magában a jövőben. Az IIoT koncepciója lehetővé teszi, hogy az ilyen rendszerek minden összetevője intelligens és hatékony módon kapcsolódjon egymáshoz és koordinálódjon. [16]

A programozható logikai vezérlők (Programmable Logical Controllers – PLC) az irányítási feladatokat tárolt program alapján valósítják meg. Napjainkban a vezérlési, irányítási feladatok túlnyomórészt PLC-k látják el. [17]

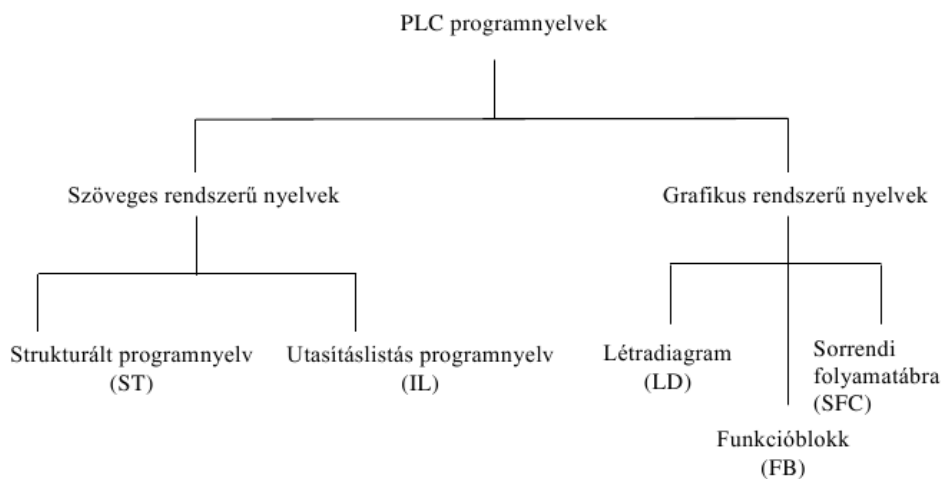
Az irányítás tárgyát képező technológiai folyamat, berendezés, gép stb. jelenti az irányított rendszert. Az irányítórendszer pedig mindazon szervek és készülékek együttese, amelyek együttműködése révén az irányított rendszer irányítása megvalósul. Az irányítási folyamat általános blokkvázlata látható a 6. ábrán: [18]



6. ábra Az irányítási folyamat általános blokkvázlata [18]

PLC programnyelvek

Az IEC 1131 szabvány a PLC felhasználói programnyelveket két osztályba sorolja: szöveges rendszerű és grafikus szimbólumokat alkalmazó programnyelvek. A szöveges szimbólumokkal leírt vezérlési feladat rendszerint egy compiler (fordító) révén kerül a programmemóriába letöltésre. A szöveges rendszerű programnyelveknek két megvalósítási formáját engedélyezik. Az egyik a magas szintű programnyelvekkel (Pascal vagy C) támogatott strukturált felhasználói programnyelv, a másik szöveges programnyelv az utasításlistás felhasználói programnyelv. A grafikus szimbólumokkal leírt és megszerkesztett vezérlési feladat a fejlesztőrendszerben egy letöltőprogram (külön menüpont) révén tölthető le a PLC-be. A 1131-3 szabvány háromféle grafikus szimbólumot alkalmaz: létradiagramos, funkcióblokkos és sorrendi folyamatábrán alapuló programnyelvet. A szabvány a 8. ábrán látható PLC nyelveket definiálja és ajánlja. [19]



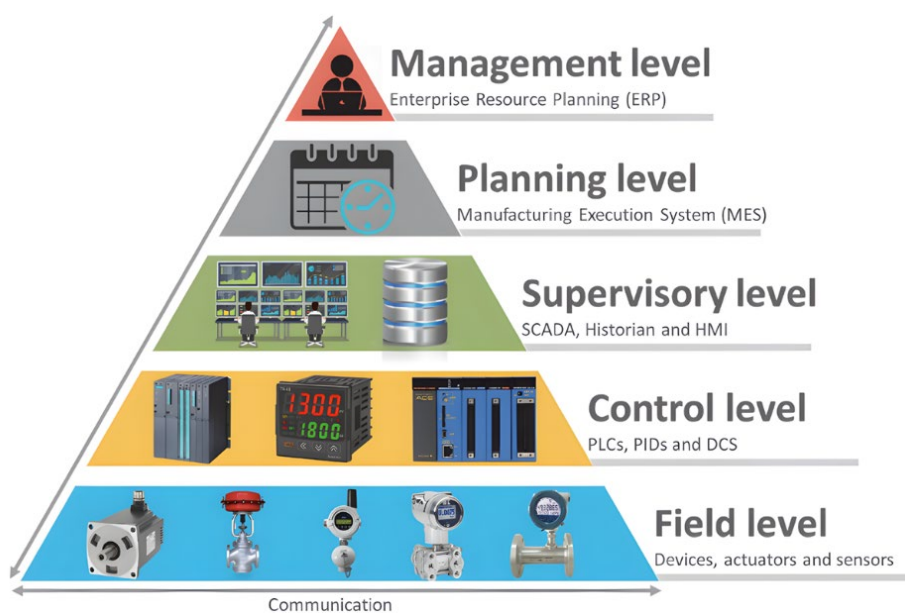
7. ábra: PLC programnyelvek [19]

A létradiagramos programozási nyelv (Ladder Diagram, LD vagy LAD) az áramútervből alakult ki, így a létradiagramok az áramútervek szoftveres megfelelői. Bár a létradiagramok (LAD) az áramútervek programozási megfelelői, lényeges különbség közöttük, hogy csak logikai egyenletek leírására szolgálnak, nem tartalmaznak speciális, és csak az érintkezős hálózatokra jellemző összevonásokat és elemei csak az adott PLC funkcionális szoftverelemei lehetnek. A létradiagramok főbb elemei:

- kontaktusok;
- huzalozás;
- logikai kimenetek (Output, Flag, MERKER);
- időzítők;
- számlálók;
- különleges elemek (pl. léptetőregiszter, PID blokk). [18]

Kommunikációs protokollok

Ahhoz, hogy bármilyen ipari rendszerről beszéljünk, elengedhetetlen a kommunikáció megemlítése, legyen szó akár a gép-gép (M2M) kommunikációról, akár a kommunikációs piramis (9. ábra) bármely szintje (ERP (Enterprise Resource Planning) - MES (Manufacturing Execution Systems) - HMI (Human Machine Interface) - PLC (Programmable Logic Controller) - Factory floor) közötti kommunikációról, például. CoAP (Constrained Application Protocol), CAN (Controller Area Network), ProfiBus (Process Field Bus), ProfiNET (Process Field Network), EtherCAT (Ethernet for Control Automation Technology), HART (Highway Addressable Remote Transducer), MODBUS, REST (Representation State Transfer), MQTT. [20]



8. ábra Kommunikációs piramis [21]

MQTT

Az MQTT (Message Queue Telemetry Transport) egy M2M vagy IoT kommunikációs protokoll, amelyet az IBM 1999-ben fejlesztett ki egyszerű, publish/subscribe alapú üzenetküldésre. A tervezett elv a hálózati sávszélesség és az eszköz erőforrásigény minimalizálása, miközben igyekszik biztosítani a biztonságot. Emellett fontos előnye, hogy a kialakított kapcsolatok nyomon követhetők, mindenféle adat átmegy rajta, és könnyen megvalósítható. Ezeknek a tulajdonságoknak köszönhetően gyorsan elterjedt az IoT világában, bár kezdetben csak a gépek közötti kommunikációra használták, de mára már szabványossá vált. [20]

PROFINET

A PROFINET egy nyílt ipari ethernet szabvány az automatizáláshoz. Két változat létezik belőle a PROFINET IO és a PROFINET CBA. A PROFINET IO a moduláris automatizálási koncepció kivitelezéséhez integrálja az elosztott I/O-val való valós idejű kommunikációt. Ezáltal a terepi buszok funkcióit valósítja meg és lehetővé teszi a paraméterezést, diagnosztizálást, konfigurálást, valamint a riasztások kezelését. [22]

S7 kommunikáció

Az S7 kommunikációt, mint homogén SIMATIC kommunikációt, a gyártóspecifikus kommunikáció jellemzi a SIMATIC CPU-k között (nem nyílt szabvány). Az S7 kommunikációt migrációra és a meglévő rendszerekhez való csatlakozásra használják. Az S7 kommunikáció segítségével a CPU adatokat cserél egy másik CPU-val, amely során a vételi adatokat automatikusan nyugtázza a küldő CPU miután a felhasználó megkapta azokat. Az S7 kapcsolatok konfigurálhatók az egyik vagy mindkét végén. A kommunikáció kétféleképpen megvalósítható. A CPU-ba integrált PROFINET vagy PROFIBUS DP interfészen vagy CP/CM interfészen. [22]

A mindkét végén konfigurált S7 kommunikáció során a kommunikáció paraméterei mindkét partnernél megtalálhatók. Ez a fajta konfiguráció lehetővé teszi PUT/GET függvények használatát, amelyek az adatok írására és olvasására használhatók a két CPU között. [22]

OPC UA

Az IIoT megvalósításának fontos szempontja a rendszerhez csatlakoztatott eszközök adatgyűjtése, az átvitelre használt különböző protokollok és az automatizáltan feldolgozott adatok fogadása. A Message Queue Telemetry Transport (MQTT) átviteli protokoll az IIoT szempontjából nagyon fontos szabvány lett; alapvetően a kis súlyú üzenetek szállítására szolgál, ezért támogatja az alapvető használatú eszközöket vagy érzékelőket. Az utóbbi időben azonban az intelligens iparban az interakcióhoz gyakrabban használt és ajánlott interfész az OPC-UA (Open Platform Communications Unified Architecture), amelyet a klasszikus OPC utódjaként javasoltak. – [24]

Az OPC UA szabvány különösen alkalmas különféle szintű adatcserére az operációs rendszerektől való függetlenségének, biztonságos továbbítási eljárásainak és az adatok szemantikai leírásának köszönhetően. A gépi adatok (vezérelt változók, mért értékek vagy paraméterek) szintén ilyen módon továbbíthatók. Ennek a koncepciónak fontos eleme, hogy az OPC UA kommunikáció párhuzamosan valósulhat meg a valós idejű kommunikációval az

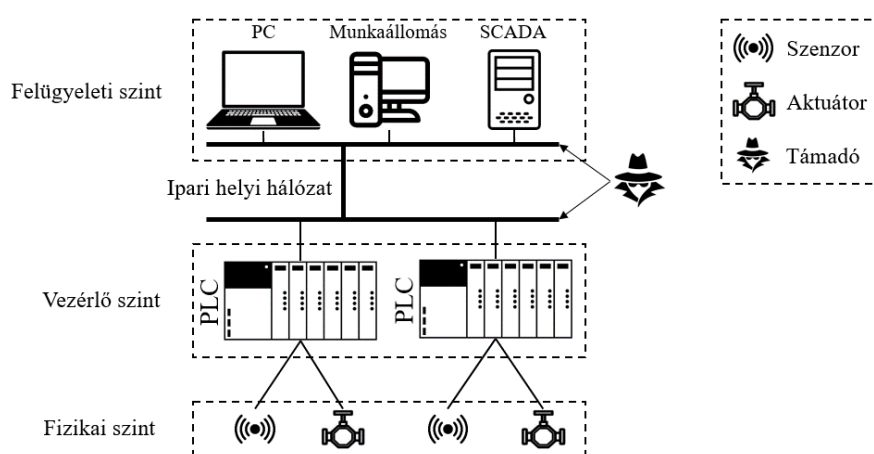
időkritikus, gépi szintű adatátvitel érdekében. Az OPC UA nagymértékben skálázható, így következetes információcsere lehetséges az érzékelők, vezérlők és MES vagy ERP rendszerek között. Az OPC UA nemcsak adatokat, hanem az adatokkal (adattípusokkal) kapcsolatos információkat is elérhetővé teszi, ugyanakkor lehetővé teszi az adatok gépileg értelmezhető hozzáférését. [23]

Az automatizálási berendezések és megoldások általában hosszú évekig képesek üzemelni, ezért sok régi rendszer létezik. A helyi örökölt rendszerek működésébe való beavatkozás elkerülhető. Az iparágat az foglalkoztatja, hogy ezeket a rendszereket interoperábilissá tegye és magasabb szintű felügyeleti alkalmazásokba integrálja. Az iparág azonban az egységek közötti együttműködéssel (ami horizontális kommunikációt és kapcsolódást is jelent), illetve a saját megoldások nagyobb rugalmasságának és alkalmazkodóképességének elérésével is foglalkozik. Ez azt jelenti, hogy a helyben megvalósított OPC UA szerverek (pl. a helyi automatizálási panelbe integrálva) érdekesebbek, mint a magasabb szintű PC-n elhelyezett központosított OPC megoldások. Ezért a helyi automatizáláshoz való nem invazív kapcsolódás, az örökölt rendszerprotokollok átalakítása, valamint a horizontális és vertikális átjárhatóságra alkalmas helyi OPC UA szerver megvalósítása követelményeinek ötvözéséhez egy middleware megoldásra van szükség. [25]

A szakirodalomban OPC UA-alapú köztesszoftver-megoldásokat kutatnak az ipar számára a helyi berendezések interoperabilitásának biztosítása érdekében. Az egyik ilyen lehetséges köztesszoftver megoldás a Node-RED. [25]

3. Alternatív ipari titkosítás fejlesztése

Az alternatív ipari titkosítás létrehozásának a célja, hogy egy olyan iparban is használható adattitkosítási megoldást valósítsak meg, amely a meglévő kommunikációs protokollok védelmén túl, további védelmi réteget képez. A dolgozatom során speciálisan az ipari vezérlőrendszerek magjára, vagyis a PLC-kre összpontosítok. A PLC-k képezik a kommunikációnak az egyik végpontját, ezért potenciális célpontjává váltak a kibertámadásoknak. Én egy olyan lehetséges esetet vettem figyelembe (9. ábra), amikor egy támadó hozzáfér az ipari eszközök közötti kommunikációhoz, ahonnan érzékeny adatokat képes kiolvasni.



9. ábra Ipari vezérlőrendszerek rendszermodellje

A PLC-k egyik legnagyobb előnye a hosszú élettartam, ami az internet világában egy problémának a forrása. Ezek az eszközök egyre gyakrabban kerülnek belső ipari hálózatokra, hogy fejlettebb IIoT eszközökkel kommunikáljanak, viszont nem támogatják a korszerű biztonsági funkciókat, ezáltal még inkább könnyű célponttá válnak a támadók számára.

3.1. Felhasznált eszközök és szoftverek

TIA Portal V17

A Totally Integrated Automation Portal (TIA Portál) mérnöki keretrendszert biztosít az automatizálási megoldások megvalósításához a világ minden iparágában. A tervezéstől, üzembe helyezéstől, üzemeltetéstől és karbantartástól kezdve az automatizálási rendszerek korszerűsítéséig a TIA Portal időt, költséget és erőfeszítést takarít meg a mérnököknek. A SIMATIC STEP 7 a TIA Portalban az összes moduláris és PC-alapú SIMATIC vezérlő konfigurálására, programozására, tesztelésére és diagnosztizálására szolgáló szoftver. [26]

Siemens S7-1200 PLC

Az S7-1200 sorozat a programozható logikai vezérlők (PLC-k) sorozata, amely számos automatizálási alkalmazás vezérlésére alkalmas. A kompakt kialakítás, az alacsony költség és a nagy teljesítményű utasításkészlet az S7-1200-as sorozatot tökéletes megoldássá teszi a legkülönbözőbb alkalmazások vezérlésére. A CPU egy mikroprocesszort, egy integrált tápegységet, bemeneti és kimeneti áramköröket, beépített PROFINET-et, nagy sebességű mozgásvezérlő I/O-t és fedélzeti analóg bemeneteket kombinál egy kompakt házban, hogy egy nagy teljesítményű vezérlőt alkosson. A CPU továbbá PROFINET portot biztosít az ipari hálózaton keresztüli kommunikációhoz. [27]

Node-RED

A Node-RED egy nyílt forráskódú, Node.js alapú JavaScript-alapú fejlesztőkörnyezet, amelyet az IBM mérnökei fejlesztettek ki, és amely leginkább IoT-rendszerek fejlesztésére alkalmas. Pontosabban egy folyamatalapú virtuális programozási környezet, amely a hardver és a szoftver összekapcsolásával "adatfolyamokat" hoz létre az érzékelőtől a felhőig. [20]

Eredetileg ez csak egy koncepciópróba (PoC - Proof of concept) volt, hogy segítsen megjeleníteni és megérteni a Message Queue Telemetry Transport (MQTT) témák közötti leképezést, de hamarosan egy nagyon népszerű eszközzé vált, amelyet könnyen ki lehetett terjeszteni különböző felhasználási célokra. [28]

Alkalmas adatfolyamatok írására, így megkönnyíti az adatfeldolgozást. Segítségével könnyen összeállítható adatfeldolgozási logika, és a feldolgozott adatok percek alatt átvihetők magasabb szintű rendszerekbe (SQL-szerver, vállalatirányítási rendszer, központi adatgyűjtő, felhőalapú szolgáltatás), vagy azonnal megjeleníthetők. A különböző érzékelőadatokat megjelenítő weblap programozása helyett a Node-RED egy Dashboard felületet biztosít, amely lehetővé teszi egy látványos felület létrehozását speciális HTLM, CSS ismeretek nélkül. [20]

A Node-RED három alapelemből áll: csomópont panel, áramlási panel, info és hibakeresési panel. A környezet alapvető fontosságú a valós idejű alkalmazások egyszerű létrehozásához és konfigurálásához. Rugalmasságának és az alkalmazások rövid idő alatt történő létrehozásának köszönhetően kiválóan alkalmas prototípusok készítésére. A Node-RED platform lehetőséget biztosíthat az ipar számára, hogy új prototípusokat építsen és teszteljen meglévő technológiákba építve vagy azokkal kombinálva, mielőtt jelentős erőforrásokat fektetne az innovációba. Az iparban eddig ezt a szerepet a TIA Portal, a Sysmac Studio vagy hasonló szoftverek látták el, de ezek hátránya, hogy használatuk megtanulása időigényes, és a licencük is nagyon drága. [20]

SiOME

Az OPC UA kommunikációs protokoll lehetővé teszi, hogy saját szabványokat alkossunk az OPC UA információs modellhez. Ezekkel a funkciókkal szabványosított szerver interfész hozható létre. A rendszerrel való kommunikációhoz az OPC UA kliensnek csak a meghatározott szabványt kell ismerniük, anélkül, hogy figyelembe venné a rendszerről és annak eszközeiről szóló részletes információkat. A platformfüggetlen OPC UA protokoll és a kísérő specifikációk kiterjesztése nagyban megkönnyíti a gyártók közötti szabványosított kommunikációt és csökkenti a megvalósítási erőfeszítéseket. [29]

Általánosságban elmondható, hogy az információs modellek az elkészített objektumtípusok, változók, referenciák, események és adatok halmaza. Az OPC UA alapinformációs modell fölötti első szintű információs modell új típusokat tartalmaz, amelyek az OPC UA alapinformációs modell alapvető típusaitól öröklődnek. A következő szintű információs modellek használhatják ezeket az új típusokat vagy az alapinformációs modell típusokat. Vannak kész általános információs modellek, amelyeket információs modell specifikációknak hívunk. Az OPC UA információs modell sok általános típust tartalmaz, amelyekre általában az OPC UA szerverek többségében van szükség. Ezek a típusok például olyan adattípusok, mint az egész szám és a karaktersorozat, vagy a hivatkozási típus, például a „has child – gyermeke van”. Így minden OPC UA szerver ugyanazzal az alapstruktúrával rendelkezik. Az információs modell meghatározza egy üres OPC UA kiszolgáló címtérét. Ez azonban nem jelenti azt, hogy minden szervernek meg kell adnia ezeket az alaptípusokat. Az OPC UA az öröklést használja a típusok meghatározásakor. Például minden adattípus öröklődik egy alapadattípustól, hasonlóan az eseményeknek, objektumoknak, referenciáknak és változóknak is megvan a saját alaptípusuk. [30]

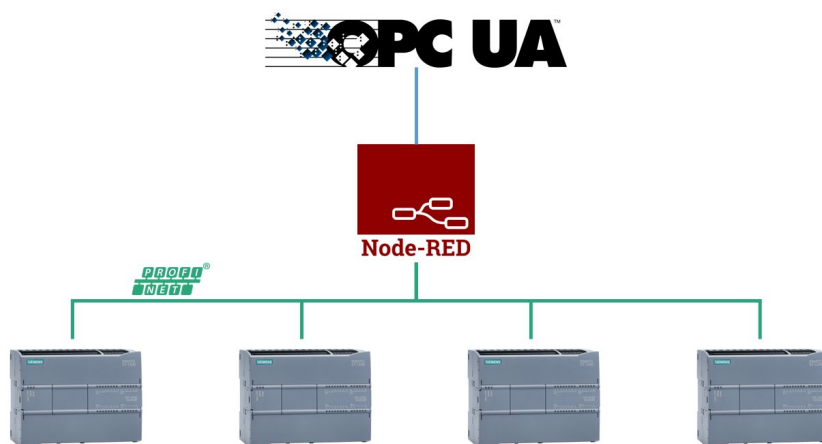
A Siemens OPC UA Modeling Editor egy olyan szoftver, amellyel lehetőség nyílik saját OPC UA információs modell létrehozására vagy meglévő információs modell módosítására. Az információs modellben meghatározott adatstruktúra összekapcsolható a TIA Portal-ban lévő adatstruktúrával, így biztosítva az összeköttetést az információs modell és a PLC között. [31]

Proslys OPC UA Client

A Proslys OPC UA Client egy általános célú OPC UA kliens. Segítségével kapcsolat létesíthető egy OPC UA szerverrel, amelynek az állapotáról visszajelzések láthatók, monitorozhatók az információs modellen lévő változók és parancsok hívhatók meg. Továbbá lehetőség van az adatok írására, amennyiben ez engedélyezve van a szerver oldalon. [32]

3.2. A fejlesztés folyamata

A célok megvalósításának első lépése, hogy egy kommunikációs hidat fejlesztek a Node-RED segítségével, amely a PLC-k és az OPC UA szerver kommunikációja között fog elhelyezkedni (10. ábra). Így a PLC-k által küldött adatok a helyi ipari hálózaton titkosítva jelennek meg, amiket a kommunikáció másik végpontján lévő OPC UA szervernek egy számítógép a Node-RED segítségével dekódolja, majd továbbítja azokat. A dolgozatom során az OPC UA szervert egy másik PLC-n fogom létrehozni, amivel szimulálni lehet egy felügyeleti szinten lévő OPC UA szervert, ezáltal a fejlesztés eredménye tesztelhetővé és bemutathatóvá válik.



10. ábra Kommunikációs híd

Az alternatív ipari titkosítás fejlesztés megvalósításához az Enigma felépítését veszem alapul. A szakirodalomban bemutatottak alapján látható, hogy az Enigma az angol ábécé betűit kódolta, viszont a Neumann-elven működő elektronikus eszközök a kettes számrendszert használják. Ez miatt mindenféleképpen át kell alakítani, oly módon, hogy az ipari PLC-ken is képes legyen működni. Mivel a PLC-k is kettes számrendszert használnak, ezért a betűk helyett számokat fogok alkalmazni. Az alapvető adattípusok közül én a WORD-öt választottam a titkosított adat bemeneti típusának, ami a TIA Portal-ban két byte-os, tehát 16 különböző bitből áll. Az egyszerűbb szenzorok az iparban bináris jelet biztosítanak, magyarul a PLC-kben egy digitális bemeneti helyet igényelnek. A 16 bit-es WORD változóval lehetőség van 16 különböző szenzor jelét lefedni. Az analóg szenzorjelekkel a dolgozatom során nem fogok foglalkozni, mivel lebegőpontos számábrázolás szükséges hozzá, ami szignifikánsan megbonyolítaná az eljárást és a dolgozatom során a működési elv bemutatásán van a hangsúly.

A titkosító algoritmusnak a bemenő adattípusának a kiválasztása után kimenő adattípust is meg kell határozni, amit érdemes egyazon típusúnak választani bemenő adattal a fölösleges

típuskonverzió műveletek elkerülése végett.

A fejlesztés több szakaszból fog állni. Először a PLC programozáshoz használt szoftverben létre kell hozni a titkosító függvényt, majd ezen belül az Enigmát alkotó elemeknek megfelelő függvényeket. Miután elkészült, önellenőrzés céljából érdemes a dekódoló függvényt is létrehozni. A kódoló-dekódoló függvények befejeztével célszerű a PLC-n egy külön függvényblokkot készíteni a kommunikációnak, mert a kiberbiztonsági elveket figyelembe véve csak azokat az adatokat érdemes érthetővé tenni, melyek elengedhetetlenek. Valamint egy külön kommunikációs függvényblokkon belül lesz az összes változó, ami egyszerűbbé teszi az adatsere folyamatát.

A következő szakasz az OPC UA szerveret biztosító PLC-nek a konfigurálása. Ehhez először a PLC-n egy egyszerű adat blokkot kell létrehozni, ahova a dekódolt WORD típusú változó érkezik. Ezután a SiOME szoftverrel kreálni kell egy információs modellt, amelyen belül szintén a dekódolt változó lesz. Az információs modell az elkészülte után feltölthető a PLC-re, ami által az OPC UA szerver elérhetővé válik.

Az utolsó szakasz a két eszköz közötti kommunikációnak az áthidalására szolgáló Node-RED konfigurálása és programozása. Először a kommunikációhoz szükséges könyvtárakat kell telepíteni, majd ezeknek a könyvtárak segítségével ki kell építeni a PLC és OPC UA közötti kommunikációt. Ezután következhet a dekódoláshoz szükséges függvény megírása. A titkosító függvény felépítéséhez hasonlóan az Enigma egyes alkotóelemeinek létre kell hozni a függvényét, majd ezeket fordított sorrendben egymás után felépíteni.

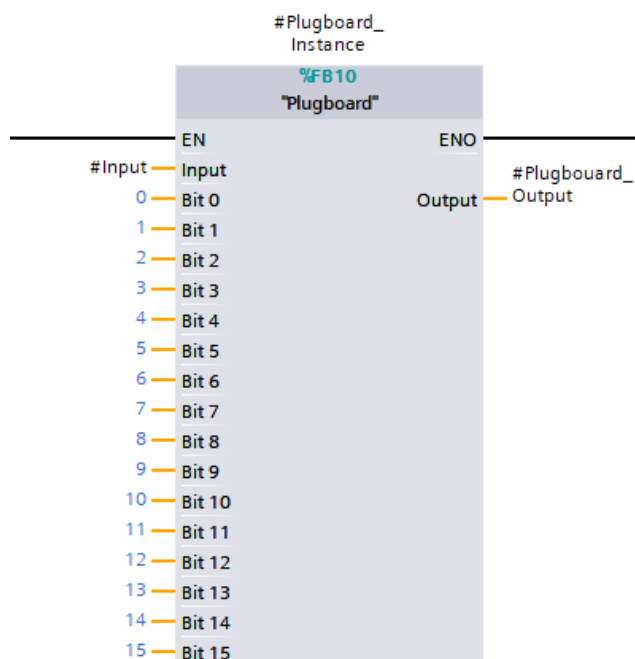
3.2.1. A titkosító függvény

Az első ilyen függvény előtt magát a titkosító függvényt kell létrehozni, amiben az egyes alkotóelemeknek a függvényei lesznek. Ezt a függvényt a TIA Portalban funkcióblokként hoztam létre, mivel saját adatblokkal rendelkezik és az elkészülte után a globális könyvtárban elmenthető, így bármikor egy másik projekt fájlban újra felhasználható. A titkosító függvénynek szüksége van még bemenetekre és kimenetekre. Az egyik bemenet nem más, mint maga a titkosítani kívánt WORD típusú adat. A másik bemenet a gomb lenyomását szimuláló Bool típusú változó. Jelen esetben ez egy belső órajel, ami az egyik legfontosabb különbséget képezi a függvényem és az Enigma között. A maradék három bemenet az Enigmán belül található rotoroknak a különböző féle előre elkészített variációinak a kiválasztására szolgál, amik 16-bit-es Integer típusú változók lesznek. A titkosító függvénynek csak egy kimenete lesz, ami a bemenethez hasonlóan egy WORD típusú változó. Ez lesz a kódolt adat. Miután elkészült

a keretfüggvény, következhetnek az alkotóelemek függvényei.

3.2.2. Kapcsolótábla

Az első alkotóelem a kapcsolótábla (Plugboard). Az Enigmában a kapcsolótábla használatakor a felhasználó be tudta állítani, hogy a gép belsejében zajló betűfelcseréléseken túl még milyen betűket cserélne meg. Mivel a bemenet egy 16 bit-es változó ezért a függvény a betűk helyett a változón belüli biteket fogja megcserélni. A függvény (11. ábra) a TIA Portal-on belül egy funkcióblokk lesz szintén, mert több funkcióblokk a szoftveren belül többszörösen összetetté válik, így továbbra is minden adat a titkosító függvényen belül marad. A kapcsolótábla függvénynek a titkosító függvénybe érkező bemenet lesz a bemenete, a kimenete pedig egy bitenként megkevert WORD típusú változó. Ahhoz, hogy meg lehessen határozni, hogy melyik bitet melyikkel szeretnénk megcserélni, további 16 bementre lesz szükség. Ezek a bemenetek 16 bit-es Integer típusúak és az értékük határozza meg, hogy a bementi WORD-nek az adott helyen lévő bitje melyik másik bitnek a helyére kerüljön. Az optimális helyfelhasználás miatt célszerű lenne Short Integert használni (8 bites), viszont vannak olyan PLC-k, amik nem képesek ennek a kezelésére.

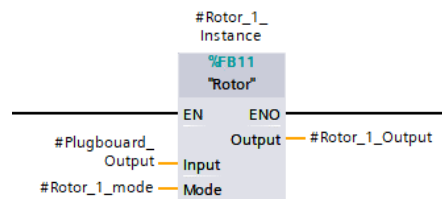


11. ábra A kapcsolótábla függvénye

3.2.3. Rotorok

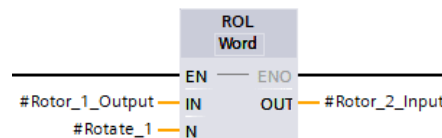
A következő alkotóelemek a rotorok. A rotorok a kapcsolótáblához hasonlóan funkcióblokkokként lettek létrehozva (12. ábra). A bemenete az első rotornak a kapcsolótábla

kimenete és a kimenet szintén egy WORD típusú változó. A második bemenete egy módválasztó, ami egy Integer típusú változó és az előre elkészített bit összekeverési sémák közül lehet kiválasztani egyet.



12. ábra A rotor függvénye

Az Enigmában egy gomb lenyomása után egy vagy több rotor fordult. A titkosító függvényen belül a forgatást, egy bitet balra forgató függvény (13. ábra) reprezentálja. Ezeknek az egyik bemenete az előttük lévő rotorok kimenete, a másik bemenet pedig egy 16-bit-es Integer, ami az forgatás mértékét határozza meg. Amint a titkosító függvénynek az órajel változóján pozitív felfutó élt észlel, növeli az forgatás mértékét. Az forgatás 0-15-ös értékig terjedhet, mivel utána már a bemenetet adja vissza. Amint az egyik forgatáshoz tartozó változó átlépi a 15-öt nullázódik, majd a következő eltoló függvény eltolási értéke növekszik 1-gyel.

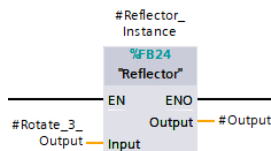


13. ábra Balra forgató függvény

A rotorok és az forgatások egymást felváltva követik az utolsó alkotóelem függvényéig. Ez a függvény a reflektor.

3.2.4. Reflektor

A reflektor függvényének (14. ábra) egy bemenete és egy kimenete van. A bemenet az utolsó forgatásnak a kimenete, a kimenete pedig már a titkosító függvény kimenete lesz. Az Enigmától eltérően a titkosító függvényben a reflektor után az adat nem megy végig az összes rotoron még egyszer. Ennek egész egyszerűen annyi az oka, hogy a dolgozatom során a célom az elvben rejlő lehetőségek bemutatása.



14. ábra A reflektor függvénye

3.2.5. Dekódoló függvény

A titkosító függvény elkészülte után önellenőrzés céljából létrehoztam a PLC-n belül a dekódoló mását. A dekódoló függvény bemeneteinek és kimeneteinek száma és típusa megegyezik a kódolóéval, valamint a függvényen belül ugyanazok az Enigmát alkotó elemeket képező függvények szerepelnek. A különbség csak annyi, hogy az alkotóelemeknek a függvényeit fordított sorrendben kell egymás után berakni, valamint a balra shiftelő függvényt az inverzére kell cserélni, vagyis jobbra shiftelőre. A fordított sorrendbe rakás esetén meg is jelent a funkcióblokkoknak az egyik előnye, mert ezeket a függvényeket nem kell újra létrehozni, hanem ismét felhasználhatók. Az ismételt felhasználása során egy újabb adat blokkot hoz létre, amit már a dekódoló függvény fog használni, így nem fordulhat elő az, hogy ugyanazon a memóriaterületen dolgozzanak.

3.2.6. Kommunikációs függvény

A kommunikációs funkcióblokk létrehozásánál a beállításain belül figyelni kell, hogy ne optimalizált data blokkot használjon, mert ebben az esetben a benne lévő változók szimbolikus elérhetőségük lesznek. Ez azt jelenti, hogy a nevük által lehet rájuk hivatkozni, nem pedig a memóriában elfoglalt helyük szerint, amit a Node-RED nem tud kezelni. Ezt figyelembe véve a 15. ábrán látható változókkal hoztam létre a funkcióblokkot:

	Name	Data type	Offset
1	Static		
2	Enigma_data	Int	0.0
3	Enigma_clock	Bool	2.0
4	Rotor_1_mode	Int	4.0
5	Rotor_2_mode	Int	6.0
6	Rotor_3_mode	Int	8.0
7	Rotate_1	Int	10.0
8	Rotate_2	Int	12.0
9	Rotate_3	Int	14.0
10	Enigma_start	Bool	16.0

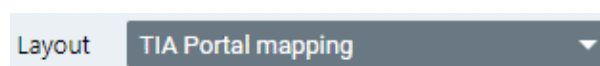
15. ábra A kommunikációs adat blokk

A kommunikációs funkcióblokk a következőféleképpen működik. Az elindításához a Node-

RED-től vár egy indító jelet, ami az ábrán „Enigma_start” névvel szerepel. Amikor a start jelen pozitív felfutó élt észlel, két másodpercen keresztül láthatóvá teszi a kommunikációban részt vevő eszközöknek az titkosító függvény beállításait, valamint elindítja a titkosító függvényt. A titkosító függvény a két másodperc letelte után elkezd küldeni a titkosított adatot a kommunikációs függvénynek, valamint a kommunikációs függvényen belül elindul egy fél Hz-es órajel. Az órajelnek fontos szerepe van. Ez határozza meg a titkosító függvénynek a lefutási gyakoriságát és tovább fokozza a titkosított adat felismerhetetlenségét. Az Enigma esetén minden betű lenyomása után érkezett a kimenetre egy másik betű, tehát ha nem nyomtak meg semmit, a gép beállítása magától nem változott. Az órajel segítségével a titkosító függvénynél ez máshogy van. A bemenő adatnak nem kell megváltoznia ahhoz, hogy kimeneten lévő adat megváltozzon, ezáltal még nehezebben kikövetkeztethető, hogy pontosan milyen adatok állhatnak a kódolt adat mögött. Így a titkosító függvény működését tekintve egy szimmetrikus kulcsú szinkron rejtjelezésről beszélhetünk.

3.2.7. OPC UA szerver létrehozása

Ahogy korábban említettem az OPC UA szerver egy másik Siemens PLC-n fog futni, ami az esetben egy magasabb szintű rendszert reprezentál. Ahhoz, hogy a dekódolt adatot fogadni tudja, létre kell hozni egy információs modellt, amihez egy adatblokk szükséges a PLC-n. Ezen a blokken belül egy változó lesz létrehozva a Node-RED által dekódolt adatnak, ami egy 16 bit-es Integer típusú változó. A változó létrehozása után a SiOME szoftveren belül meg kell alkotni az információs modellt. Ennek a legegyszerűbb módja, hogy a jobb felső sarkon a layout-ot a 16. ábrán látható módon állítjuk be.



16. ábra SiOME layout

Ezután lehetőség nyílik az aktív TIA Portal projekt megnyitásához. A projekt megnyitása után meg kell keresni az adat blokkot, rákattintani és a tőle jobbra lévő táblázatból a változót a CTRL gomb nyomva tartásával áthúzni az „Objects” mappába. Ilyenkor automatikusan létrehozza a SiOME a változót az információs modellben. Miután megtörtént a változó létrehozása a „Save as” gomb megnyomásával lementhető az információs modell. Az információs modellt ezután a TIA Portal-ban a PLC „OPC UA communication” füle alatt a „Server interfaces”-ben hozzá kell adni, majd a hozzáadást követően feltölthető a PLC-re és onnantól látható az OPC UA szerveren.

A láthatóságát a Prosys OPC UA Client szoftver segítségével lehet ellenőrizni, ahol meg kell nézni a változónak az információs modellben lévő azonosítóját (17. ábra), hogy a Node-RED-ben ezután meg lehessen adni.

Attribute	Value
► Nodeld	ns=4;i=6051

17. ábra A változó OPC UA azonosítója

3.2.8. Node-RED

Ahogy a fejlesztési szakaszoknál említettem, az első lépés a kommunikációhoz és a megvalósításhoz szükséges könyvtárak telepítése. Ezekben a könyvtárakban ún. node-ok találhatók. A node-ok a Node-RED fő építőelemei, amelyek jól definiált funkcionalitással bírnak. Mivel a Node-RED nyílt forráskóddal rendelkezik ezért bárki létrehozhat benne egyedi node-okat, amiket másokkal is megoszthat.

Az első könyvtár, amire szükség van, az a „node-red-contrib-s7”. A nevéből is látszik, hogy a Siemens S7 kommunikáció használatához lett tervezve. A következő könyvtár a „node-red-contrib-opcua”, ami az OPC UA-val való kommunikációhoz szükséges. Mivel a titkosító függvény bitműveletek használ, ezért a dekódoláshoz a Node-RED-en belül is bitműveleteket kell végrehajtani, amire önmagában nem képes, mivel nem erre lett tervezve. Az ehhez szükséges egyik könyvtár a „node-red-contrib-bit”. Ebben van egy olyan node, ami a 16bit-es beérkező jelet 16 különböző Boolean változóra bontja szét. A másik szükséges könyvtár a „node-red-contrib-edge-trigger”. Itt megtalálható a függvény működéséhez elengedhetetlen felfutó él vizsgálat. Ezáltal a függvény nem fog folyamatosan futni, csak akkor, amikor a beérkező órajel igazra vált. Az utolsó könyvtár, amit fel kell telepíteni az nem más, mint a „node-red-dashboard”. Ez a könyvtár egy UI-t hoz létre a Node-RED-en belül, ahol szemléltetni lehet a kódolás és dekódolás eredményét.

A könyvtárak feltelepítése után az első feladat a PLC-vel való kommunikáció kialakítása. Ehhez a Node-RED-en belül az S7 kommunikációhoz tartozó node-ot be kell húzni a munkaterületre és konfigurálni kell.

The screenshot shows the 'Connection' configuration window in TIA Portal. The 'Variables' tab is also visible. The configuration includes:

- Transport:** Ethernet (ISO-on-TCP)
- Address:** 192.168.191.60
- Port:** 102
- Mode:** Rack/Slot
- Rack:** 0
- Slot:** 1
- Cycle time:** 1000 ms
- Timeout:** 2000 ms
- Name:** (empty field)

18. ábra S7 kommunikáció konfigurálása

A 18. ábrán látható, hogy mely információk szükségesek a konfigurációhoz. Meg kell adni a PLC-nek az IP címét, a kommunikációhoz használt portot, a TIA Portal beállításában található Rack és Slot helyzetét, a ciklusidőt, az időtúllépést, valamint legutoljára nevet adhatunk ennek a node-nak. A konfigurálás után a node még nem tölthető fel, mivel a változók nevű fülön legalább egy változót meg kell határozni.

A kommunikációs adat blokkban lévő változók nevük, típusuk és az adat blokkban elfoglalt helyük alapján létre kell hozni ugyanezeket a változókat a változók listájában, ami a 19. ábrán látható.

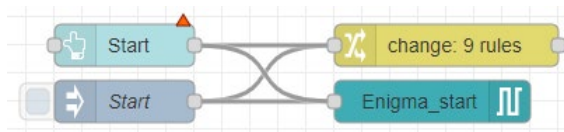
Variable list

DB28,I0	Enigma_data
DB28,X2.0	Enigma_clock
DB28,I4	Rotor_1_mode
DB28,I6	Rotor_2_mode
DB28,I8	Rotor_3_mode
DB28,I10	Rotate_1
DB28,I12	Rotate_2
DB28,I14	Rotate_3
DB28,X16.0	Enigma_start

19. ábra Változók listája

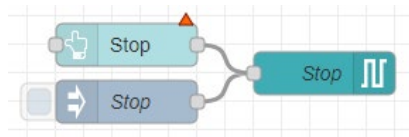
A PLC-vel való kommunikáció konfigurálása után következhet a dekódoló függvény megalkotása. Először ehhez ki kell alakítani a kommunikáció elindítására és megállítására szolgáló logikát. Az elindításhoz két különböző gombot hoztam létre, az egyikkel a Node-RED programozó felületén indítható el a kommunikáció a másikkal a UI felületen létrehozott start gombbal (20. ábra). A start megnyomása után a „change” node-al létrehozok két tizenhat bitből

álló tömböt és a kommunikációhoz szükséges adatokat és mindegyiket alapértelmezetten nullára állítok. Valamint az „s7 out” node-al a start jelet továbbítom a PLC felé.



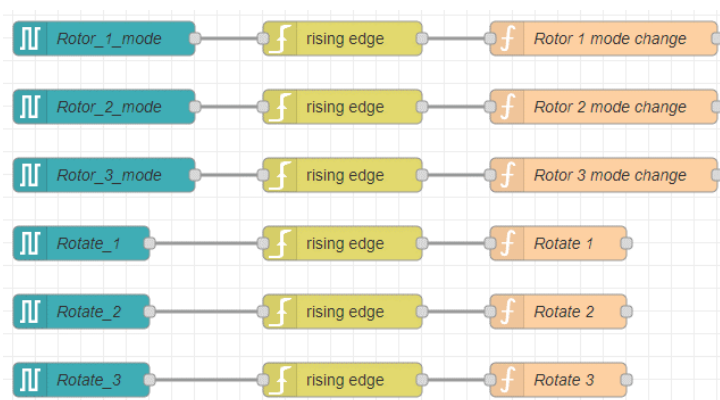
20. ábra Kommunikáció elindítása

A folyamat megállításához szintén két gombot hoztam létre, az egyiket a programozó felületen belül, a másikat a UI felületen (21. ábra). Bármelyik stop gomb megnyomása esetén a kimeneten a PLC-nek küldött start jel nullára állítása fog történni.



21. ábra Kommunikáció megállítása

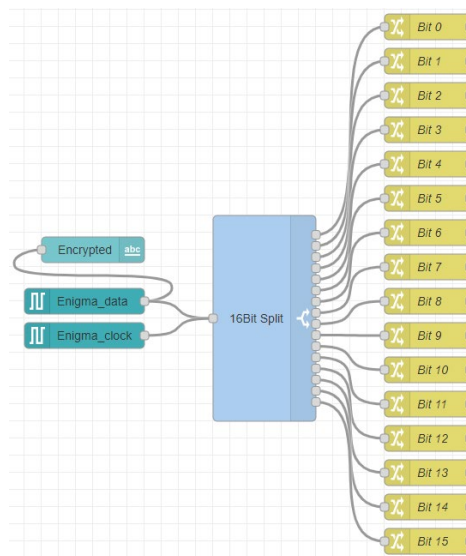
Az elindítás és megállítás létrehozása után következhet a két másodpercig küldött titkosító függvény beállításainak a lementése (22. ábra). Ehhez az „s7 in” node-ot kell felhasználni. Mindegyik kommunikációban résztvevő változót egyenként ki kell választani a node-on belül és annyi node-ot kell létrehozni, ahány változó van. Mivel a start jel esetén az összes változót alapértelmezetten nullára állítok, így a lementéshez a „rising edge” nevű felfutó él vizsgáló node-ot fogom használni. Ezáltal amikor nullától különböző érték érkezik, akkor a felfutó él vizsgáló tovább küldi a jelet az utána következő függvénybe, ahol elmenti a Node-RED a változók értékeit.



22. ábra Adatok lementése

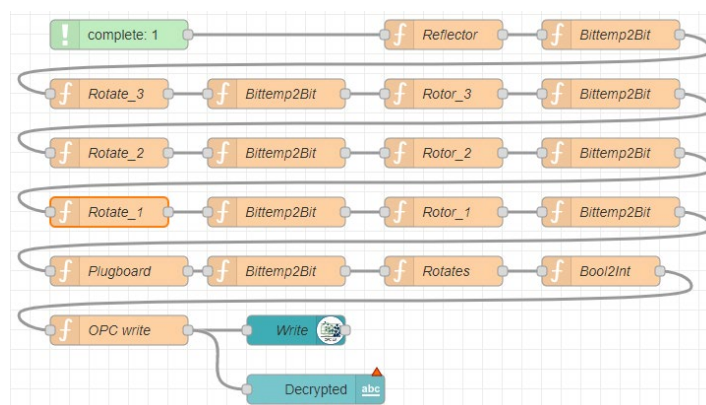
Az előkészületek után a következő lépés a dekódoló függvény megírása. Ehhez először az „s7 in” node-al be kell olvasni a start jel után érkező titkosított adatot és az órajelet, majd szétbontani a 16 bit-es WORD változót. Ehhez a „16 Bit Split” node szükséges (23. ábra). A

node beillesztése után a tizenhat kimenetére egyenként létre kell hozni egy „change” node-ot, aminek a segítségével az egyes biteket az előre létrehozott bit tömbbe el lehet menteni.



23. ábra Felbontás bitekre

Az elindítás után az órajelre vagy a bemenő adat megváltozására a függvény lefut és szétbontja bitekre a titkosított adatot. Az átláthatóság miatt célszerű a „complete” node alkalmazása, mellyel a jelfolyam átláthatóbbá tehető. Ez a node akkor aktiválódik, amikor a hozzákötött másik node befejezte a műveletét, ami jelen esetben a „16 Bit Split” node. Így a „complete” node után elkülönítve következhet a titkosító függvények az összes függvénye fordított sorrendben. Ez a sorrend a 24. ábrán látható.

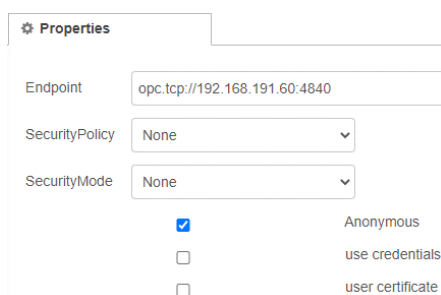


24. ábra A dekódoló függvény

A titkosító függvénynek a függvényein kívül az ábrán látható több „Bittemp2Bit” függvény, valamint egy „Bool2Int” függvény. A bitek mozgatásához létrehozott függvények először a biteket egy ideiglenes tizenhat bitből álló tömbbe kerülnek elmentésre, majd ezután a „Bittemp2Bit” függvény segítségével visszairódnak a helyükre. A legvégén, amikor minden bit

a megfelelő tömbben található, a „Bool2Int” függvény segítségével átalakítja egy 16 bit-es Integer típusú változóra.

Az átalakítás után az „OPC write” függvényen belül meg kell határozni, hogy az OPC UA szerverén melyik változóba kerüljön beírásra. Ezután az „OPC UA – Client” node-ot konfigurálni kell. A konfigurálás során meg kell adni az OPC UA szerver végpontját, magyarul az OPC UA-t biztosító szerver IP címét, valamint a kommunikációs port számát, ami a 25. ábrán látható. Az alatta lévő beállításokban lehetőség van a kommunikáció biztonságát beállítani, amennyiben az az OPC UA szerveren is megtörtént. Mivel az én esetemben az OPC UA szerver egy magasabb szinten lévő szervert reprezentál, tehát nem egy valós ipari hálózat része, így nem állítottam be semmiféle extra titkosítást.



The image shows a configuration window for an OPC UA Client. It has a title bar with a gear icon and the word 'Properties'. Inside, there are three main fields: 'Endpoint' with the value 'opc.tcp://192.168.191.60:4840', 'SecurityPolicy' with a dropdown menu showing 'None', and 'SecurityMode' with a dropdown menu showing 'None'. Below these fields, there are three radio button options: 'Anonymous' (which is checked), 'use credentials', and 'user certificate'.

25. ábra OPC UA szerver konfigurálása a Node-RED-en

A konfiguráció befejeztével feltölthető a Node-RED programja. A feltöltés után elindul a kommunikáció a PLC-vel és az OPC UA szerverrel és várja start jelet. A start megnyomása után a PLC két másodpercig küldi a titkosító függvénynek a beállításait, amit a Node-RED elment. A két másodperc lejártá után megadott órajelenként érkezik a titkosított adat, melyet dekódol és az OPC UA szervernek továbbít. A szerverre érkező dekódolt adat megegyezik a PLC-n lévő dekódoló függvény kimenetelével, tehát a fejlesztés sikeresnek mondható.

4. Eredmények értékelése és további fejlesztési lehetőségek

A fejlesztés során sikerült elérnem a kitűzött célokat, az alternatív ipari titkosítás alkalmazásának hatására a PLC és az OPC UA szerver közötti kommunikációban résztvevő adat titkosítottá vált. Ezenfelül sikerült egy olyan dekódoló megoldást létrehoznom, amely a kommunikáció két végpontja között helyezkedik el, így a közöttük lévő információcsere a kommunikációs protokollok védelméen túl az adat titkosításával is védve lett. Így hiába férne hozzá bárki ehhez a kommunikációban lévő adathoz, az teljes mértékben értelmezhetetlen.

A PLC-n létrehozott titkosító függvény a funkcióblokk mivolta miatt könnyedén implementálható további Siemens PLC-kben. Továbbá amennyiben egynél több változót szeretnénk titkosítani csak újból meg kell hívni a PLC-n belül a függvényt majd konfigurálni, ami nagyon minimális erőfeszítést igényel. Tehát nagyon könnyen alkalmazható.

A Node-RED-ben lévő dekódoló függvény a funkcióblokkokhoz hasonlóan egyszerűen újra felhasználható, így könnyen skálázható és a konfigurálása szintén gyorsan elvégezhető. A másik nagy előnye a Node-RED-nek, hogy nagyon sokféle IIoT eszközzel képes kommunikálni, így szinte bármilyen helyi ipari kommunikáció két végpontja közé könnyedén beilleszthető egy kommunikációs hidat képezve ezzel.

Végezetül pedig kiszámoltam az alternatív ipari titkosításnak a szakirodalomban taglaltak alapján a lehetséges beállítási variációit majd összehasonlítottam az ötletem alapját szolgáló Enigmával és a következőt kaptam eredményül:

A kapcsolótábla tizenhat bittel rendelkezik és egy bitnek a mozgatása esetén az adott bit egy másik bit helyére kerül, tehát maximum nyolc különböző mozgatás lehetséges. Ez alapján $\binom{16}{2m}$ különböző kombináció választható, ahol m a mozgatások számát jelenti. ($0 \leq m \leq 8$)

Amennyiben az első bitet mozgatnánk át egy másik bitbe a mozgatásnál $2m - 1$ szabad bit áll rendelkezésre. A második bit mozgatása esetén $2m - 3$ szabad bit van. Tehát az m mozgatás $2m$ szabad bitbe történő mozgatásának összes lehetséges módját a következő egyenlet adja meg.

$$(2m - 1) \cdot (2m - 3) \cdot (2m - 5) \cdot \dots \cdot 1 \quad (6)$$

Így az m -mozgatások összes lehetséges száma a következő:

$$\binom{16}{2m} \cdot (2m - 1) \cdot (2m - 3) \cdot (2m - 5) \cdot \dots \cdot 1 = \frac{16!}{(16-2m)! \cdot m! \cdot 2^m} \quad (7)$$

Tehát ebből az összefüggésből az összes lehetséges mozgatáskombináció száma minden m

értékhez a 2. táblázaton látható.

2. táblázat

m	Kombinációk
0	1
1	120
2	5 460
3	120 120
4	1 351 350
5	7 567 560
6	18 918 900
7	16 216 200
8	2 027 025

Mivel az itt összeállított kombinációk m minden egyes értékére kölcsönösen kizárják egymást, így a lehetséges mozgatókombinációk teljes száma a fenti esetek összegzésével adódik, vagyis

$$\sum_{m=0}^8 \frac{16!}{(16-2m)! \cdot m! \cdot 2^m} = 46\,206\,736 \quad (8)$$

A következő a sorban a rotorok, amelyek tizenhat bemenettel és tizenhat kimenettel rendelkeznek. $16!$ egyedi rotor variáció létezik. Ebből bármelyik kiválasztható az első szabad helyre. A középső pozícióba $16! - 1$ rotor választható. Az utolsó rotort pedig a még megmaradt $16! - 2$ rotorok bármelyikéből ki lehet választani. Tehát az összes lehetséges sorrendkombináció a következő:

$$16! \times (16! - 1) \times (16! - 2) \quad (9)$$

ami 9 159 226 129 830 100 000 000 000 000 000 000 000, azaz $9,16 \cdot 10^{39}$.

A függvény harmadik változó összetevője a három rotor kezdeti pozíciója. Mivel a három rotor mindegyike tizenhat különböző pozícióba állítható, a rotor kombinációinak száma 16^3 , azaz 4096.

A függvény negyedik változó eleme a rotor elfordulását szimuláló változó. Az első rotor elfordulásának az értéke minden pozitív felfutó órajelre növekszik. Tizenhat felfutó él múlva a középen elhelyezkedő rotort fogja elfordítani. A középső rotor $16 \cdot 16$ felfutó él múlva növeli eggyel az utolsó rotor elfordulásának az értékét. Mivel nincs több rotor, így a harmadiknak nincs hatása, tehát 16^2 , vagyis 256 kombináció lehetséges.

Az Enigma ötödik és egyben utolsó változó eleme a reflektor. A reflektornak annyi beállítási variációja létezik, mint a kapcsolótáblának az összes mozgatás felhasználása esetén, ami

$$\frac{16!}{8! \cdot 2^8} = 2\,027\,025 \quad (10)$$

Most már minden megvan a lehetséges beállítási konfigurációk számának a kiszámításához, ami a fentebb kiszámított öt érték szorzata:

899 545 408 387 617 000 000 000 000 000 000 000 000 000 000 000 000 000
ami körülbelül $9 \cdot 10^{59}$.

Az Enigma beállítási variációkhoz képest ($3 \cdot 10^{114}$) kisebb számot kaptam eredményül, de ez nem vonja azt maga után, hogy feltörhető, mivel ugye az Enigma is fel lett törve. Az Enigma feltörésének az egyik oka, hogy a beállítási variációk közül nem voltak képesek felhasználni mindet, hisz elég csak belegondolni, hogy ahhoz például az összes rotor variációt a katonáknak magukkal kellett volna vinni, ami $26!$ ($4,03 \cdot 10^{26}$) darab rotort jelent. Nyilvánvaló, hogy egy mechanikus gép esetén ez a nagy szám, csak egy elméleti maximum, nem pedig a valóság. Azonban az általam írt titkosító függvény esetén már nem elméleti maximum, hanem egy elérhető érték.

Ahhoz, hogy ki lehessen jelteni, hogy a titkosított adat brute force módszerrel nem törhető fel belátható időn belül, néhány számítás elvégzése szükséges. A világon a dolgozatom megírásának az időpontjában a nyilvántartott leggyorsabb szuperszámítógép a HPE Frontier, aminek számítási teljesítménye 1 102 TFLOP/sec, ami 1 102 000 000 000 FLOP/sec. [33]

Feltételezve, hogy körülbelül 1000 FLOP szükséges egy kombináció ellenőrzéséhez (becsült érték). [34]

Ezek alapján az általam fejlesztett eljárás feltöréséhez: $\frac{9 \cdot 10^{59}}{102\,000\,000} = 8,82 \cdot 10^{51}$ másodperc szükséges, ami $2,8 \cdot 10^{45}$ év.

További fejlesztési lehetőségek

Ahogy az előző fejezetben említettem, a titkosító függvény összes lehetséges variációjának a száma az esetemben egy elérhető érték, viszont még nem elért. Ennek az oka, hogy a rotoroknak, valamint a reflektornak nem készült el az összes létező variációja, hanem azok közül csak néhány. Így a későbbiekben ezeket a függvényeket át lehetne alakítani, hogy ténylegesen az összes létező beállítási variáció közül lehessen választani.

Valamint a titkosító függvény még optimalizálásra szorul, mivel a PLC-k kevés memóriával rendelkeznek, így nagy mennyiségű adatok titkosítása esetén könnyedén megtelhetnek.

5. Összefoglalás

A dolgozatomban egy alternatív ipari titkosítás fejlesztését mutattam be. A fejlesztés megkezdése előtt felkutattam és elemeztem a szükséges irodalmakat, majd ismertettem az elérni kívánt célt. Ezután bemutattam a fejlesztéshez felhasznált eszközöket és szoftvereket, valamint a fejlesztési folyamatnak a különböző szakaszait.

Következő lépésként végrehajtottam az eltervezett fejlesztéseket. Először a PLC-n megalkottam a titkosító függvényt, majd ezt követően a titkosító függvény belsejében található alfüggvényeket, melyek az Enigma alkotóelemeit reprezentálják. Ennek elkészültével létrehoztam egy dekódoló függvényt, az önellenőrzés céljából. Majd végül megcsináltam a kommunikációhoz szükséges függvényt.

Miután elkészültem a titkosító függvénnyel következhetett az OPC UA szerver létrehozása. Ehhez egy másik PLC-t használtam, hogy szimulálni tudjak egy felügyeleti szinten lévő OPC UA szervert. A szerver létrehozásához először megalkottam a kommunikációhoz szükséges adat blokkot, majd következett az információs modell létrehozása. Ennek elkészültével feltöltöttem a PLC-re és az OPC UA szerver elérhetővé vált.

Legvégül a Node-RED következett. Először felkészítettem a Node-RED-et a kommunikációra a két előbb említett eszközzel, valamint feltelepítettem a bitműveletekhez szükséges könyvtárakat. Az felkészítés után létrehoztam a PLC és az OPC UA szerver közötti kommunikációt, majd megalkottam a titkosító függvénynek a dekódoló függvényét. Ehhez a titkosító függvényben lévő, az Enigma alkotóelemeit reprezentáló függvényeket létrehoztam, majd fordított sorrendben egymás után elhelyeztem őket.

A fejlesztés befejeztével a tesztelés során minden az elvártak szerint működött. Sikerült megalkotnom egy olyan szimmetrikus kulcsú szinkron rejtjelező algoritmust, amely a belső ipari hálózaton belüli kommunikációnak a két végpontja között egy kommunikációs hidat alkot, ezáltal további biztonságot garantálva az adatcserének.

6. Irodalomjegyzék

- [1] Alp U., Emre C.: *Industry 4.0: Managing The Digital Transformation*, Springer, Switzerland, 293p., 2018
- [2] Alasdair G.: *Industry 4.0: The Industrial Internet of Things*, Apress, New York, 263p., 2016
- [3] SAP:
<https://www.sap.com/insights/what-is-industry-4-0.html>, 2023.01.11
- [4] Simone C., Marco P., Gianluigi F., Luca V.: *Internet of Things: Architectures, Protocols and Standards*, Wiley, Hoboken, 408p., 2018
- [5] Lane T., Dirk S.: *Cybersecurity for Industry 4.0 : Analysis for Design and Manufacturing* Springer, Switzerland, 265p., 2017
- [6] Krasznay Cs.: *Kiberbiztonság a negyedik ipari forradalom korában*, Nemzeti Közszerolgalati Egyetem, Híradástechnika: Hírközlés-informatika LXXIV, 5p., 2019
- [7] Ramjee P., Vandana R.: *Cyber Security: The Lifeline of Information and Communication Technology*, Springer, Switzerland, 312p., 2019
- [8] Sven S., Hamed S., Jesus M., JP LeBlanc, Frederick H., Marcellus B., Andrew G., Robert M., Harsha B., Shrinath E., Kaveri R., Andrew K., Qinqing Z., Peter M., Brian W.: *Industrial Internet of Things Volume G4: Security Framework*, CreateSpace Independent Publishing Platform, 174 p., 2016
- [9] Buttyán L., Vajda I.: *Kriptográfia és alkalmazásai*, Typotex Elektronikus Kiadó Kft., Budapest, 445 p., 2004.
- [10] Duncan B.: *Fundamentals of Cryptography: Introducing Mathematical and Algorithmic Foundations*, Springer, Switzerland, 296p., 2021
- [11] Gonda J.: *Kódolás és rejtjelezés*, Eötvös Loránd Tudományegyetem, Budapest, 127p., 2009
- [12] Dr. A. Ray M.: *The Cryptographic Mathematics of Enigma*, National Security Agency, 28p., 2019
- [13] Frank R.: *Secret Key Cryptography: Ciphers, from simple to unbreakable*, Manning, Shelter Island, 344p., 2022
- [14] YouTube:
<https://www.youtube.com/watch?v=ybkkiGtJmkM>, 2023.01.11
- [15] Kalika P., Munesh K.: *A review on mathematical strength and analysis of Enigma*,

Department of Mathematics, Central University of Jharkhand, India, 14p., 2020

- [16] Zheng Y., Zhiting B., Chenglu J., Zhe L., Jianying Z.: *PLCrypto: A Symmetric Cryptographic Library for Programmable Logic Controllers*, IACR Transactions on Symmetric Cryptology, pp. 170-217, 2021
- [17] Ajtonyi I.: *PLC és SCADA-HMI rendszerek*, AUT-INFO Kft., Miskolc, 290 p., 2007
- [18] Hodossy L.: *Programozott vezérlések I.*, Széchenyi István Egyetem, Győr, 121 p., 2006
- [19] Dr. Ajtonyi I., Dr. Gyuricza I.: *Programozható irányítóberendezések, hálózatok és rendszerek*, Műszaki Könyvkiadó Kft., Budapest, 525 p., 2010
- [20] Katalin F., József D.: *Using Node-RED platform in an industrial environment*, XXXV. Jubileumi Kandó Konferencia, Budapest, pp. 52-63, 2019
- [21] Izaro:
<https://www.izaro.com/la-automatizacion-y-las-pymes/c-1647784422>, 2023.01.11
- [22] Siemens:
https://cache.industry.siemens.com/dl/files/908/78028908/att_32073/v1/78028908_SIMATIC_Comm_DOKU_v21_e.pdf, 2023.01.11
- [23] Siemens:
https://cache.industry.siemens.com/dl/files/925/59192925/att_901175/v1/s71500_communication_function_manual_en-US_en-US.pdf, 2023.01.11
- [24] Teresa G, Sajid A., Marcelo L., Filipe J.: *Information and Knowledge in Internet of Things*, Springer, Switzerland, 486p., 2021
- [25] Silviu-Iulian T., Adrian K.: *Modbus-OPC UA Wrapper Using Node-RED and IoT-2040 with Application in the Water Industry*, 2018 IEEE 16th International Symposium on Intelligent Systems and Informatics, pp. 99-104, 2018
- [26] Mathworks:
https://www.mathworks.com/products/connections/product_detail/simatic-step7-tia-portal.html, 2023.01.11
- [27] Siemens:
https://cache.industry.siemens.com/dl/files/465/36932465/att_106119/v1/s71200_system_manual_en-US_en-US.pdf, 2023.01.11
- [28] Taiji H.: *Practical Node-RED Programming*, Packt Publishing, Brimingham, 326 p., 2021
- [29] Siemens:

[https://support.industry.siemens.com/cs/document/109755133/siemens-opc-ua-modeling-editor-\(siome\)-for-implementing-opc-ua-companion-specifications?dti=0&lc=en-MX](https://support.industry.siemens.com/cs/document/109755133/siemens-opc-ua-modeling-editor-(siome)-for-implementing-opc-ua-companion-specifications?dti=0&lc=en-MX), 2023.01.11

[30] Harju J.: *Plant information models for OPC UA: case copper refinery*, Tampere University of Technology, Automation Engineering, 2015

[31] Siemens:

https://support.industry.siemens.com/cs/attachments/109755133/109755133_SiOME_MAN_V2512_en.pdf, 2022.11.04

[32] Prosys:

<https://downloads.prosysopc.com/opcua/apps/UaBrowser/dist/4.0.6-268/Prosys OPC UA Browser UserManual.pdf>, 2023.01.11

[33] TOP500:

<https://www.top500.org/lists/top500/2022/06/>, 2023.01.11

[34] Akhilesh S., Sweta S, Rakesh K.: *Secure swipe machine with help of biometric security*, 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 53-61, 2016